The Sea Views project described in is the basis for a system that integrates a trusted operating system to form a trusted database manager. The layered implementation as described is shown in Figure 1.

1.Layer 1: Access Control :The lowest layer, the reference monitor, performs file interaction, enforcing the BellLa Padula access controls, and does user authentication. Part of its function is to filter data passed to higher levels.
2 Layer2 : Secure Index: The second level performs basic indexing and computation functions of the database.
3.Layer3 :View Translation: The third level translates views into the base relations of the database.
These three layers make up the trusted computing base (TCB) of the system.
4.The remaining layers implement normal DBMS functions 5 and the user interface

Privileges and Authorizations

- Allow or deny access
- Principle of Least Privilege
- Different Models:
  - ACM: Access control matrix
  - ACL: Access Control List
  - C-list: Capability List
  - DAC: Discretionary Access Control
  - MAC: Mandatory Access Control
  - RBAC: Role Based Access Control
  - ABAC: Attribute Based Access Control
  - FGAC: Fine Grained Access Control

Access Control by SQL Model

- **Data Manipulation Language (DML):**
  - Insert, Update, Delete , Merge
  - Permission to subject/user by using DCL TO DO DML
- **Data Definition Language (DDL):**
  - Create, Alter, Drop, Rename, Truncate.
  - (Permission to ACCESS(DDL) - OBJECT by using DCL)
- **Data Control Language (DCL):**
  - Grant, Revoke.
- **Transaction Control:**
  - Commit, Rollback, Savepoint.

Managing Users

How can you use SQL statements to create users and roles, assign privileges and control passwords?

Profile

1.Creating profile [passwords setting]

```
CREATE PROFILE RAMESH LIMIT
FAILED_LOGIN_attempts 3
PASSWORD_LOCK_TIME UNLIMITED
PASSWORD RESUME_TIME 30
PASSWORD_LIFE_TIME 30
PASSWORD_VERIFY_FUNCTION verify_function
PASSWORD_GRACE_TIME 5;
```

Dropping a profile

```
DROP PROFILE developer_prof CASCADE ;
```

2. CREATING a Profile : Resource limit

```
CRATE PROFILE developer_proof LIMIT
SESSION_PER_USER 2
CPU_PER_SESSION 10000
IDLE_TIME 60
CONNECT_TIME 480;
```

Operating System Authentication

- Log on to OS by that user account
- Run sqlplus, automatically connected to that user in oracle

```
CREATE USER user_name
  IDENTIFIED  {  BY  password  |  EXTERNALLY  |  GLOBALLY  AS
'external_name'}
  [ DEFAULT TABLESPACE tablespace
  [ TEMPORARY TABLESPACE {tablespace | tablespace_group_name} ]
    [ QUOTA integer [ K | M | G | T | P | E ]
        | UNLIMITED }
            ON tablespace     ]
  [ PROFILE profile_name ]
  [ PASSWORD EXPIRE ]
  [ ACCOUNT { LOCK | UNLOCK } ]
```

GRANT{PrivilegeList|ALLPRIVILEGES}

ON ObjectName

TO {AuthorizationIdList|PUBLIC} [WITH GRANT OPTION] [WITH GRANT OPTION]

- PrivilegeList consists of one or more of above privileges separated by commas.
- ALL PRIVILEGES grants all privileges to a user

Access Control - Authorization Identifiers and Ownership

Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password.

- Used to determine which objects user may reference and what operations may be performed on those objects. performed on those objects.
- Each object created in SQL has an owner, as defined in AUTHORIZATION clause of schema to which object belongs.
- Owner is only person who may know about it.

