IT 4153 Advanced Database

***Module 8 Database Security***

|  |
| --- |
| **Introduction and Module Summary** |
| In this module you will learn   * Operating System Security * DBMS Security * Database Application Security |
| **Objectives and Outcomes** |
| This module directly supports **highlighted** course outcome(s)  Students who complete this course successfully will be able to   1. Describe current and emerging database models and technologies; 2. Develop functions and procedures for data manipulation and database access auditing; 3. Describe database monitoring and performance tuning; 4. **Describe database security and administration issues, including backup and recovery;** 5. Explain the concepts of data warehousing and data mining   **Module outcomes and activities:**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | After completing this module, students will be able: | create and remove database user | create Profiles and Roles | GRANT, REVOKE , DENY Permissions to users and roles | Use Views and Stored Procedures for Security | | Readings | introduced | introduced | introduced | introduced | | Practice exercises | reinforced | reinforced | reinforced | reinforced | | Lab |  | mastered | mastered | mastered | |
| **Assigned Reading** |
| 1. Managing Security for Oracle Database Users <https://docs.oracle.com/database/121/DBSEG/users.htm#DBSEG99778> 2. Guidelines for choosing account names <http://www.indiana.edu/~edss/Services/Naming/nvgbns.html> 3. Oracle security check lists <http://web.nvd.nist.gov/view/ncp/repository?keyword=oracle&startIndex=0> 4. Multitenant Architecture <https://docs.oracle.com/database/121/CNCPT/cdbovrvw.htm#CNCPT89234> |
| **Optional Reading** |
| 1. Keeping Your Oracle Database Secure <https://docs.oracle.com/database/121/DBSEG/guidelines.htm#DBSEG009> 2. Database application security <http://download.oracle.com/oll/tutorials/SQLInjection/index.htm> 3. Misconfigured By Default <https://s3-us-west-2.amazonaws.com/utoug.documents/Training+Days+2017/utoug17_misconfig_default.pdf> 4. Oracle Security for DBAs and Developers <http://morganslibrary.com/pres/azora/azora1711_db_sec.pdf> |
| **Assessments and Assignments** |
| 1. Lab (10 points) |
| **Topics** |
| Open the navigation pane |
| **Module Checklist** |
| This is the suggested order of the completion of this module.  Save a copy of this file on your computer and make notes in this document while you are completing your assignments. Use the table below to keep track of your progress.   |  |  | | --- | --- | | **Activity** | **Completion** | | Read this module and assigned materials (2 hour) | NO | | Complete all exercises from the module (2 hours) | NO | | Complete the lab (1 hours) | NO | | Complete a feedback section at the end of the module | NO | | Read feedback provided for your discussion and lab. | NO | |
|  |

# Operating System Security

Both Oracle and SQL Server allow users to be authenticated by Operating System to login to the Database without having to supply password. It is advisable to limit the number of Operating System users and to l**imit the privileges of the operating system accounts** (e.g administrative, root-privileged ) to the least privileges needed for the user's tasks.

It is also advisable to disable all unnecessary operating system services. Both UNIX and Windows platforms provide a variety of operating system services, most of which are not always necessary. Such services include FTP, TFTP, TELNET, TIME, and so forth. Be sure to close both the UDP and TCP ports for each service that is being disabled. Disabling one type of port and not the other does not make the operating system more secure.

Always apply all relevant and current security patches for the Operating System

# Database Security

Database Administrators responsibility is to implement and manage Database Security. This includes defining and determining the following:

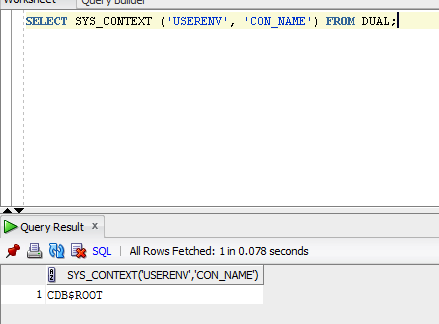
* Who is allowed access to the database
* How a user will be authenticated
* The authority level a user is granted
* The data that a user is allowed to read and/or modify
* The database objects a user is allowed to create, alter, and/or drop

# Oracle pluggable database

The multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a non-CDB. All Oracle databases before Oracle Database 12c were non-CDBs.

Each container has a unique ID and name in a CDB. You can use the CON\_ID and CON\_NAME parameters in the USERENV namespace to determine the current container ID and name with the SYS\_CONTEXT function. To find the current container name:

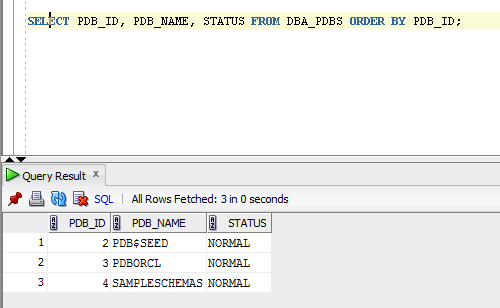
SELECT SYS\_CONTEXT ('USERENV', 'CON\_NAME') FROM DUAL;



Default container in CCSE Vlab installation.

