UNIT - III

Database Application Security Models & Virtual Private Databases

- ✓ Introduction
- ✓ Types of Users
- ✓ Security Models
- ✓ Application Types
- ✓ Application Security Models
- ✓ Data Encryption
- ✓ Overview of VPD (**Virtual Private Databases**)
- ✓ Implementation of VPD using Views
- ✓ Application Context in Oracle
- ✓ Implementing Oracle VPD
- ✓ Viewing VPD Policies and Application contexts using Data Dictionary
- ✓ Policy Manager Implementing Row
- ✓ Column level Security with SQL Server

Introduction

- ✓ A Database user being used to log on (be authenticated) to an application

- ✓ For each application user , a database account must be created and assign specific privileges.

- ✓ **Application**
  - A program that solves a problem or performs a specific business function

- ✓ **Database**
  - A collection of related data files used by an applications

- ✓ **DBMS**
  - A collection of programs that maintain data files (Database)

Types of Users

- ✓ Application Administrator – Has application privileges to administer application users and their roles (do not require any special database privileges)
- ✓ Application owner – User who owns application tables and objects
- ✓ Application user – Perform tasks within the application
- ✓ DBA – Perform any administration tasks
- ✓ Database user- user account that has database roles and/or privileges assigned to it.
- ✓ Proxy user – User is employed to work on behalf of an application user (Proxy user is a user that acts as a substitute for another user to complete tasks.)
- ✓ Schema owner - User that owns database objects
- ✓ Virtual user – An account that has access to the database through another database account; a virtual user is referred to in some cases as a proxy user

Security Models

- ✓ There are two security models
  1. Access Matrix Model
  2. Access Modes Model

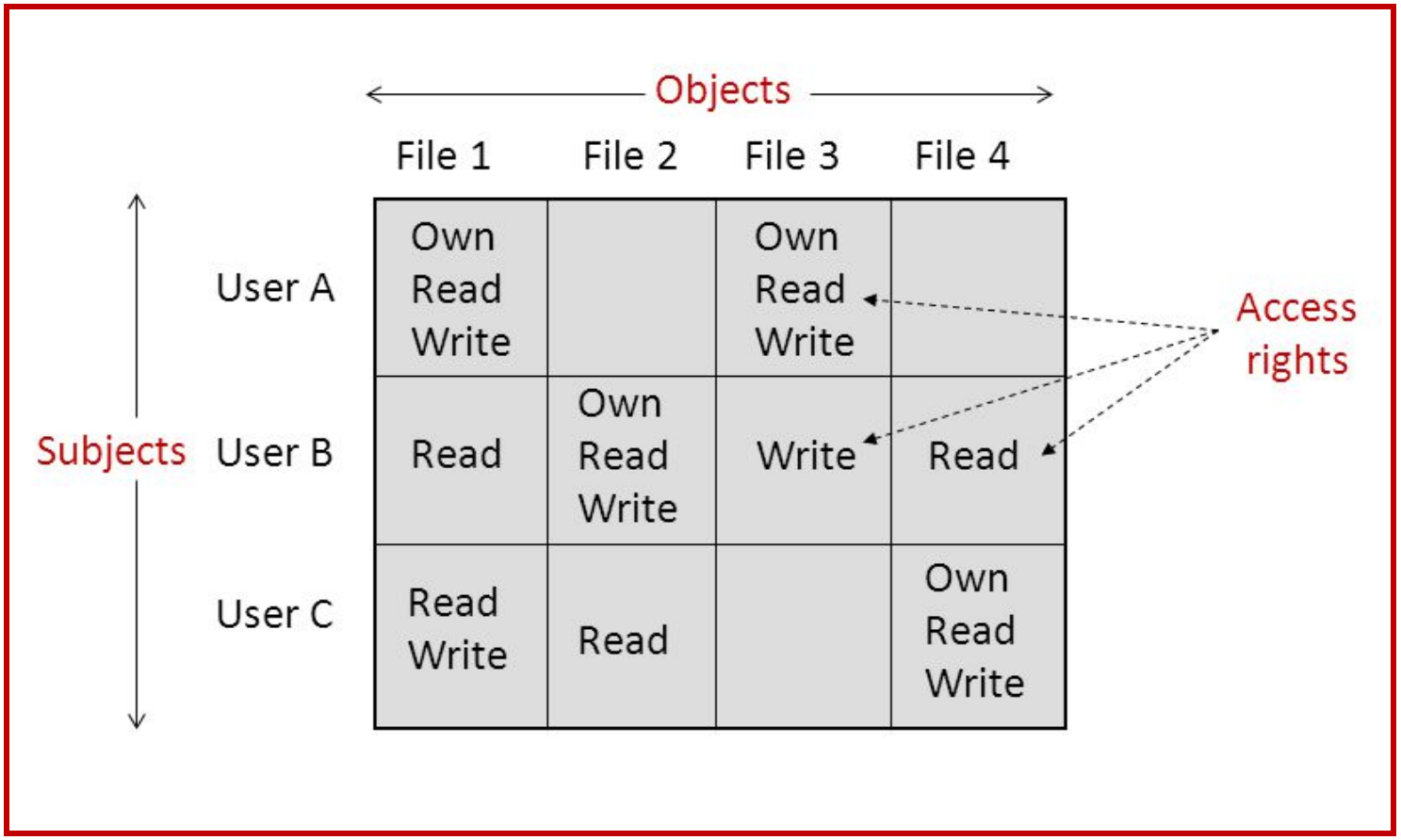
Security Models...

- ✓ Access Matrix Model
  - A conceptual model that specifies the right that each subject
  - possesses for each object
  - Subjects in rows and objects in columns

	Object 1	Object 2	. . .	Object m
Subject 1	Access [S1,01]	Access [S1,02]	. . .	Access [S1,0m]
Subject 2	Access [S2,01]	Access [S2,02]	. . .	Access [S2,0m]
. . .	. . .	. . .		. . .
Subject n	Access [Sn,01]	Access [Sn,02]	. . .	Access [Sn,0m]

Security Models...

Access Matrix Model - Example



Security Models...

Access Modes Model

- ✓ This model based on the Take-Grant models
- ✓ It uses both subject and object
- ✓ Object is the main security entity
- ✓ Access mode indicates that the subject can perform any task or not

There are two modes

- Static Modes
- Dynamic Modes

Security Models…

Access Modes – Static Modes

Access Mode	Level	Description
Use	1	Allows the subject to access the object without modifying
Read	2	Allows the subject to read the content of the object
Update	3	Allows the subject to modify the content of the object
Create	4	Allows the subject to add instance to the object
Delete	4	Allows the subject to remove instance to the object

Security Models…

Access Modes – Dynamic Modes

Access Mode	Level	Description
Grant	1	Allows the subject to grant any static access mode to any other subject
Revoke	1	Allows the subject to revoke a granted static access mode from the subject
Delegate	2	Allows the subject to grant the grant privileges to other subjects
Abrogate	2	Allows the subject to grant the revoke privileges to other subjects

Application Types

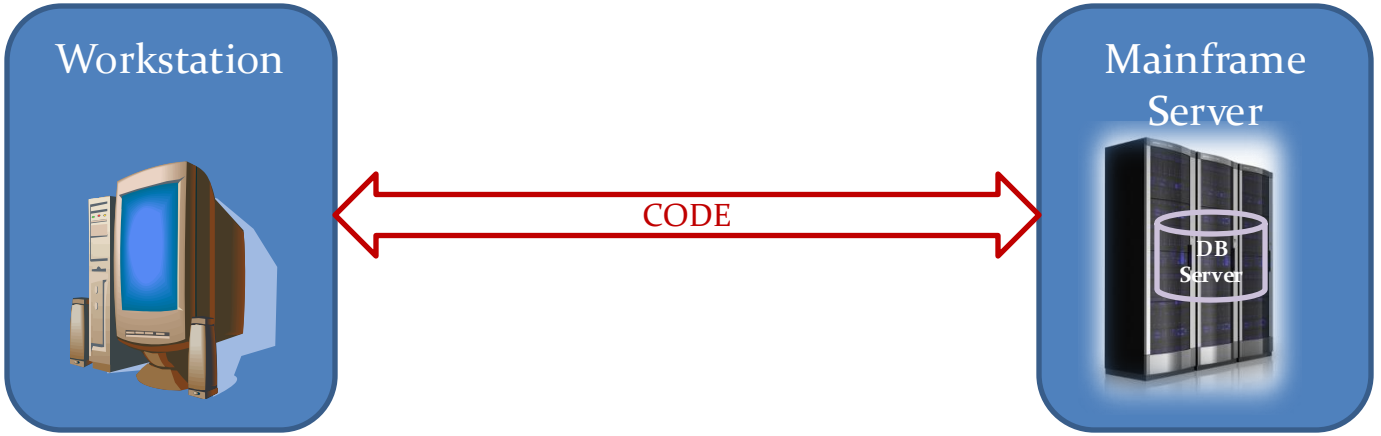
- ✓ Mainframe applications
- ✓ Client / Server Applications
- ✓ Web Applications
- ✓ Data warehouse applications

Application Types ...

Mainframe applications

- ✓ Years back computing in corporations was centralized in the Management Information System(MIS)
- ✓ MIS department is responsible for all information
- ✓ MIS mainly developed for Mainframe projects

The following figure is Mainframe application architecture



Application Types ...

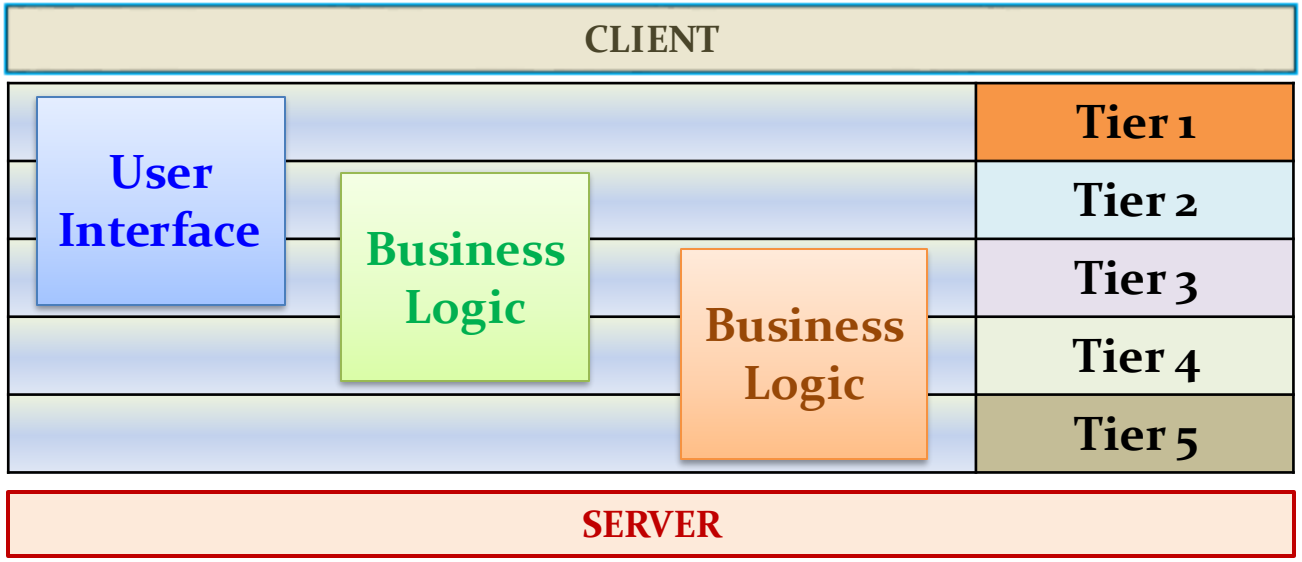
Client / Server Applications

- ✓ To overcome the limitations in MIS department the client / server architecture was introduced
- ✓ It is based on a business model, client request and the server respond
- ✓ Client / Server architecture became a dominating configuration for all applications
  - Flexible
  - Scalable
  - Processing power
- ✓ Three main components typically found in Client / Server architecture
  - User interface component – Represents all screens, reports, etc.,
  - Business logic component – Contains all the codes related to data validations
  - Data access component – Contains all the codes related to retrieves, inserts, deletes and updates

Application Types ...

Client / Server Applications

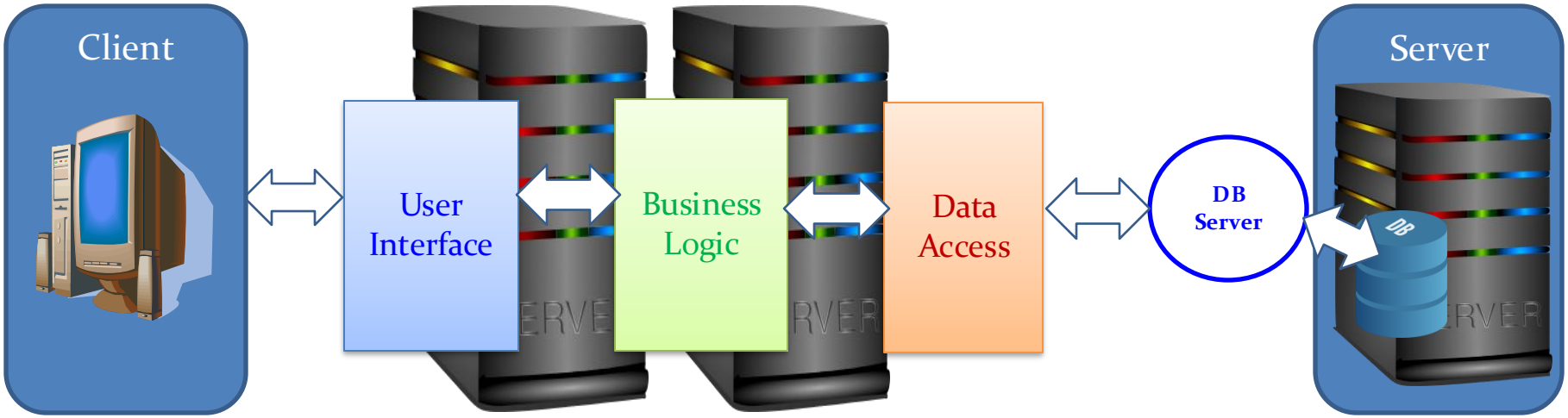
- ✓ A client / server application consists of minimum of two tiers .
- ✓ Normally four to five tiers is the maximum configuration
- ✓ The following figure represents the logical components of a client server architecture



Application Types ...