•creating a role

You use the following command to create a role

CREATE ROLE role[NOT IDENTIFIED | IDENTIFIED {by password | EXTERNALLY}]

Modifying roles

ALTER ROLE role {NOT IDENTIFIED | IDENTIFIED {by password | EXTERNALLY }];

Enabling and disabling roles

SET ROLE {role [IDENTIFIED BY PASSWORD] [, role [IDENTIFIED BY PASSWORD]]...|ALL [EXCEPT role [, role].]| NONE}

SQL Security Model-DAC

Discretionary Access Control (DAC)

- Discretion: freedom to decide what to do
- Subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.
- Types:
  - Central administration: Predefined subject can grant permissions to others
  - Ownership administration: Creator is the owner
- Cascading and non cascading revoke variations .
- GRANT and REVOKE in Oracle

2 Types of DAC privileges in Oracle

- GRANT command is used to assign a particular privilege to a user.
- REVOKE command is used to revoke an assigned privilege from any user.
- There are 2 types of privileges:
  - System privilege: Granted by database admin or other users who have system privileges.
  - Object privilege: granted by owner of object.

Object Privileges

- ALTER—e.g., alter table.
- CONNECT—connect to a database (i.e., create a session).
- DELETE—for tables, views, and materialized views (when updateable)
- EXECUTE—for procedures, functions, packages, etc.
- INSERT—for tables, views, and materialized views.
- SELECT—for tables, views, materialized views, and sequences.
- UPDATE—for tables, views, and materialized views.

Mandatory Access Control (MAC)

- Multilevel relational database
- Usually called Non-DAC
- Access control policies are fixed → mandatory as per the order as below:
- Objects: Classification level, L(O)
- *TOP SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED*
- Subject: Clearance level, L(S)

TOP SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED

Comparison way :L(O) *CONFIDENTIAL* < L(S) *SECRET*

- ☐ Denial of Service 23603
- ☐ Execute Code 32718
- ☐ Overflow 18081
- ☐ XSS 15303
- ☐ Directory Traversal 4130
- ☐ Bypass Something 6375
- ☐ Gain Information 10989
- ☐ Gain Privilege 5006
- ☐ Sql Injection 7853
- ☐ File Inclusion 2235
- ☐ Memory Corruption 5339
- ☐ CSRF 2521
- ☐ Http Response Splitting 166

3. SENSITIVE DATA EXPOSURE

-It occurs when an application or any other entity exposes personal data.

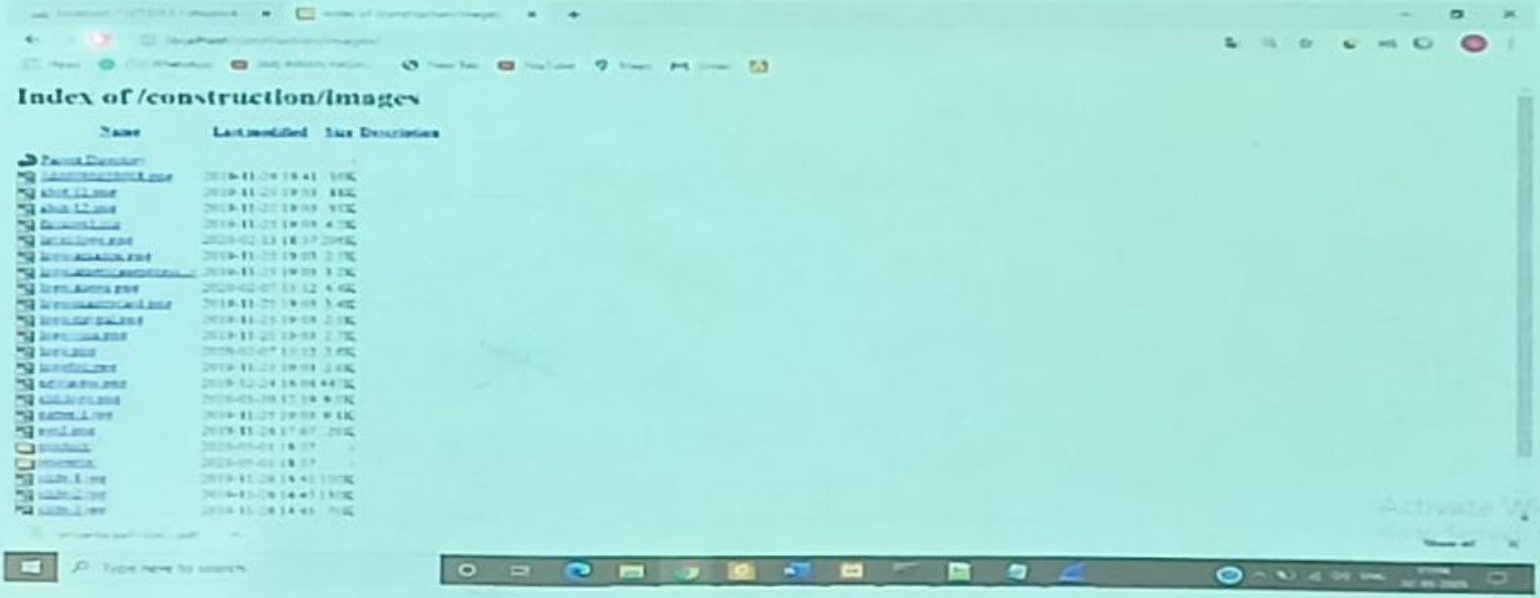
-When databases are not protected adequately sensitive data exposure can occur.