Client / Server Applications

- ✓ The following figure represents the physical architecture of a client/server application



- ✓ The data access component of client server architecture is the component responsible for retrieving and manipulating data.
- ✓ The security model should be embedded in this component.

Application Types …

Web Applications

- ✓ Client server application once dominated but not for long.
- ✓ Another architecture evolved with rise of dot-com and Web-based companies
- ✓ The new client / server architecture is based on the web and it is referred as a web application or a Web-based application
- ✓ Web application uses HTTP protocol to connect and communicate to the server.
- ✓ Web pages are embedded with other web services.
- ✓ The following figure represents the logic components of Web application architecture

CLIENT	
Web browser layer	Tier 1
Web server layer	Tier 2
Application server layer	Tier 3
Business logic layer	Tier 4
Database server layer	Tier 5
SERVER	

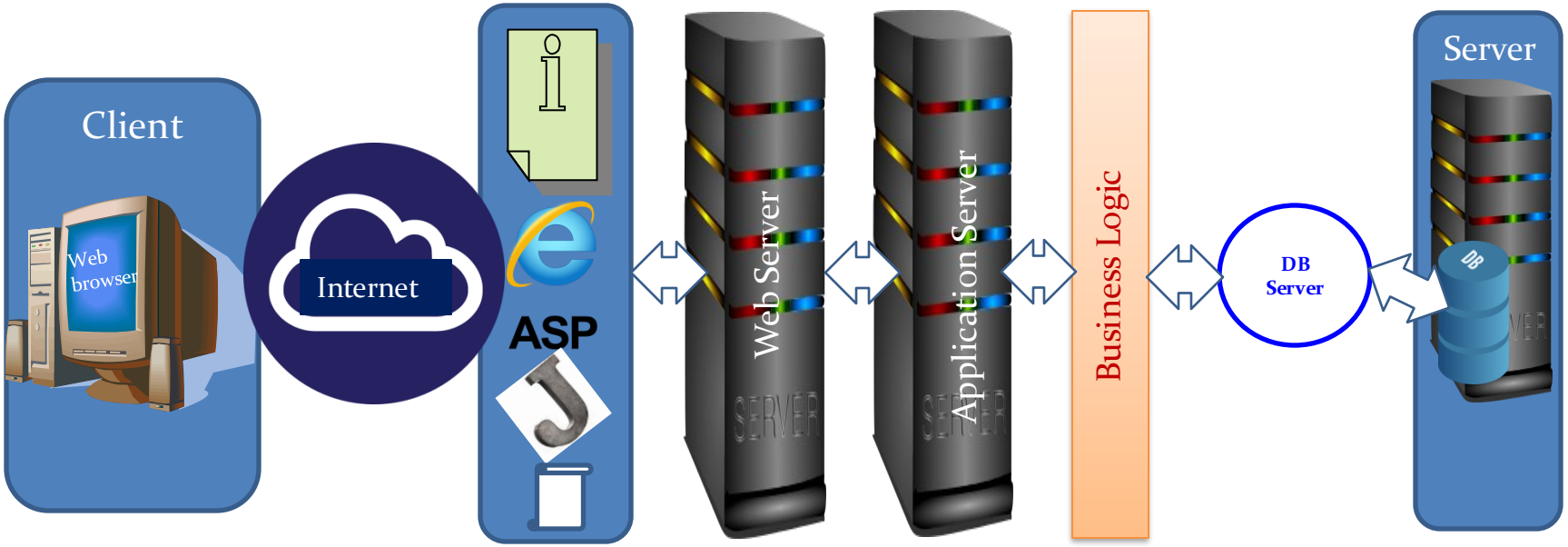
Application Types …

Components of Web application

- ✓ Web browser layer - A typical browser program that allows user to navigate through web pages found on the internet.
- ✓ Web server layer – A software program residing on a computer connected to Internet
- ✓ Application server layer - A software program residing on a computer that is used for data processing
- ✓ Business logic layer – A software program that implements business rules
- ✓ Database server layer – A software program that stores and manages data

Application Types ...

- ✓ The following figure shows a physical architecture that is typical for a web-based application.
- ✓ In this architecture , each layer resides on a separate computer
- ✓ One or more web application layers could be housed on one computer
- ✓ The main reason for separating web application layers to reside on different computers is to distribute the processing load



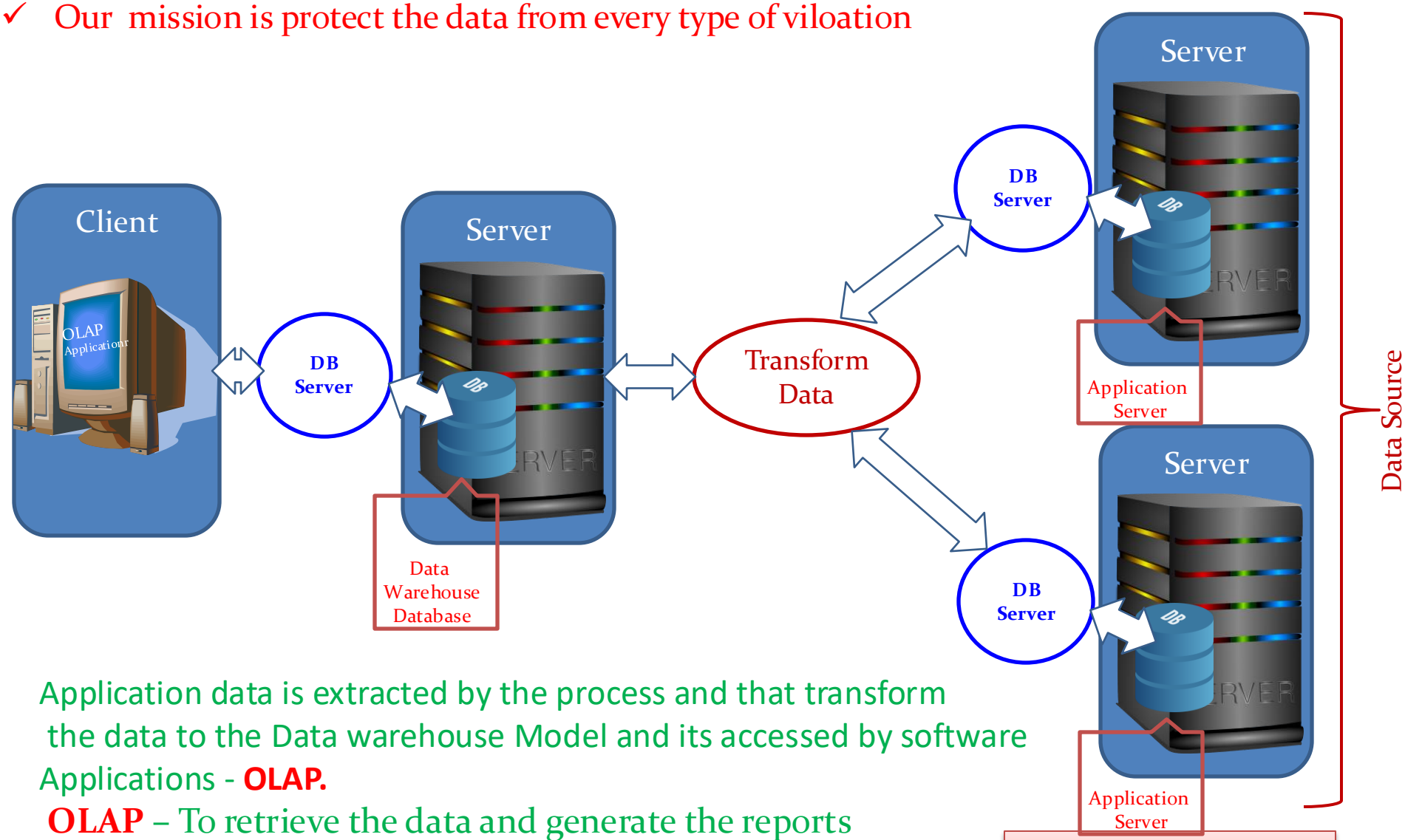
Application Types ...

Data Warehouse Applications

- ✓ DW is subject oriented , time variant, non volatile and integrated system.
- ✓ DWs are decision support system. (Support Executive Mgt for Decision Making processes.)
- ✓ DW is a collection of many types of data taken from different data sources.
- ✓ The architecture of these types of data warehousing applications is typically of a database server on which the application resides.
- ✓ The DW is accessed by software applications or reporting applications called OLAP (OnLine Analytical Processing)

Application Types ...

- ✓ The following figure shows the Physical and Logical structure of a data warehouse
- ✓ Our mission is protect the data from every type of viloation



Application data is extracted by the process and that transform the data to the Data warehouse Model and its accessed by software Applications - **OLAP.**

OLAP – To retrieve the data and generate the reports with capabilities of data drilling and data mining.

Application Security Models

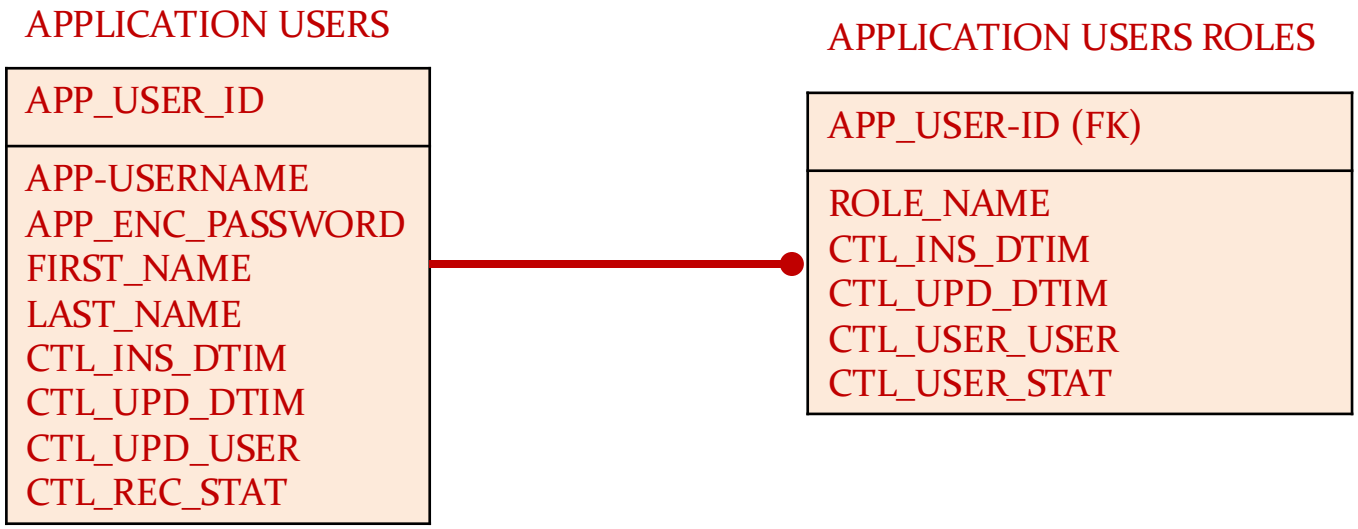
Five different application security models that are commonly used by the industry to provide data security and access protection at the table level (row and column level).

1. **Database role based**
2. **Application role based**
3. **Application function based**
4. **Application role and function based**
5. **Application table based**

Application Security Models …

I-Security Model based on Database Roles

- ✓ This model depends on the application to authenticate the application users by maintaining an end users in a table with their encrypted passwords
- ✓ In this model each end user is assigned a database role
- ✓ The user can access whatever the privileges are assigned to the role
- ✓ In this model proxy user needed to activate assigned roles
- ✓ The following figure shows the data model for this application (Security data model based on database roles)



CTL - Control

Application Security Models ...

The following list presents the a brief description of these columns

CONTROL COLUMN	DESCRIPTION
CTL_INS_DTIM (Control insert Date and Time)	Contains when date and time when the record was inserted
CTL_UPD_dtim (Control update Date and Time)	Contains when date and time when the record was last updated
CTL_UPD_USERS (Control update user)	Contains the username that created the record or last updated the record
CTL_REC_STAT (Control Record Status)	Can be used to indicate the status of the record

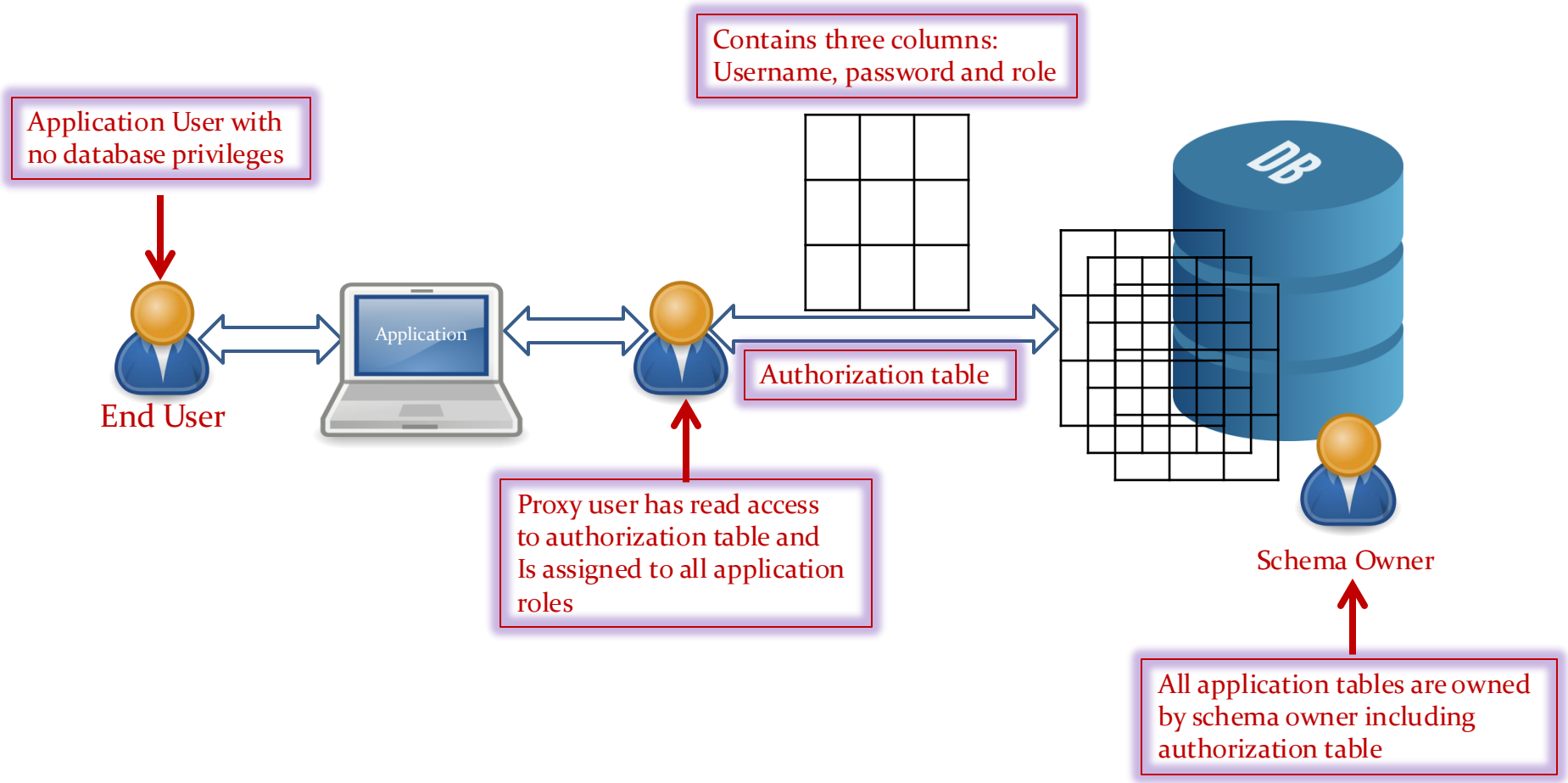
Application Security Models …

Tables used in security data model based on database roles

TABLE NAME	DESCRIPTION
APPLICATION_USERS	Stores and maintain all end users of the applications with their encrypted passwords
APPLICATIONS_USERS_ROLES	Contains all roles defined by the application and for each role that a privilege is assigned , the privileges can be read, write or read/write

Application Security Models ...