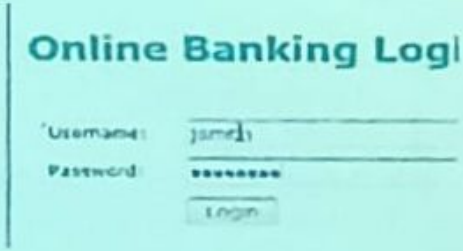
-Different types of data can be exposed such as bank account details, session IDs, Card details, user account information.

-Sensitive data exposure differs from a data breach, in which attacker accesses and steals information.

Sensitive Data Exposure Contd...

- IF the web application is vulnerable to this attack try following
- Enter into URL localhost/construction/images/
- Then the all the files in between the folder is display so that the attacker know the coding of all files.





- Weak crypto algorithms are **susceptible to attacks** and give out sensitive data.
- In the below example the username and password are sent using **base64 encoding.**

The request can be easily intercepted and decoded. The attacker can also launch SQL attacks by gaining such knowledge Check the password in the below intercepted and decoded request.

You can use BurpSuite for interception and decoding.

SENSITIVE DATA EXPOSURE....Contd..

Example – An application encrypts card numbers in a database by automatic database encryption and we decrypt this data automatically when retrieved.

Disallow a SQL injection flaw to retrieve data.

With HTTPS, traffic is encrypted such that even if the packets are sniffed or otherwise intercepted, they will come across as nonsensical characters

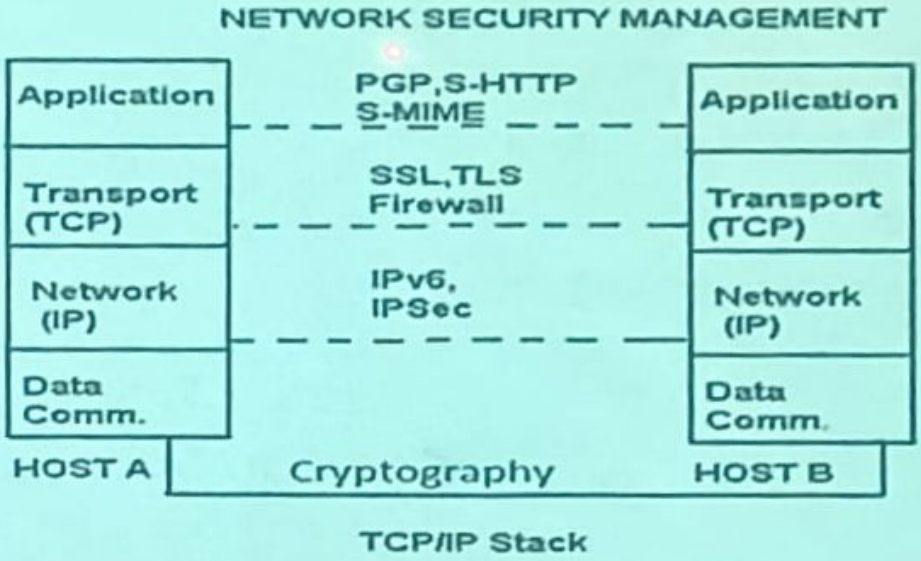
- **Before encryption:**
- *This is a string of text that is completely readable*
- **After encryption STRING OF TEXT:**
- ITM0IRyiEhVpa6VnKyExMiEgNveroyWBPlgGyfkflYjDa aFf/Kn3bo3OfghBPDWo6AfSHlNtL8N7ITEwlXc1g