Architecture of a security data model based on database roles



Application Security Models …

The following points on this type of security model are worth noting:

- ✓ This model uses the DB role functionality
- ✓ Therefore it is DB independent
- ✓ If the roles are implemented poorly , the model does not work properly
- ✓ Privileges to table are also DB dependent
- ✓ Can isolate the application security from the DB
- ✓ Maintenance of the application security does not require specific DB privileges
- ✓ Password must be surely encrypted
- ✓ The application must use proxy users to log on and connect to the application database and activate specific roles for each database session

Application Security Models ...

Implementation in ORACLE

1. Creating the users by entering the following code:

Creating Application Owner

```
SQL > CREATE USER APP_OWNER IDENTIFIED BY APP_OWNER
      2  DEFAULT TABLESPACE USERS
      3  TEMPORARY TABLESPACE TEMP
      4 QUOTA UNLIMITED ON USERS;
User created

SQL> GRANT RESOURCE, CREATE SESSION TO APP_OWNER;
Grant succeeded
```

Creating Proxy User

```
SQL > CREATE USER APP_PROXY IDENTIFIED BY APP_PROXY
      2  DEFAULT TABLESPACE USERS
      3  TEMPORARY TABLESPACE TEMP;
User created

SQL> GRANT CREATE SESSION TO APP_PROXY;
Grant succeeded
```

Application Security Models …

Creating Application tables

```
SQL> CONN APP_OWNER@DB
Enter password : *********
Connected

SQL> CREATE TABLE CUSTOMERS
  2  ( CUSTOMER_ID NUMBER PRIMARY KEY,
  3  CUSTOMER_NAME VARCHAR2(50) );
Table created

SQL>  CREATE TABLE AUTH_TABLE
  2 (  APP_USER_ID NUMBER,
  3  APP_USERNAME  VARCHAR2(20),
  4  APP_PASSWORD  VARCHAR2(20),
  5   APP_ROLE  VARCHAR2(20) );
Table created
```

Application Security Models ...

II- Creating Application Roles

```
SQL> CONNECT SYSTEM@DB
Enter password: *******
Connected

SQL> CREATE ROLE APP_MGR;
Role created

SQL> CREATE ROLE APP_SUP;
Role created

SQL> CREATE ROLE APP_CLERK;
Role created

SQL> GRANT APP_MGR, APP_SUP, APP_CLERK TO APP_PROXY;
Grant succeeded

SQL> ALTER USER "APP_PROXY" DEFAULT ROLE NONE;
User altered
```

Application Security Models ...

Assign grants

SQL> CONNECT APP_OWNER@DB
Enter password : *********
Connected

SQL> GRANT SELECT,INSERT,UPDATE,DELETE ON CUSTOMER TO APP_MGR;
Grant succeeded

SQL> GRANT SELECT,INSERT,UPDATE,DELETE ON CUSTOMER TO APP_SUP;
Grant succeeded

SQL> GRANT SELECTON CUSTOMER TO APP_CLREK;
Grant succeeded

SQL > GRANT SELECT ON AUTH_TABLE TO APP_PROXY;
Grant succeeded

2. Add rows to the CUSTOMER table

```
SQL> CONN APP_OWNER@DB
Enter password : *********
Connected

SQL> INSERT INTO CUSTOMERS VALUES (1, 'John');
1 row inserted

SQL> INSERT INTO CUSTOMERS VALUES (2, 'RAM');
1 row inserted

SQL> COMMIT
Commit complete
```

Application Security Models …

3. Add a row for an application user called APP_USER:

```
SQL> INSERT INTO AUTH_TABLE VALUES (100, 'APP_USER'
'd323deq4fdfgdgg', 'APP_CLERK');
1 row inserted
```

4. Now assume that APP_USER is trying to log in through PROXY_USER. Your application should look up the role of the user by using the SELECT statement and activating that role:

```
SQL> SELECT APP_ROLE FROM AUTH_TABLE WHERE APP_USERNAME =
'APP_USER';
APP_ROLE
--------------------------
APP_CLERK
```

Application Security Models ...

5.Activate the role for this specific APP_USER session:

```
SQL> CONN APP_PROXYUSER
Enter password : **********
Connected

SQL> SET ROLE APP_CLERK;
Role set

SQL> SELECT * FROM SESSION_ROLES;
ROLE
------------------------
APP_CLERK
```

Application Security Models ...

Implementation in SQL Server

- ✓ In SQL Server 2000 you are using application roles.
- ✓ Application roles are the special roles you create in the database, that are then activated at the time of authorization.
- ✓ Application roles requires a password and cannot contain members
- ✓ Application roles are inactive by default
- ✓ Application roles can be activated using the **SP_SETAPPROLE** , system stored procedure

Creating Application Roles using the command line

- ✓ To create an application role in the Query Analyzer, use the SP_ADDPROFILE system-stored procedure

```
sp_addapprole [ @rolename = ] 'role', [@password =] 'password'
```

Where :

@rolename – The name of the application role (The value must be a valid identifier and cannot already exist in the database)

@password – The password required to activate the role. (SQL Server stores the password as an encrypted hash)

Example :

To create the application role of clerk for your Pharmacy database , use this command

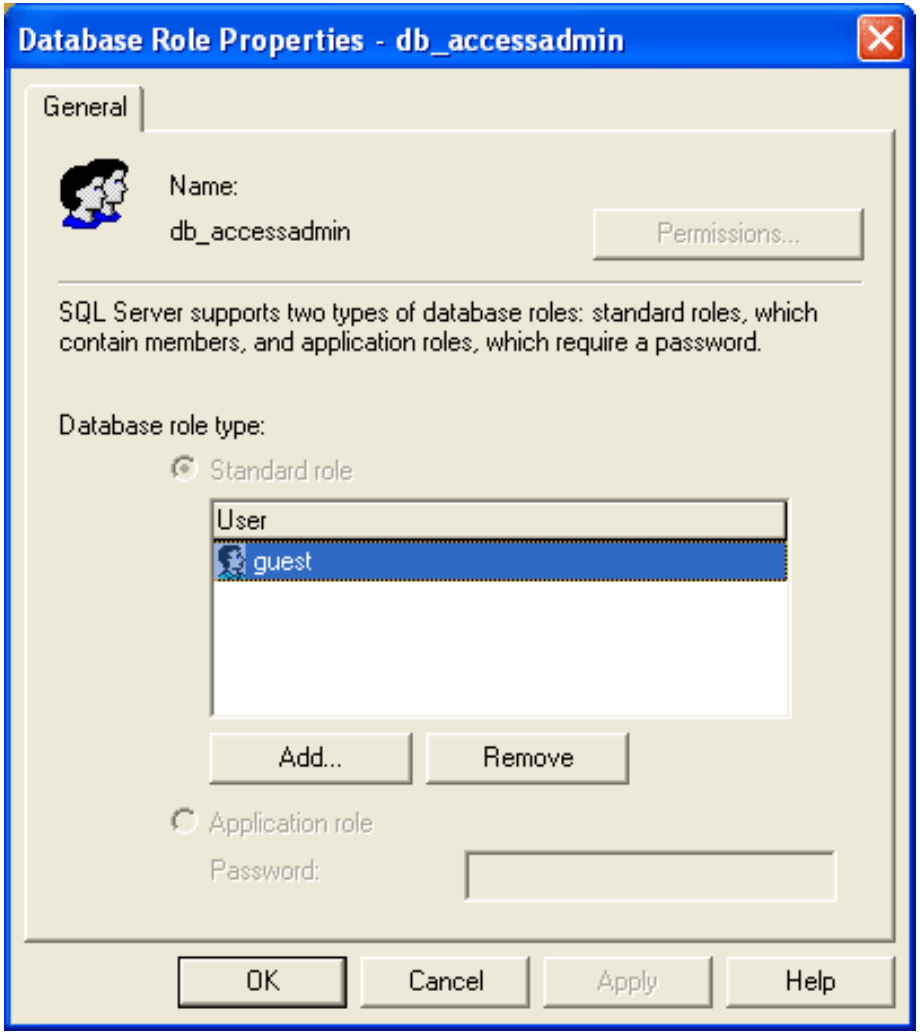
```
exec sp_addapprole 'clerk', 'Clerk@ccess'
```

Application Security Models …

Creating Application Roles using SQL Server Enterprise Manager

Follow the steps

1. Open Enterprise Manager
2. Expand the Role container for your Pharmacy database. Right click in the right pane, the select New Database Role
3. Type the name db_accessadmin in the name box
4. Select Application Role under Database role type
5. Enter password db@ccess in the text box
6. Click OK to create the role.



Application Security Models …

Dropping application Roles using Command line

- ✓ To drop an application role , using the Query Analyzer ,use the SP_DROPAPPROLE system-stored procedure

```
sp_dropapprole [@rolename = ] 'role'
```

Where

@rolename – The Application role to drop.

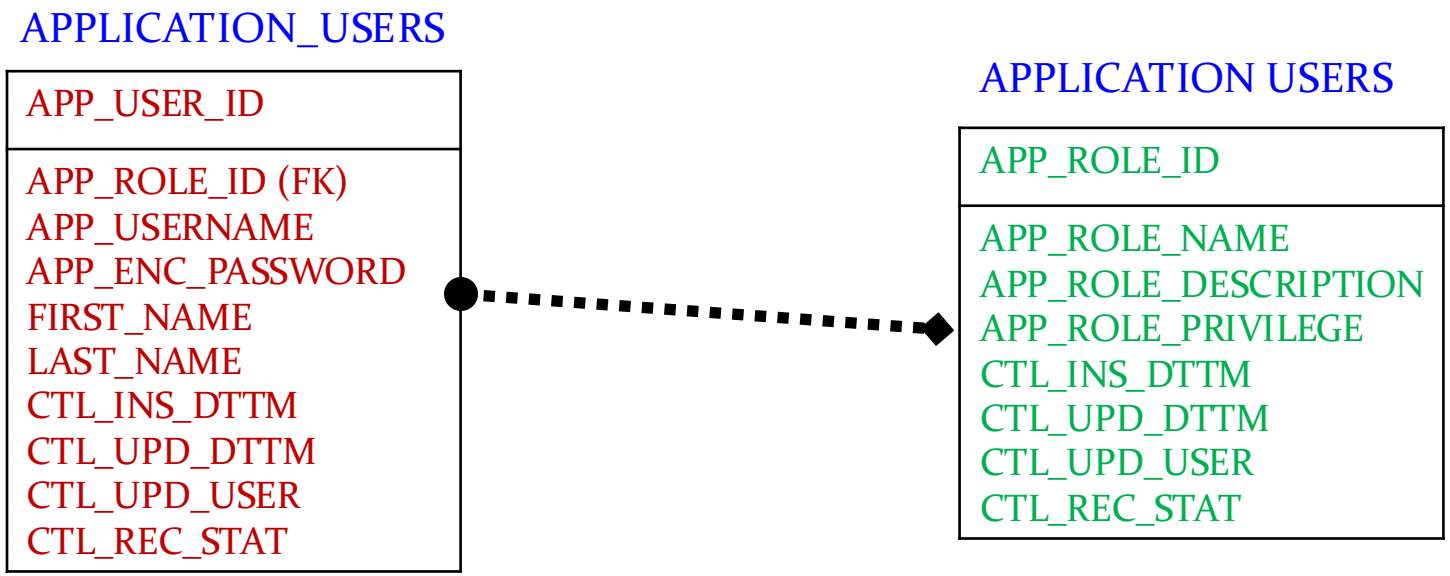
Dropping application Roles using Enterprising Manager

- ✓ Follow the steps
  1. Open Enterprise Manager
  2. Expand the roles container of the database from which you are dropping the role
  3. Select and Delete the desired role

Application Security Models ...