- There were two main programs that were solely responsible for the complete email security that the people needed. One was **S/MIME** which we will see later and the other was **PGP**.
- **PGP (Pretty Good Privacy)**, is a popular program that is used to provide **confidentiality and authentication services** for electronic mail and file storage

Sensitive Data Exposure contd..

- "Attackers can sniff or modify the sensitive data if not handled securely by the application. A few examples include use if weak encryption keys, use of weak TLS."
- The goal is to identify sensitive data bits and exploit them.
- **Example**
- Weak crypto algorithms are susceptible to attacks and give out sensitive data.
- In the below example the username and password are sent using base64 encoding.

3)Use Secure Protocols and Algorithms.



VULNERABILITY IN SECURITY PROTOCOLS ?

Encrypt all data in **transit from HOST A to HOST B**

Mandatory Access Control (MAC)

- Mandatory Access Control (MAC) is the strictest of all levels of control.
- The design of MAC was defined, and is primarily used by the government.
- MAC takes a hierarchical approach to controlling access to resources. Under a MAC enforced environment access to all resource objects (such as data files) is controlled by settings defined by the system administrator.
- As such, all access to resource objects is strictly controlled by the operating system based on system administrator configured settings.
- It is not possible under MAC enforcement for users to change the access control of a resource.
- Mandatory Access Control begins with *security labels* assigned to all resource objects on the system. These security labels contain two pieces of information
- - A CLASSIFICATION (top secret, confidential etc) and
- A CATEGORY (which is essentially an indication of the management level, department or project to which the object is available).

ACCESS CONTROL BY -DAC,MAC,RBAC.

A system may employ DAC, MAC or their combination of both or RBAC fro protection.

A **DAC** specifies the access types of users on data, and the rules whereby users can, at their discretion, grant and revoke their authorizations of other users.

A **MAC(used by government-strict policy)** policy specifies the rules whereby users can obtain direct or indirect access to classified data, and the rules for sanitizing and reclassifying data. These policies apply only to multilevel databases, which are databases that contain information of different classification.

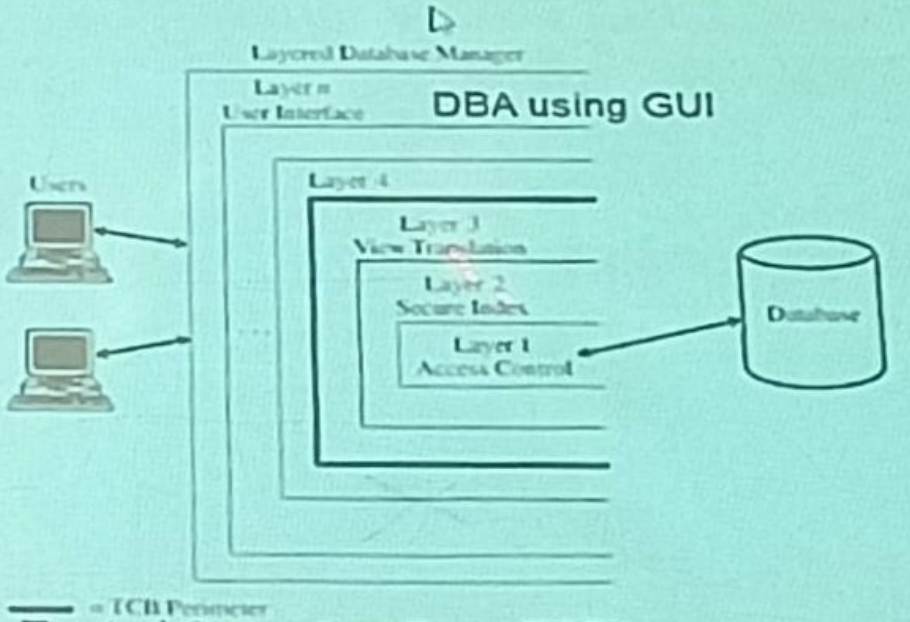
Mandatory Access Control (MAC) where **access** to system resources is **controlled** by the operating system (under the **control** of a system administrator),