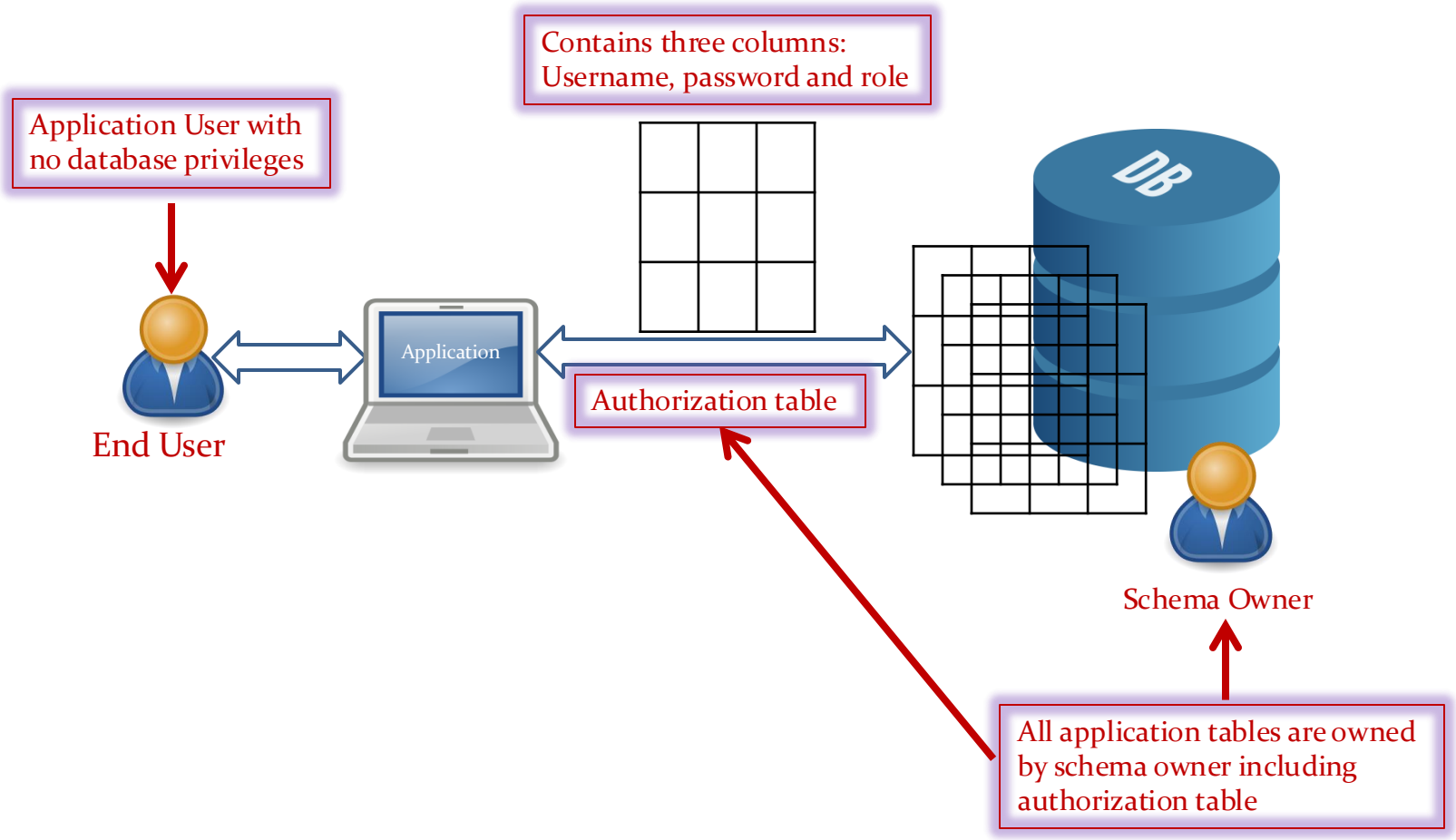
III - Security Model based on Application Roles

- ✓ Depends on the application authenticate the application users.
- ✓ Authentication is accomplished by maintaining all end users in a table with their encrypted passwords.
- ✓ Each end user is assigned an application **role to read / write specific modules** of the applications.
- ✓ The following table contains the description of tables used for this model.



Application Security Models ...

Architecture of Security Model based on Application Roles



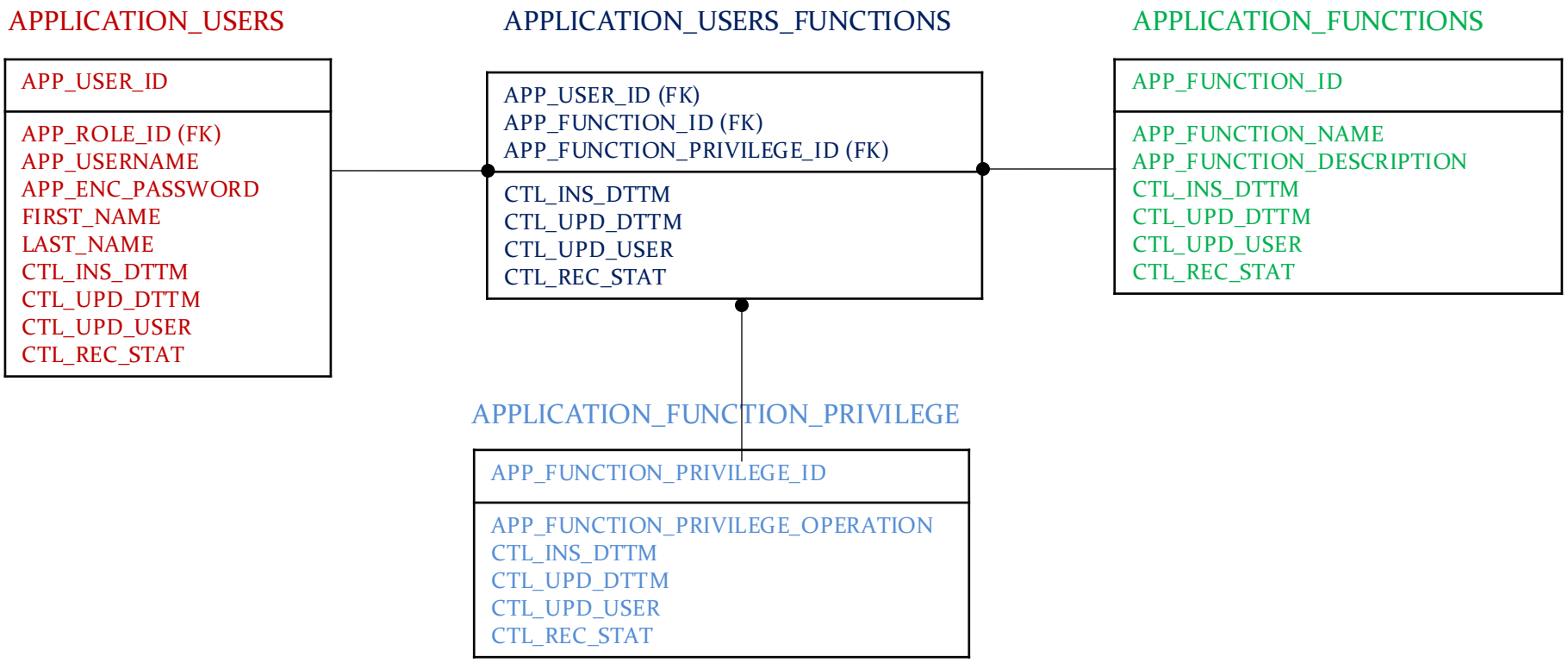
Application Security Models …

Security Model based on Application Roles

- ✓ When considering this security model , keeps this point in mind
  - This model is primitive and does not allow the flexibility required to make changes necessary for security
  - Privileges are limited to any combination like read, add, read / update / admin and so on
- ✓ The following list presents characteristics of this security model
  - Isolating the application security from the database
  - Only one role is assigned to an application user
  - This lowers the risk of database violations
  - Passwords must be securely encrypted
  - The application must use a real database user to log on and connect to the application database

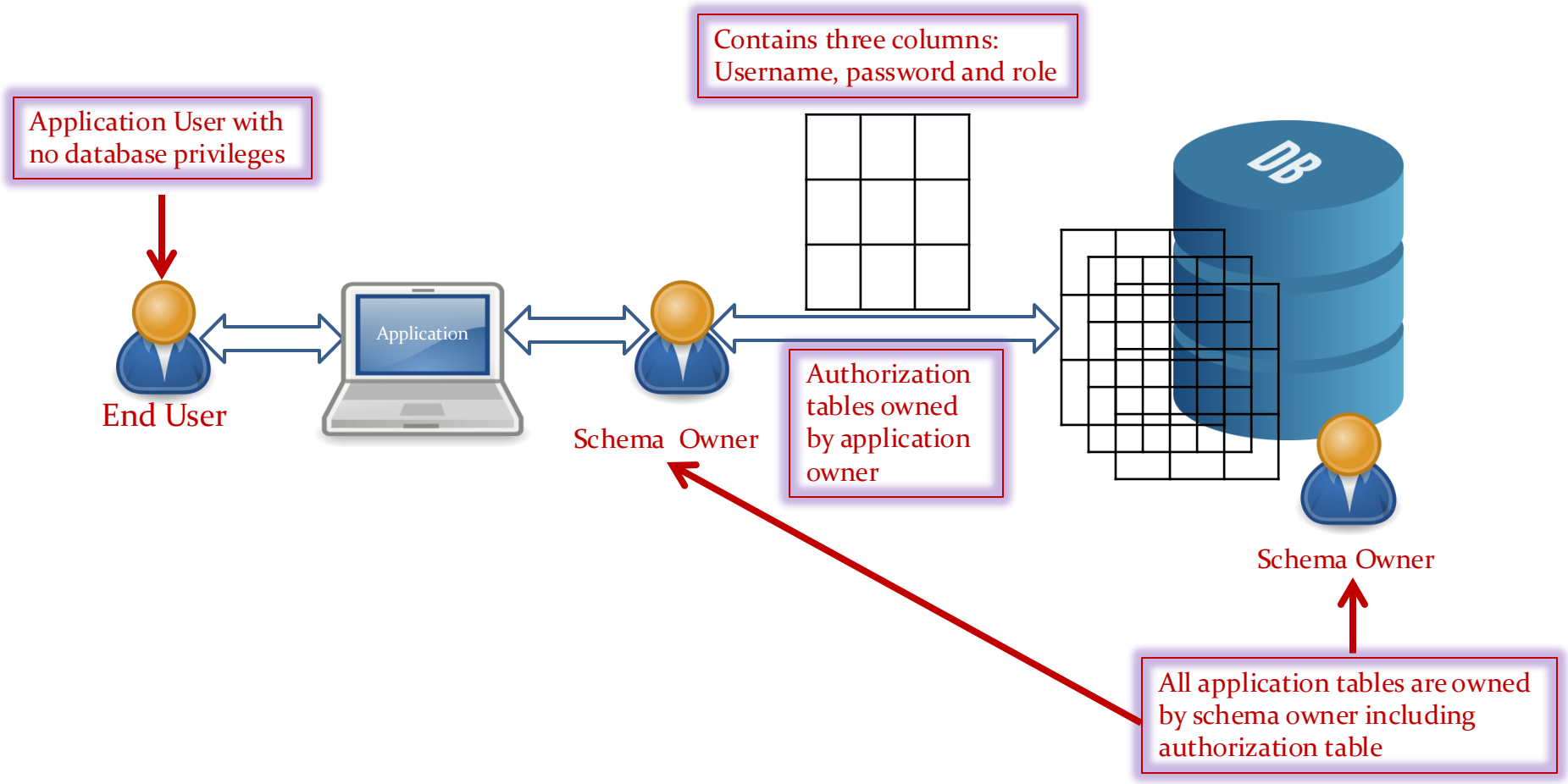
IV - Security Model based on Application Functions

- ✓ Based on application functions depends on the application to authenticate the application users
- ✓ Application divided into functions
- ✓ The following figure represents a data model for this type of application



Application Security Models ...

Architecture of Security Model based on Application Functions



Application Security Models ...

The following list presenting the characteristics of this security model

- Isolating the application security from the database
- Only one role is assigned to an application user
- This lowers the risk of database violations
- Passwords must be securely encrypted
- The application must use a real database user to log on and connect to the application database
- The application must be designed in a granular module. (Granularity: **It is the size of data item allowed to lock**)

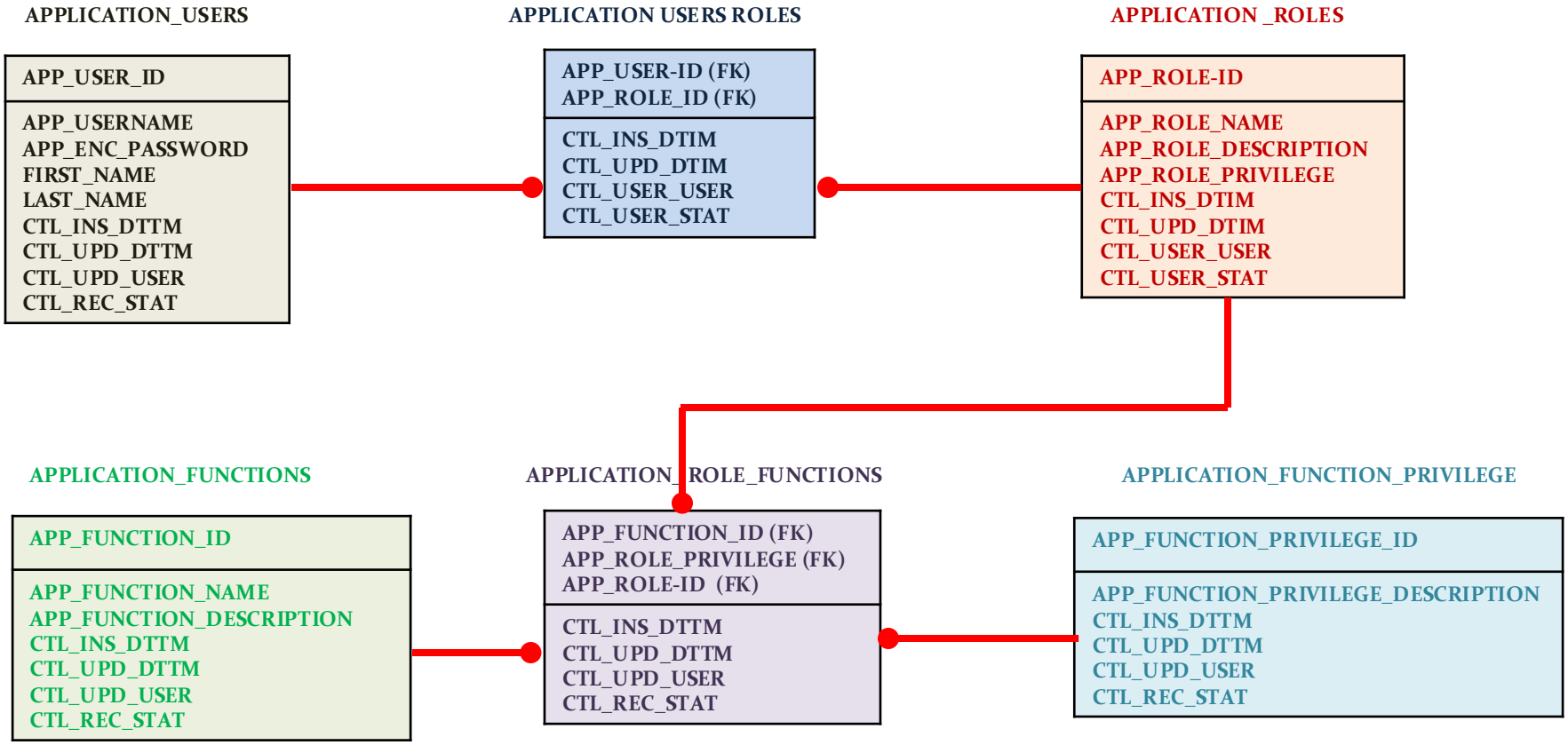
Application Security Models ...