Discretionary Access Control (DAC) allows each user to **control** access to their own data. ... User A can, however, set **access** permissions on a file that she owns to B or C....

A **RBAC** provides mechanisms for the enforcement of the least privilege and separation of duties

Secure Database Decomposition.

There's no silver bullet solution with cyber security, a layered defense is the only viable defense."
James Scott, Institute for Critical Infrastructure Technology



Make comm.systems, DBMS, OS secure

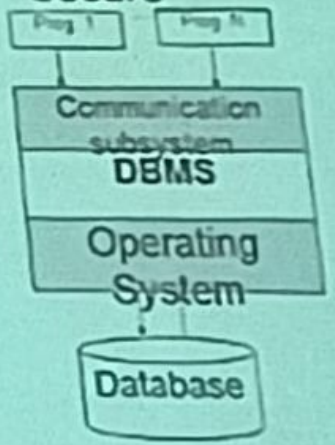


Figure 1 Generic view of a DBMS

Bell LaPadula (BLP) Model-BLP-C

- Two rules: **no read up, no write down**
  - **Simple Security Condition**: Subject *S* can read object *O* if and only if $L(O) \leq L(S)$. → **no read up**

For example : L(O) *CONFIDENTIAL* < L(S) *SECRET:* → **no read up**

- ***-Property (Star Property)**: Subject *S* can write object *O* if and only if $L(S) \leq L(O)$. → **no write down**

Upward flow of data permitted

Provides CONFIDENTIALITY

ACCESS CONTROL

- SEA VIEWS -Database Security(by commands access of tables and attributes)
- Authentication And Authorization
- Access Control by SQL Model
- Privileges and Authorizations-Different Models
- **ACCESS CONTROL BY** -DAC,MAC,RBAC or their combination.
- **PRACTICALS**

- Syntax for granting/revoking system privileges:

```
GRANT {system_privilege | role | ALL PRIVILEGES }
      [, { system_privilege | role | ALL PRIVILEGES }]…
TO {user | role | PUBLIC } [, { user | role | PUBLIC
}]…
[IDENTIFIED BY password] [WITH ADMIN OPTION]
```