Security model based on Application Roles and Functions

- ✓ It is a combination of both the role and function security model
- ✓ Depends on the application to authenticate the application users
- ✓ The application authenticates users by maintaining all end users in a table with their encrypted passwords
- ✓ Applications are divided into functions and roles are assigned to functions that are in turn assigned to users.
- ✓ This model is highly flexible in implementing application security.

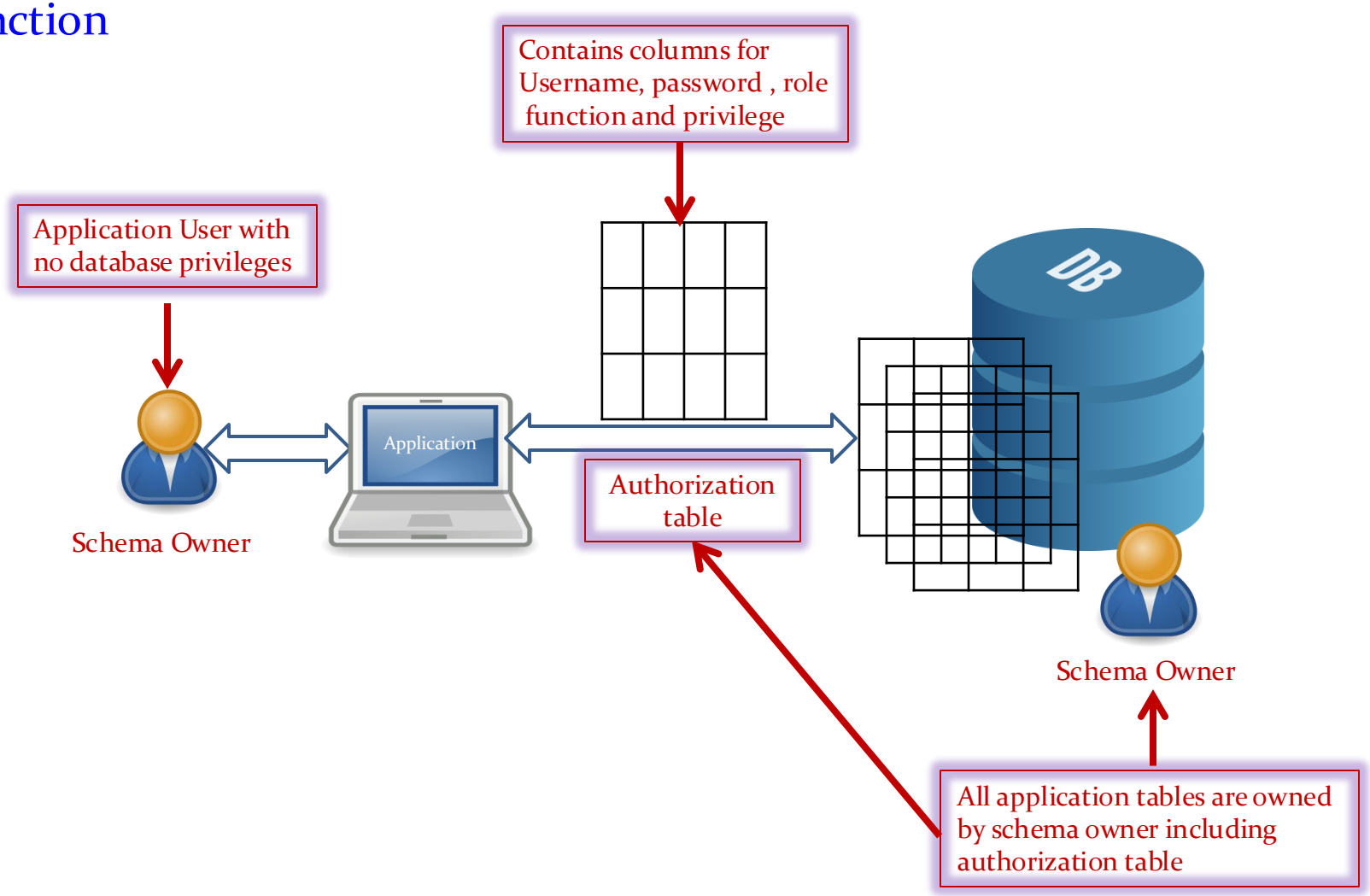
Application Security Models ...

- ✓ The following figure represents a data model for Security Model Based on Application showing the ER Diagram



Application Security Models ...

- ✓ Architecture of a Security data model based on application roles and function



Application Security Models ...

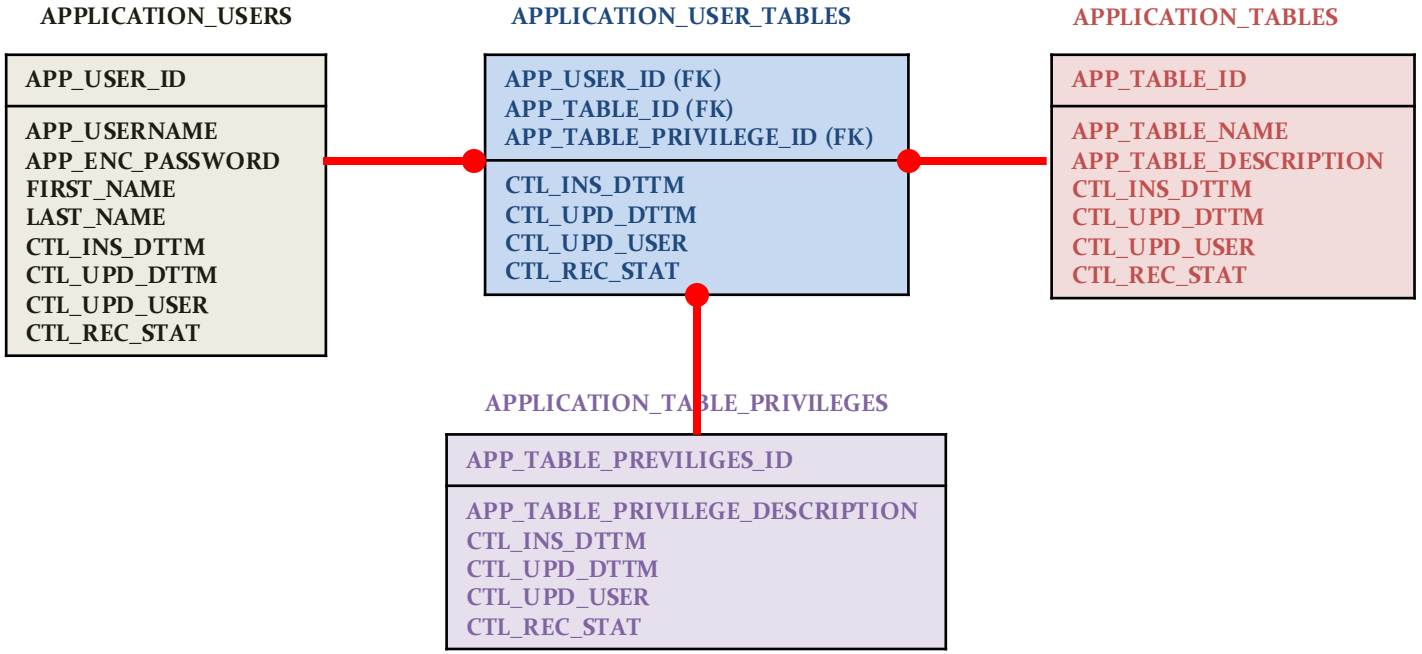
- ✓ The following list presents the characteristics of security model based on application roles and functions

  - Provides utmost flexibility for implementing application security
  - Isolate the application security from the database
  - Maintenance of the application security does not require specific database privileges
  - Lowers the risk of database violations
  - Password must be surely encrypted
  - The application must be designed in a very granular fashion

Application Security Models ...

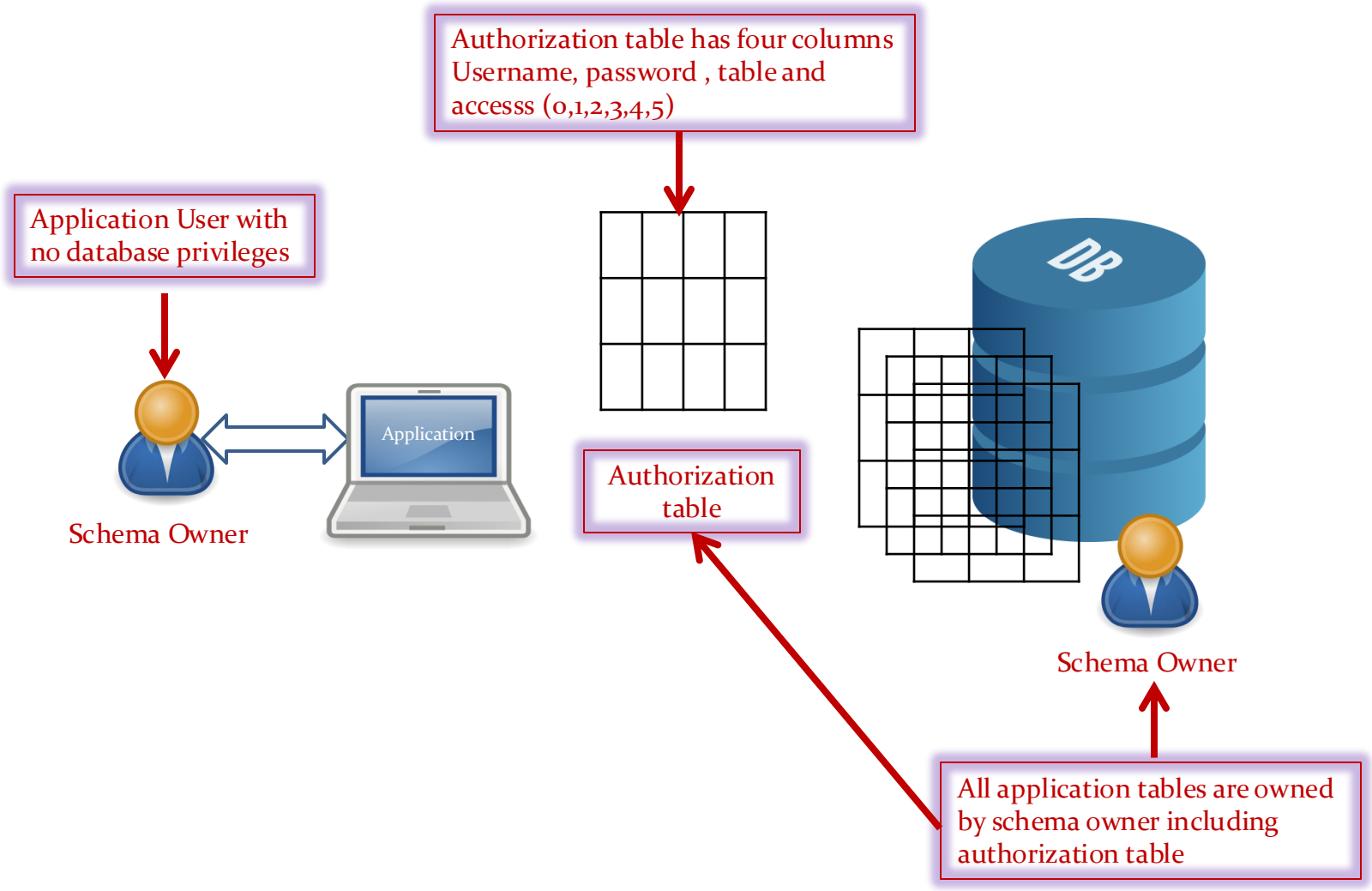
V - Security Model Based on Application Tables

- ✓ Depends on application to authenticate users by maintaining all end users in a table with their encrypted passwords
- ✓ All application provides privileges to the user based on tables
- ✓ User is assigned access privilege to each table owned by the application owner
- ✓ The following figure represents a data model for this security model



Application Security Models ...

Architecture of a Security Model Based on Application Tables



Application Security Models ...

- ✓ The following list presents the characteristics of security model based on application tables

  - Isolate the application security from the database
  - Maintenance of the application security does not require specific database privileges
  - Lowers the risk of database violations
  - Security is implemented easily by using table access privileges

Application Security Models ...

Characteristics of Security Model

Security Model / Characteristics	Database Role based	Application Role based	Application Function Based	Application Role and Function Based	Application Table Based
Is flexible in implementing application security	No	No	No	Yes	No
Isolates application security from the DB	Yes	Yes	Yes	Yes	Yes
Maintenance of application security does not require specific DB privileges	No	No	No	Yes	No
Password must be securely encrypted	Yes	Yes	Yes	Yes	Yes
Uses real DB user to log on	No	Yes	Yes	Yes	Yes
Is business-function specific	No	No	Yes	Yes	No

Data Encryption

- ✓ Encryption is a security method in which information is encoded in such a way that only authorized user can read it.