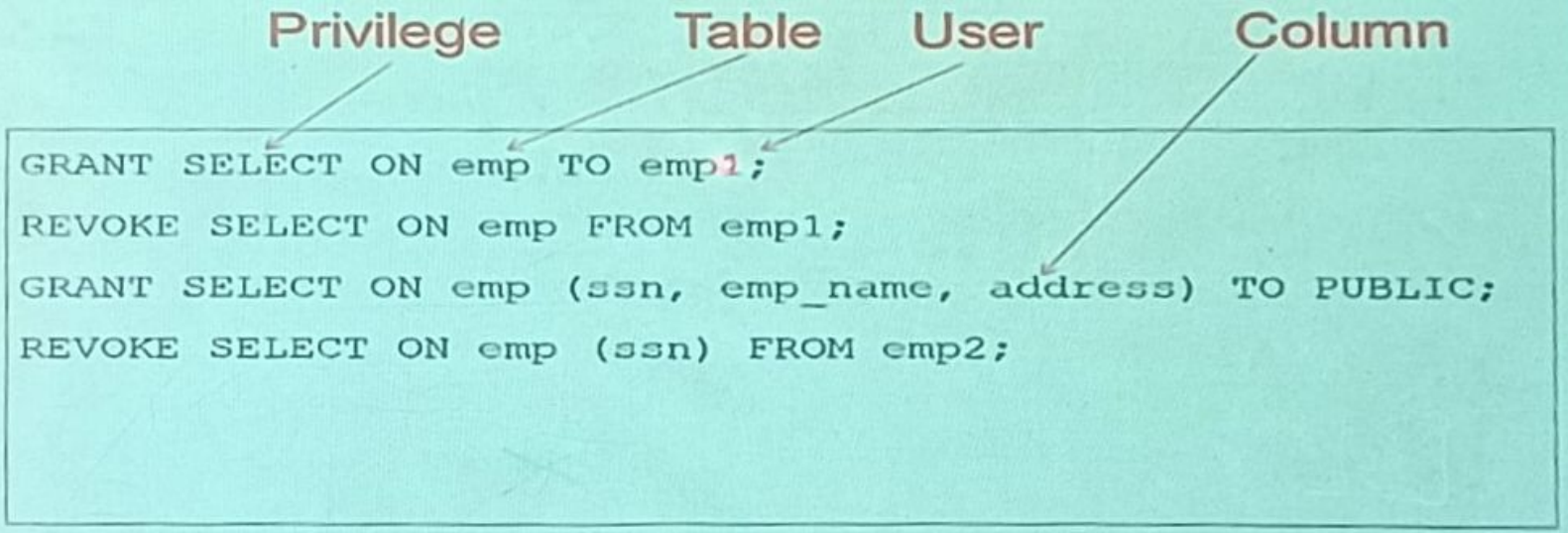
```
REVOKE {{system_privilege | role | ALL PRIVILEGES }
      [, { system_privilege | role | ALL PRIVILEGES }]…
FROM {user | role | PUBLIC } [, { user | role | PUBLIC
}]…
};
```

```
GRANT SELECT ANY TABLE TO emp1;
REVOKE SELECT ANY TABLE FROM emp1;
```

System Privileges

- System privileges:
  - CREATE/DROP TABLE
  - CREATE/DROP ANY TABLE
  - CREATE/DROP ANY VIEW
  - CREATE/DROP PROCEDURE
  - EXECUTE ANY PROCEDURE
  - etc.

Object Privileges (contd.)



DAC is based on granting and revoking privileges.
Informally, there are two levels for assigning privileges to use the database system: 1. The account level 2.Relation level.
(Fine grained Access Control-Account Level and Relation Level)

- The owner of the relation is given all privileges on the relation.
- The owner account holder can pass priviledges of the owned relations to other users by granting privileges to their accounts.

We can propagate privileges using the grant option such as

GRANT SELECT ON EMPLOYEE, DEPARTMENT TO sontakke WITH GRANT OPTION

We can also REVOKE privileges by REVOKE command such as
REVOKE SELECT ON EMPLOYEE FROM sontakke

Revoking system privileges and roles:

You can use the REVOKE command to remove a role or privilege from the user.

- The following command revoke create table from the user named josheph.:

```
REVOKE CREATE TABLE FROM Josheph,
```

- If you have granted a privilege to PUBLEC you can revoke it by using PUBLIC in place of the username.

```
REVOKLE UNLIMITED TABLESPACE FROM PUBLIC. ;
```

As with a GRANT statement, you may revoke any number of roles and privileges using one command.

Discretionary Access Control (DAC)

- SQL security model implements DAC based on
  - *users*: users of database - *user identity* checked during login process;
  - *actions*: including **SELECT**, **UPDATE**, **DELETE** and **INSERT**;
  - *objects*: tables (*base relations*), views, and columns (*attributes*) of tables and views
- Users can protect objects they own
  - when object created, a user is designated as 'owner' of object
  - owner may grant access to others
  - users other than owner have to be granted *privileges* to access object

•You can list as many as privileges and roles in the command as you need.