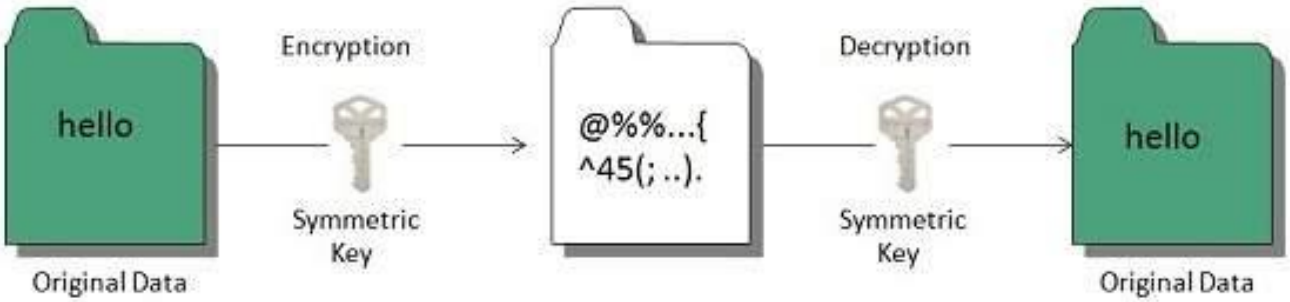
- ✓ It uses encryption algorithm to generate ciphertext that can only be read if decrypted.

Types of Encryption

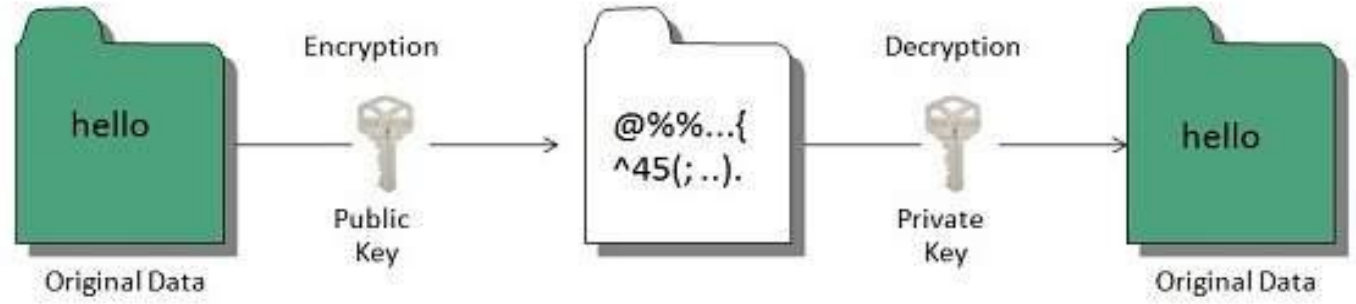
- ✓ There are two types of encryptions schemes as listed below:
  - Symmetric Key encryption
  - Public Key encryption

Data Encryption

- ✓ **Symmetric key encryption** algorithm uses same cryptographic keys for both encryption and decryption of cipher text.



- ✓ **Public key encryption** algorithm uses pair of keys, one of which is a secret key and one of which is public. These two keys are mathematically linked with each other.



Objectives

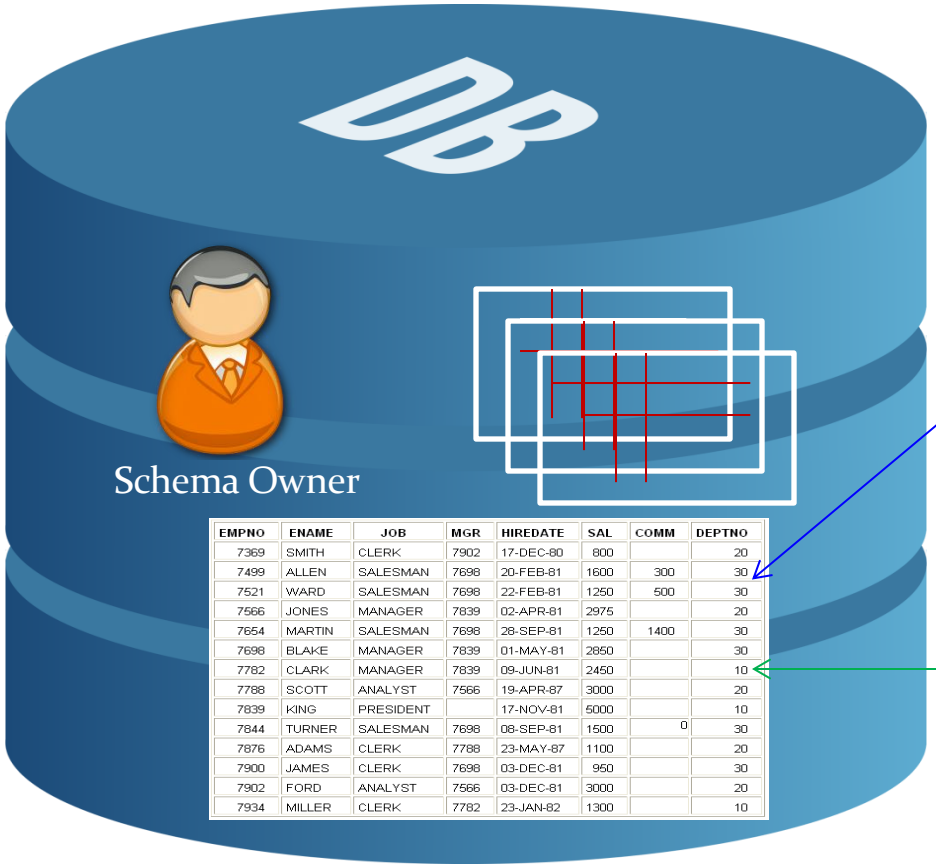
- Define the term “virtual private database” and explain its importance
- Implement a virtual private database by using the VIEW database object
- Implement a virtual private database by using Oracle’s application context
- Implement row-level and column-level security

VPD

- Each department allow to access its own data and visible only to dept.
- One table – each dept. allow sees its own data.

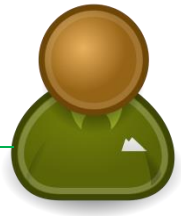
Virtual Private Databases

- ✓ VPD (Virtual Private Database) is shared database schema containing data that belongs to many users , and each user can view or manipulate the data the user owns



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

User can only see and modify data of deptno 20



User can only see and modify data of deptno 10

Virtual Private Databases

- ✓ Not every database system offers a mechanism to implement VPD with out VIEW objects.

- ✓ ORACLE offered VPD in several versions before the release of 10G

ORACLE uses two other names to refer VPDs

- Row Level Security (RLS)
- Fine Grain Access (FGA)

Oracle VPD

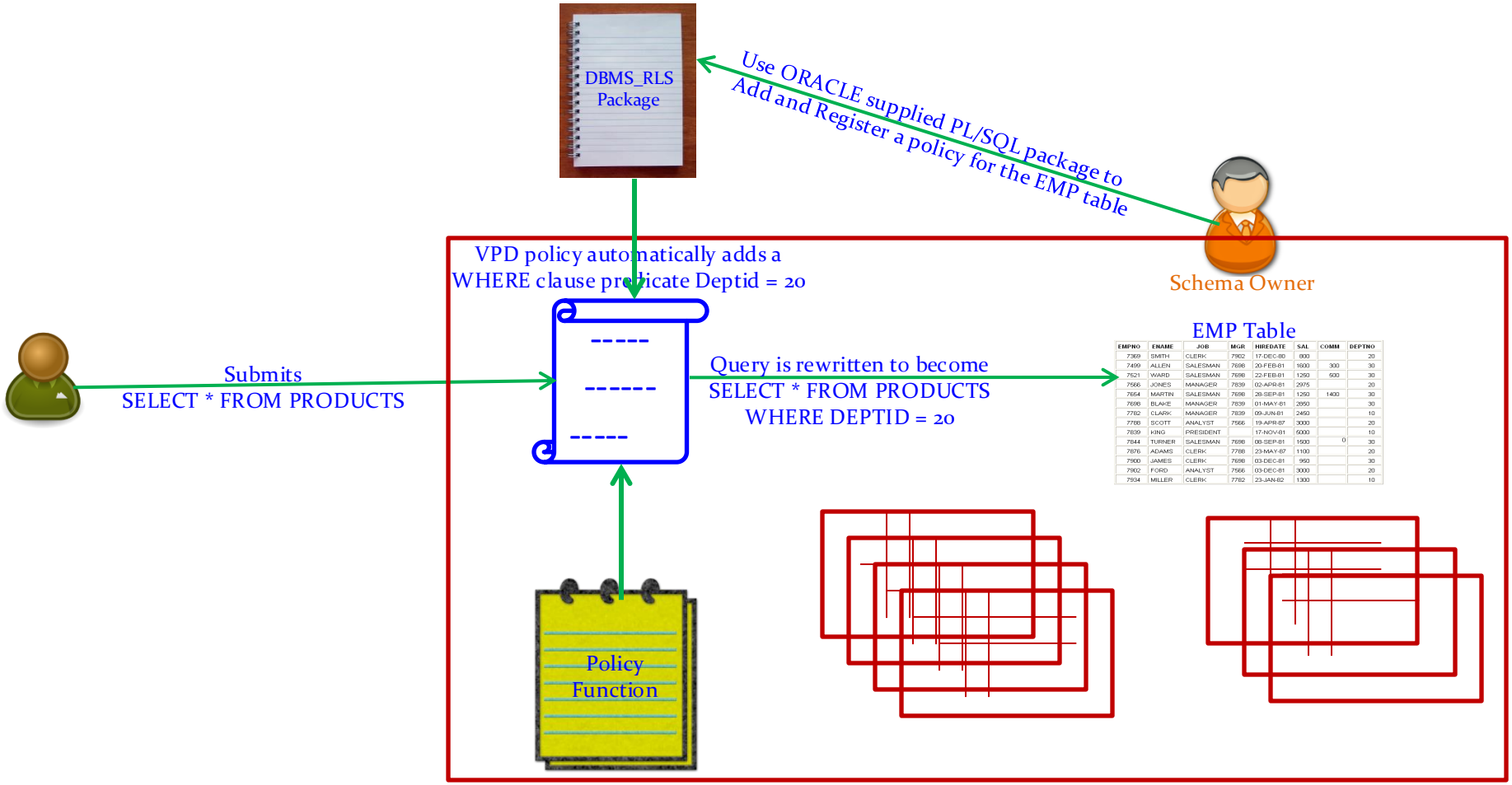
How does it work?

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 by appending the returned predicate to the WHERE clause.
4. The modified SQL query is executed.

Virtual Private Databases

Architecture of Virtual Private Database



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

1. Logon as DBSEC schema and display a column listing of the EMPLOYEES table. If this table does not exist, then create it according to the displayed structure:

```
SQ> DESC EMPLOYEES
Name                                      Null?    Type
----------------------------------------- -------- ------------
EMPLOYEE_ID                               NOT NULL NUMBER(6)
FIRST_NAME                                         VARCHAR2(20)
LAST_NAME                                 NOT NULL VARCHAR2(25)
EMAIL                                     NOT NULL VARCHAR2(25)
PHONE_NUMBER                                       VARCHAR2(20)
HIRE_DATE                                 NOT NULL DATE
JOB_ID                                    NOT NULL VARCHAR2(10)
SALARY                                             NUMBER(8,2)
MANAGER_ID                                         NUMBER(6)
DEPARTMENT_ID                                      NUMBER(4)
CTL_UPD_USER                                       VARCHAR2(30)
```

Implementing a VPD Using Views

2. Create the table EMPLOYEES_VER1 as specified in the previous code:

```
SQL> CREATE TABLE EMPLOYEES_VER1
  2  (
  3      EMPLOYEE_ID      NUMBER(6),
  4      FIRST_NAME       VARCHAR2(20),
  5      LAST_NAME        VARCHAR2(25),
  6      EMAIL            VARCHAR2(25),
  7      PHONE_NUMBER     VARCHAR2(20),
  8      HIRE_DATE        DATE,
  9      JOB_ID           VARCHAR2(10),
 10      SALARY           NUMBER(8,2),
 11      MANAGER_ID       NUMBER(6),
 12      DEPARTMENT_ID    NUMBER(4),
 13      CTL_UPD_USER     VARCHAR2(30)
 14  )
 15  /

Table created.
```

Implementing a VPD Using Views

3. Create a VIEW object to display rows that belong only to the logged on user. Note that USER is a function that returns the user name value of the person who is logged on. If HR is logged on, the returned value is HR.

```
SQL> CREATE VIEW EMPLOYEES_VIEW1 AS
  2     SELECT EMPLOYEE_ID, FIRST_NAME,
  3            LAST_NAME, EMAIL,
  4            PHONE_NUMBER, HIRE_DATE,
  5            JOB_ID, SALARY,
  6            MANAGER_ID, DEPARTMENT_ID,
  7            CTL_UPD_USER USER_NAME
  8       FROM EMPLOYEES_VER1
  9      WHERE CTL_UPD_USER = USER
 10  /

View created.
```

Implementing a VPD Using Views

```
SQL> CREATE VIEW EMPLOYEES_VIEW1 AS
  2     SELECT EMPLOYEE_ID, FIRST_NAME,
  3            LAST_NAME, EMAIL,
  4            PHONE_NUMBER, HIRE_DATE,
  5            JOB_ID, SALARY,
  6            MANAGER_ID, DEPARTMENT_ID,
  7            CTL_UPD_USER USER_NAME
  8       FROM EMPLOYEES_VER1
  9      WHERE CTL_UPD_USER = USER
 10  /

View created.
```

4. Grant SELECT and INSERT on this view to another user, such as SCOTT:

Implementing a VPD Using Views

4. Grant SELECT and INSERT on this view to another user, such as SCOTT:

```
SQL> GRANT SELECT, INSERT ON EMPLOYEES_VIEW1 TO SCOTT
  2 /

Grant succeeded.
```

Implementing a VPD Using Views

5. Insert a row using the EMPLOYEES_VIEW1 VIEW object just created:

```
SQL> INSERT INTO DBSEC. EMPLOYEES_VIEW1 (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
  2                                      PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY,
  3                                      MANAGER_ID, DEPARTMENT_ID, USER_NAME)
  4                              VALUES (100, 'Sam', 'Afyouni', 'safyouni',
  5                                      '123.234.3456', sysdate, 'WM_CLK', 1000,
  6                                      1000, 10, USER);

1 row created.

SQL> COMMIT;

Commit complete.
```

Implementing a VPD Using Views

6. Now log on as SCOTT and insert a row:

```
SQL> INSERT INTO EMPLOYEES_VIEW1 (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
  2                               PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY,
  3                               MANAGER_ID, DEPARTMENT_ID, USER_NAME)
  4                         VALUES (101, 'Julia', 'Ronaldo', 'jronaldo',
  5                                 '456.567.3678', sysdate, 'WM_ENG', 2000,
  6                                 2000, 20, USER);

1 row created.

SQL> COMMIT;

Commit complete.
```