For Example.

GRANT CREATE SESSION, CREATE TABLE, CONNECT,

UNLIMITED TABLESPACE TO Joseph;.

•By default, only DBA can grant roles or privileges to other users.
For Example:

GRANT CREATE TABLE to Joseph WITH ADMIN OPTION;

For Example:

•To give all privileges to a user say JEMAN

SQL> create user JEMAN IDENTIFIED by BRILLIANT

SQL> grant connect, resource to JEMAN

Due to this command all privileges are granted to JEMAN

by DBA.

- Use the following command for Granting object privileges
- GRANT {object_priv [{column_list}][, object_prive[(column_list)]. | ALL [PRIVILEGES}]
- ON [schema.]object
- TO {user |role|PUBLIC} [, {user|role|PUBLIC}].
- [WITH GRANT OPTION]
- Revoking object privileges
- REVOKE {object_priv [, object_priv].|[PRIVILEGES]} ON [schema.]object FROM {user|role|PUBLIC} [, {user|role|PUBLIC}].[CASCADE CONSTRAINTS]

Operating System Authentication (contd.)

```
C:\Windows\system32>SQLPLUS SYSTEM

SQL*Plus: Release 10.2.0.1.0 - Production on Sun Dec 29 12:05:11 2013

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Enter password:

Connected to:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

SQL> SHOW PARAMETER OS_AUTHENT_PREFIX

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
os_authent_prefix                    string
SQL>
SQL> CREATE USER "COMPAQ\USER2" IDENTIFIED EXTERNALLY;

User created.

SQL> GRANT CREATE SESSION TO "COMPAQ\USER2";

Grant succeeded.

SQL> SHOW USER
USER is "SYSTEM"
SQL>
```

Privileges

- Actions user permitted to carry out on given base table or view:

SELECT Retrieve data from a table. INSERT Insert new rows into a table.

UPDATE Modify rows of data in a table. UPDATE Modify rows of data in a table.

DELETE Delete rows of data from a table.

REFERENCES Reference columns of named table in integrity constraints.

USAGE Use domains, collations, charactersets,and translations

Creating a new user : operating system Authentication :

Use the INDENTIFIED EXTERNALLY clause of the CREATE USER command to specify that a user must be authenticated by the operating system.

CONNECT <user> USING<password>

In init.ora,

OS_AUTHENT_PREFIX = " "

Using OS-AUTHENT-PREFIX=OPS$ gives the flexibility of having a user authenticated by the operating system of the oracle server.

In this case, the DBA can create the user by entering the command of the form:

CREATE USER OPS$ user

IDENTIFIED BY password .....

DBA can set another initialization parameter,

REMOTE_OS_AUTHENT=True to authenticate the user by a remote operating system.

OPERATING SYSTEM AUTHENTICATION

Authentication & identification mechanisms
- CONNECT <user> USING<password>
- DBMS may chose OS authentication or its own authentication mechanism
  - Kerberose
  - PAM

Creating users

```
CREATE USER SCOTT
IDENTIFIED By TIGER
DEFAULT TABALESPACE   app_ data
TEMPORARY TABLESPACE temp
QUOTA 15M ON     app_  data
PASSWORD EXPIRES;
```

Dropping user

```
DROP USER Username;
DROP USER Username CASCADE;
```

The CASCADE option drops all objects in the schema before dropping the user.

Password Authentication

```
CREATE USER user_name
  IDENTIFIED { BY password | EXTERNALLY | GLOBALLY AS
'external_name'}
  [ DEFAULT TABLESPACE tablespace
  [      TEMPORARY      TABLESPACE      {tablespace      |
tablespace_group_name} ]
    [ QUOTA integer [ K | M | G | T | P | E ]
        | UNLIMITED }
            ON tablespace     ]
  [ PROFILE profile_name ]
  [ PASSWORD EXPIRE ]
  [ ACCOUNT { LOCK | UNLOCK } ]
```