Implementing a VPD Using Views

7. You know that there are two rows in the table EMPLOYEES_VER1.

```
SQL> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, CTL_UPD_USER
  2    FROM EMPLOYEES_VER1
  3  /
EMPLOYEE_ID FIRST_NAME           LAST_NAME                CTL_UPD_USER
----------- ----------           ---------                ------------
        100 Sam                  Afyouni                  DBSEC
        101 Julia                Ronaldo                  SCOTT
```

Implementing a VPD Using Views

8. As SCOTT, select the EMPLOYEES_VIEW1 VIEW object, and you see only the row that belongs to SCOTT.

```
SQL> SELECT DBSEC. EMPLOYEE_ID, FIRST_NAME, LAST_NAME, USER_NAME
  2    FROM EMPLOYEES_VIEW1
  3  /

EMPLOYEE_ID FIRST_NAME          LAST_NAME                 USER_NAME
----------- ----------          ---------                 ---------
        101 Julia               Ronaldo                   SCOTT
```

Virtual Private Databases

- ✓ Setup Test Environment
- ✓ Create an Application Context
- ✓ Create Login Trigger
- ✓ Create Security Policies
- ✓ Apply Security Policies to Tables
- ✓ Test VPD

Virtual Private Databases

Setup Test Environment

- ✓ First we must create a user to act as the schema owner for this example. Obviously, you will perform the following tasks using your current schema owner.

```
CONNECT sys/password@service AS SYSDBA;

CREATE USER schemaowner IDENTIFIED BY schemaowner DEFAULT TABLESPACE
users TEMPORARY TABLESPACE temp;

GRANT connect, resource TO schemaowner;

CREATE USER user1 IDENTIFIED BY user1 DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp;

GRANT connect, resource TO user1;

 CREATE USER user2 IDENTIFIED BY user2 DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp;

GRANT connect, resource TO user2;
GRANT EXECUTE ON DBMS_RLS TO PUBLIC;
```

Virtual Private Databases


```
CONN schemaowner/schemaowner@service

CREATE TABLE users (id NUMBER(10) NOT NULL, ouser VARCHAR2(30) NOT
NULL, first_name VARCHAR2(50) NOT NULL, last_name VARCHAR2(50) NOT
NULL);

CREATE TABLE user_data (column1 VARCHAR2(50) NOT NULL, user_id
NUMBER(10) NOT NULL);

INSERT INTO users VALUES (1,'USER1','User','One');

INSERT INTO users VALUES (2,'USER2','User','Two');

COMMIT;

GRANT SELECT, INSERT ON user_data TO user1, user2;
```

Virtual Private Databases

Create an Application Context

- ✓ Grant CREATE ANY CONTEXT to the schema owner then create the context and context package.

```
CONNECT sys/password@service AS SYSDBA;

GRANT create any context, create public synonym TO schemaowner;

CONNECT schemaowner/schemaowner@service;

CREATE CONTEXT SCHEMAOWNER USING SCHEMAOWNER.context_package;

CREATE OR REPLACE PACKAGE context_package AS PROCEDURE
set_context;
END;
 /
```

Virtual Private Databases

✓ Next we create the context_package body which will actually set the user context.

```
CREATE OR REPLACE PACKAGE BODY context_package IS
  PROCEDURE set_context IS v_ouser VARCHAR2(30); v_id NUMBER;
  BEGIN
  DBMS_SESSION.set_context('SCHEMAOWNER','SETUP','TRUE');
  v_ouser := SYS_CONTEXT('USERENV','SESSION_USER');

            BEGIN
                        SELECT id INTO v_id FROM users WHERE ouser = v_ouser;

            DBMS_SESSION.set_context('SCHEMAOWNER','USER_ID', v_id);
            EXCEPTION WHEN NO_DATA_FOUND THEN
            DBMS_SESSION.set_context('SCHEMAOWNER','USER_ID', 0);
             END;
 DBMS_SESSION.set_context('SCHEMAOWNER','SETUP','FALSE');
 END set_context;
END context_package;
```

Virtual Private Databases

- ✓ Next we make sure that all users have access to the Context_Package.

```
GRANT EXECUTE ON SCHEMAOWNER.context_package TO PUBLIC;

CREATE PUBLIC SYNONYM context_package FOR SCHEMAOWNER.context_package;
```

Create Login Trigger

- ✓ Next we must create a trigger to fire after the user logs onto the database.

```
CONNECT sys/password@service AS SYSDBA;

CREATE OR REPLACE TRIGGER SCHEMAOWNER.set_security_context
AFTER LOGON ON DATABASE
BEGIN
  SCHEMAOWNER.context_package.set_context;
END;
```

Virtual Private Databases

Create Security Policies

- ✓ In order for the context package to have any effect on the users interaction with the database, we need to define a security_package for use with the security policy. This package will tell the database how to treat any interactions with the specified table.

```
CONNECT schemaowner/schemaowner@service;

CREATE OR REPLACE PACKAGE security_package AS
  FUNCTION user_data_insert_security(owner VARCHAR2, objname VARCHAR2)
    RETURN VARCHAR2;

  FUNCTION user_data_select_security(owner VARCHAR2, objname VARCHAR2)
    RETURN VARCHAR2;
END security_package;
```

Virtual Private Databases

- ✓ Next we create the security_package body.

```
CREATE OR REPLACE PACKAGE BODY Security_Package IS
 FUNCTION user_data_select_security(owner VARCHAR2, objname VARCHAR2) RETURN VARCHAR2 IS
  predicate VARCHAR2(2000);
 BEGIN
  predicate := '1=2';
  IF (SYS_CONTEXT('USERENV','SESSION_USER') = 'SCHEMAOWNER') THEN
   predicate := NULL;
  ELSE
   predicate := 'USER_ID = SYS_CONTEXT(''SCHEMAOWNER'',''USER_ID'')';
  END IF;
  RETURN predicate;
 END user_data_select_security;

 FUNCTION user_data_insert_security(owner VARCHAR2, objname VARCHAR2) RETURN VARCHAR2 IS
  predicate VARCHAR2(2000);
 BEGIN
  predicate := '1=2';
  IF (SYS_CONTEXT('USERENV','SESSION_USER') = 'SCHEMAOWNER') THEN
   predicate := NULL;
  ELSE
   predicate := 'USER_ID = SYS_CONTEXT(''SCHEMAOWNER'',''USER_ID'')';
  END IF;
  RETURN Predicate;
 END user_data_insert_security;
END security_package;
```

Virtual Private Databases

- ✓ Next we make sure that all users have access to the Security_Package.

```
GRANT EXECUTE ON SCHEMAOWNER.security_package TO PUBLIC;

CREATE PUBLIC SYNONYM security_package FOR SCHEMAOWNER.security_package;
```

Apply Security Policies to Tables

- ✓ The DBMS_RlS package is used to apply the security policay, implemented by security_package, to the relevant tables.

```
BEGIN
 DBMS_RLS.add_policy('SCHEMAOWNER', 'USER_DATA',
 'USER_DATA_INSERT_POLICY',
              'SCHEMAOWNER', 'SECURITY_PACKAGE.USER_DATA_INSERT_SECURITY',
              'INSERT', TRUE);
 DBMS_RLS.add_policy('SCHEMAOWNER', 'USER_DATA',
 'USER_DATA_SELECT_POLICY',
              'SCHEMAOWNER', 'SECURITY_PACKAGE.USER_DATA_SELECT_SECURITY',
              'SELECT');
END;
```

Virtual Private Databases

Test VPD

- ✓ Finally, test that the VPD is working correctly.

```
CONNECT user1/user1@service;
INSERT INTO schemaowner.user_data (column1, user_id) VALUES ('User 1', 1);
INSERT INTO schemaowner.user_data (column1, user_id) VALUES ('User 2', 2);
COMMIT;
CONNECT user2/user2@service
INSERT INTO schemaowner.user_data (column1, user_id) VALUES ('User 1', 1);
INSERT INTO schemaowner.user_data (column1, user_id) VALUES ('User 2', 2);
COMMIT;
CONNECT schemaowner/schemaowner@service
SELECT * FROM schemaowner.user_data;
CONNECT user1/user1@Service;
SELECT * FROM schemaowner.user_data;
CONNECT user2/user2@Service
SELECT * FROM schemaowner.user_data;
```

Virtual Private Databases

Column level Security with SQL Server

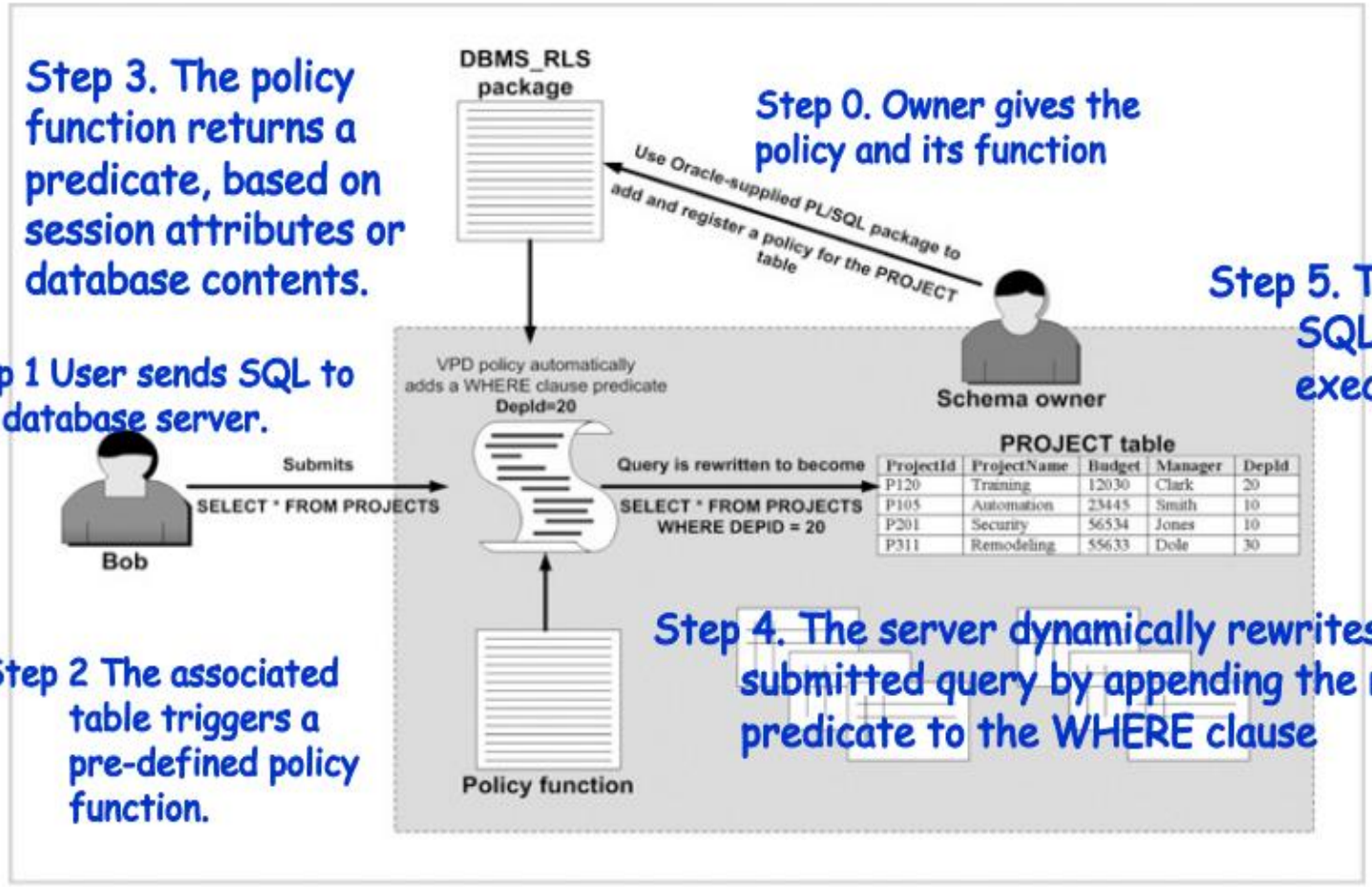
- ✓ Column level permissions provide a more granular level of security for data in your database. You do not need to execute a separate GRANT or DENY statements for each column; just name them all in a query:

```
GRANT SELECT ON data1.table (column1, column2) TO user1;

GO

DENY SELECT ON data1.table (column3) TO user1;

GO
```

- ✓ If you execute a DENY statement at table level to a column for a user, and after that you execute a GRANT statement on the same column, the DENY permission is removed and the user can have access to that column. Similarly, if you execute GRANT and then DENY, the DENY permission will be in force.

References :

1) Hassan A. Afyouni, “Database Security and Auditing”, Third Edition, Cengage Learning, 2009
2) Charu C. Aggarwal, Philip S Yu, “Privacy Preserving Data Mining”: Models and Algorithms, Kluwer Academic Publishers, 2008
3) Ron Ben Natan, ”Implementing Database Security and Auditing”, Elsevier Digital Press, 2005.
4) http://adrem.ua.ac.be/sites/adrem.ua.ac.be/files/securitybook.pdf
5) www.docs.oracle.com

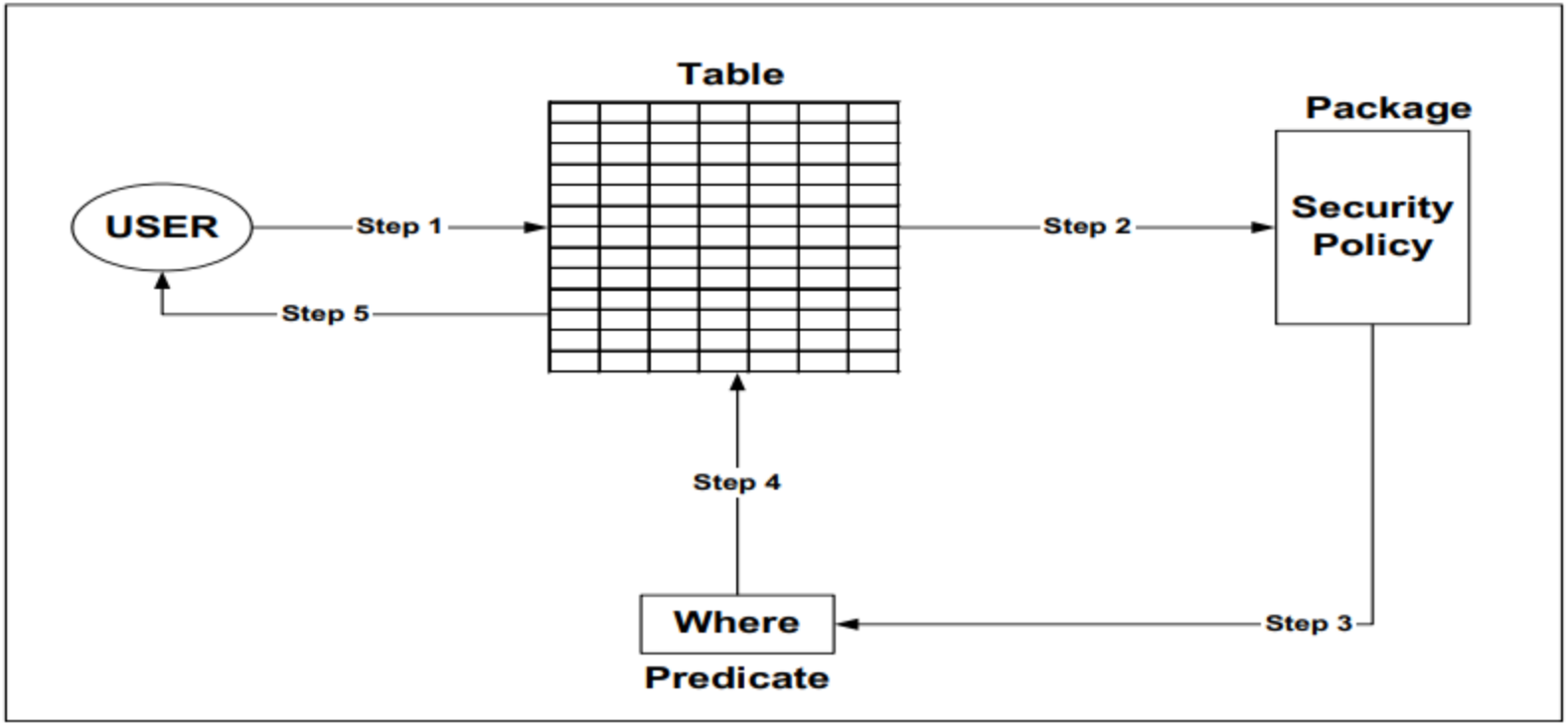


Architecture of Oracle virtual private database feature

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.



VPD enhances security inside the database kernel no matter the source of the statement ... compiled in an application, generated with dynamic SQL, entered into SQL*Plus from a keyboard

For example

Statement Before VPD

```
SELECT employee, salary
FROM payroll
WHERE country_code = 'US';
```

Statement After VPD

```
SELECT employee, salary
FROM payroll
WHERE country_code = 'US'
AND division = 'West Coast'
AND department = 'Finance';
```

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 number(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 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

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 set an attribute value in an application context,

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

- To get 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

Viewing VPD Policies and Application contexts using Data Dictionary

- Oracle is rich with data dictionary view that enable you to everything created and stored in the database.
- VPD's are no exception.
- Oracle provides a set of data dictionary views for all application contexts.
- Oracle also provides data dictionary views for VPD policies.

Viewing VPD Policies and Applications contexts using Data Dictionary

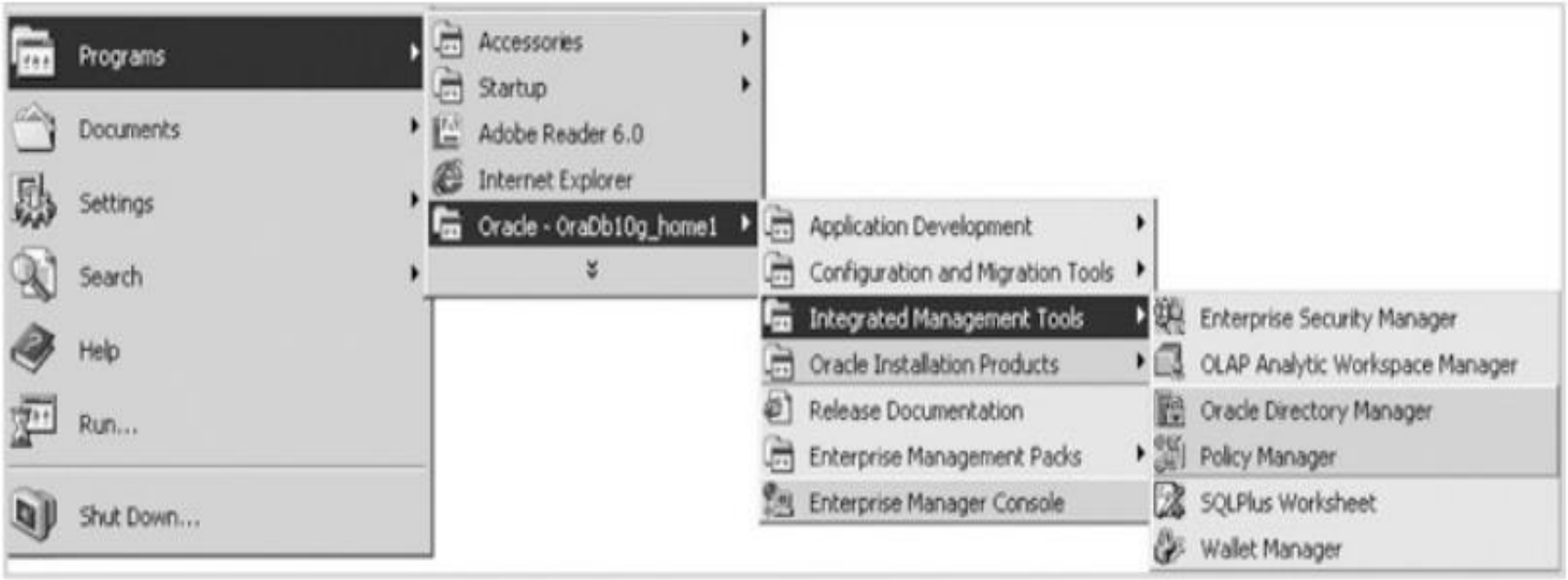
Oracle policy and application context data dictionary views

View Name	Description
DBA_POLICIES	Contains all policies that are created in the database and their attributes
ALL_POLICIES	Contains all policies that the current user owns and has access to and their attributes
USER_POLICIES	Contains all policies owned by the current user and their attributes
V$VPD_POLICY	Lists all policies that have been used and executed and are still cached in memory
ALL_CONTEXT	Lists all contexts that the current user owns or has privileges to view
SESSION_CONTEXT	Lists all active contexts for the current session

Viewing VPD Policies and Applications Context Using Policy Manager

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

Viewing VPD Policies and Applications Context Using Policy Manager

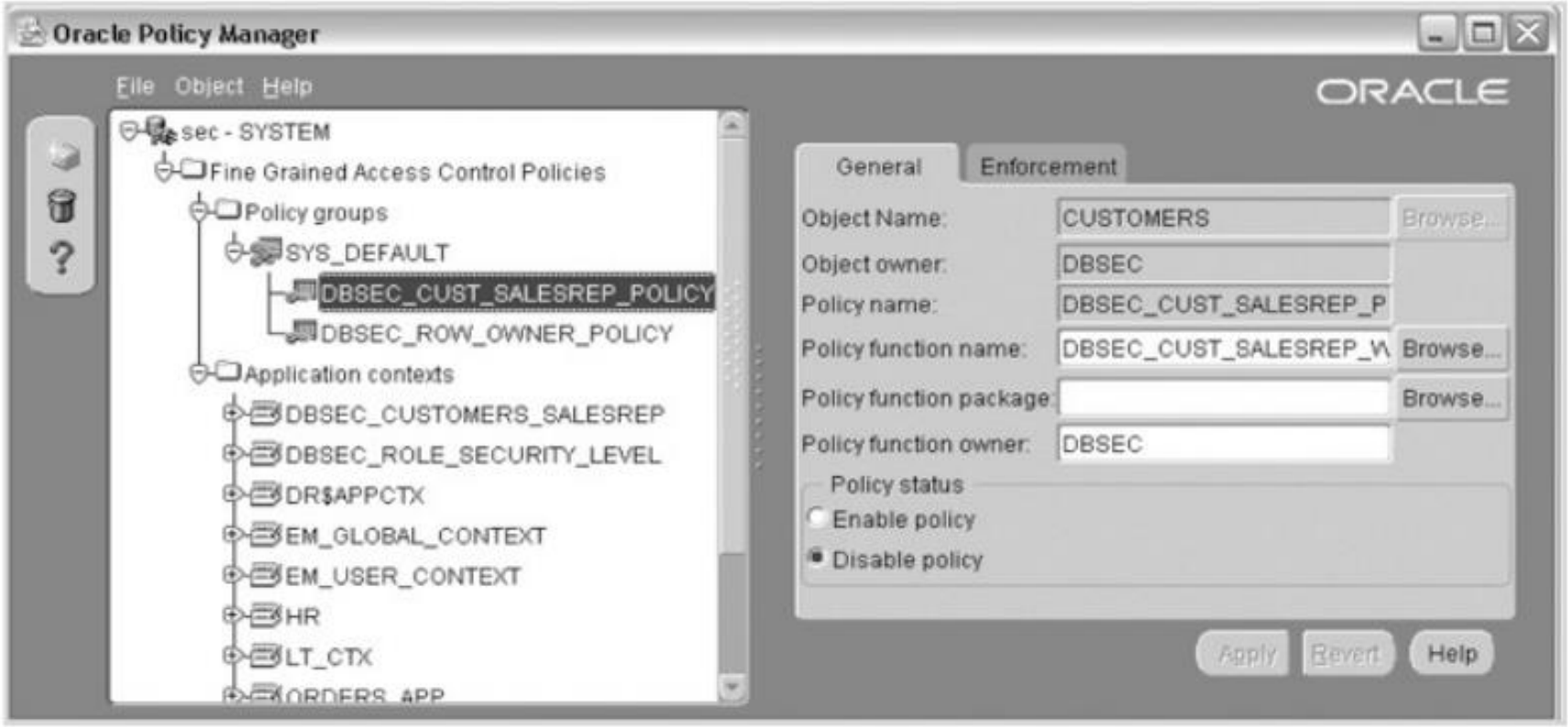


Policy Manager shortcut in the Start menu

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

Logging into Oracle Policy Manager

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



Oracle Policy Manager

Implementing Row and Column level Security with SQL Server

- SQL Server 2000 does not support VPDs; you can mimic their functionality
- Use views and expand security models

Implementing Row and Column level Security with SQL Server

Row-based Security Using Access Levels

- Variation of both:
  - Application table-based security model
  - Application function-based security model
- Access levels:
  - 0 = No access
  - 1 = select
  - 2 = select, insert
  - 3 = select, insert, update

Implementing Row and Column level Security with SQL Server

Row-based Security Using Access Levels (continued)

- Access levels (continued):
  - 4 = select, insert, update, delete
  - 5 = administrator access
- Steps:
  - Create the APPLICATION USERS table
  - Alter the CUSTOMER table to include the ACCESS CONTROL column
  - With the security structure in place use a view to retrieve data

Implementing Row and Column level Security with SQL Server

Row-based Security Using Application Functions

- Steps (continued): apply privileges
- Drawbacks: it allows insertion, update, and deletion of records
- Alternatives:
  - Use stored procedures
  - Use application functions: access table list a function instead of a level

Implementing Row and Column level Security with SQL Server

1. Create the APPLICATION USERS table:

```
create table app_user_access (
    username varchar(128) not null primary key,
    access_level  tinyint not null default 0
)
go

insert into app_user_access values ('sam', 4)
insert into app_user_access values ('jason', 0)
go
```

2. Alter the CUSTOMER table to include the ACCESS CONTROL column:

```
alter table customer
    add access_level integer
go
```

Implementing Row and Column level Security with SQL Server

3. With the security structure in place, you need a way to retrieve the data. One way to retrieve data is by using a view:

```
create view vCustomer_Secure
as
      select customer_id, first_name, last_name, street, city, state, zip,
 phone,
       gender, date_of_birth, comments
      from customer
      where access_level > 0
and access_level <= (select isnull(access_level, 0)
                           from app_user_access where username = user)
go
```

4. Apply privileges:

```
grant select on vCustomer_Secure to sam
grant select on vCustomer_Secure to jason
go
```

Implementing Row and Column level Security with SQL Server

1. Create a procedure:

```
create procedure Customer_Sel
as
      select customer_id, first_name, last_name, street, city, state, zip,
 phone,
       gender, date_of_birth, comments
      from customer
      where access_level > 0
and access_level <= (select isnull(access_level, 0)
from app_user_access where username = user)
go
```

2. Apply privileges:

```
grant execute on Customer_Sel to sam
grant execute on Customer_Sel to jason
go
```

Implementing Row and Column level Security with SQL Server

Column-based Security

- VPD and Column Access Using Oracle steps:
  - Log in as VPD_CLERK2 and view rows and columns in the CUSTOMERS table
  - Log in as the DBSEC user and recreate the policy on customers
  - Log in as VPD_CLERK2 and query the CUSTOMERS table

Implementing Row and Column level Security with SQL Server

Column-based Security

- Column privileges in Oracle steps:
  - Log in as DBSEC and create a table called TEST
  - Grant SELECT on the TEST table to SCOTT
  - Grant UPDATE only on the column TEXT in the TEST table to SCOTT
  - Insert a row into the TEST table and save it
  - Log in as SCOTT and query the TEST table owned by DBSEC

Implementing Row and Column level Security with SQL Server

Column-based Security (continued)

- Column privileges in Oracle steps (continued):
  - Update the TEXT column in the TEST table
  - Try to update the NUM column in the TEST table

Implementing Row and Column level Security with SQL Server

Column-based Security (continued)

- Access-level control with SQL Server steps:
  - Create the APP_TABLES table
  - Create the APP_COLUMNS columns
  - All access to the tables must be performed with stored procedures

Implementing Row and Column level Security with SQL Server

- 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)

Assignment

Subject & Title : 18CSE455T - DATABASE SECURITY AND PRIVACY

Total Marks : 10

(Assignment Submission : 5 Marks, Viva : 5 Marks)

Explain the following:

1. **Privacy-Preserving Data Mining Algorithms**
2. **General Survey-Randomization Methods**
3. **Applications of Privacy-Preserving Data Mining**

Assignment submission last date: 12.06.2024

Note : Minimum 4 pages for each question.