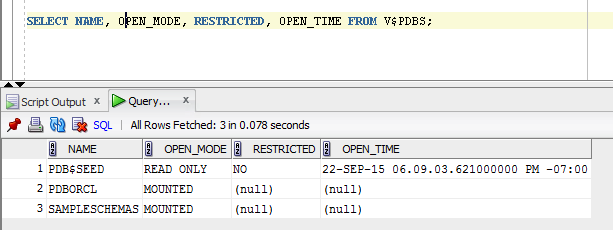
To find Container ID, Name, and Status of Each PDB

SELECT PDB\_ID, PDB\_NAME, STATUS FROM DBA\_PDBS ORDER BY PDB\_ID;



To find which PDB is opened

SELECT NAME, OPEN\_MODE, RESTRICTED, OPEN\_TIME FROM V$PDBS;



PDBORCL is not opened, so if we need to use sample schemas, we have to:

1. Open PDBORCL alter pluggable database pdborcl open;
2. Alter session and set container to PDBORCL  
    alter session set container=pdborcl;

Every CDB has the following containers:

Exactly one root CDB$ROOT that keeps metadata and common users (users known in every container. The username must start with c## e.g. c##USER1.

* Exactly one seed PDB PDB$SEED to create new PDBs.
* Zero or more user-created PDBs (e.g. PDBORCL)

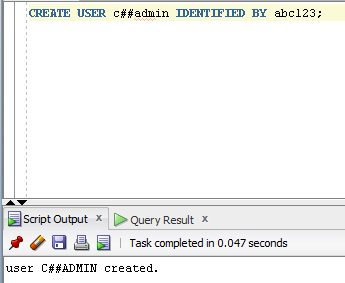
# Who is allowed to access Oracle database

To allow users to access the database you have to create users. In a multitenant environment, you can create users called common users who have access to the entire CDB, or you can create local users who are specific to a PDB.

**Common Users**

A common user username must start from c##

CREATE USER c##admin IDENTIFIED BY abc123 container = all;



To test the account use SQL\*Plus

Username: c##admin

Password: abc123



In order to connect to the Oracle database, a user must have CREATE SESSION privilege.

GRANT CREATE SESSION TO c##admin;

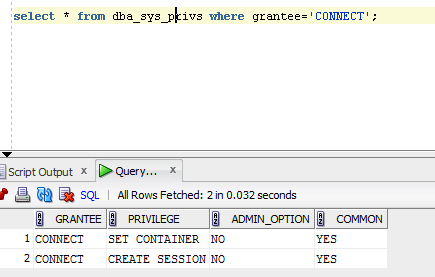
Or

GRANT CONNECT TO c##admin;

CONNECT is a role, to check what privileges are granted to this role, run

select \* from dba\_sys\_privs where grantee='CONNECT';

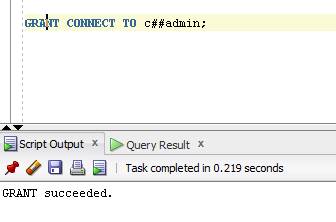
The only two system privileges CONNECT role has is "CREATE SESSION" and "SET CONTAINER" see below



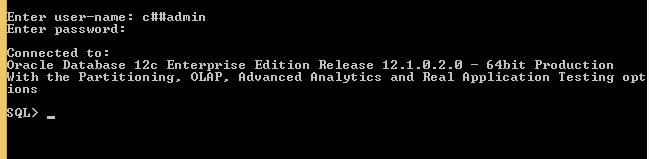
It will be better to run

GRANT CONNECT TO c##admin container = all;

Then this user will be able to change containers.



To check that c##admin can connect, use SQL\*Plus again



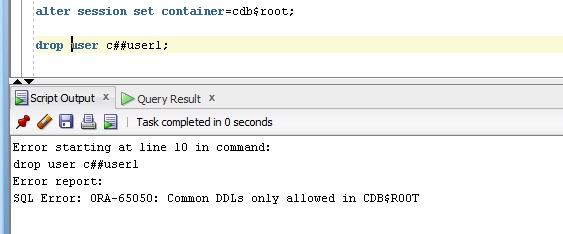
Changing container alter session set container=pdborcl;

The container must be opened (sys as sysdba run "alter pluggable database pdborcl open;").

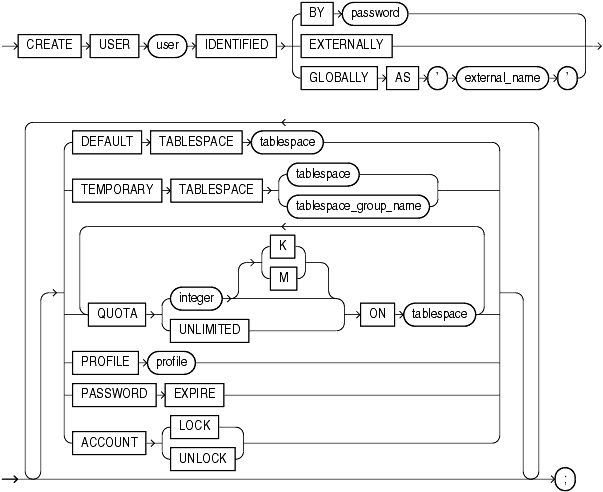
Now setting the container to PDBORCL is successful



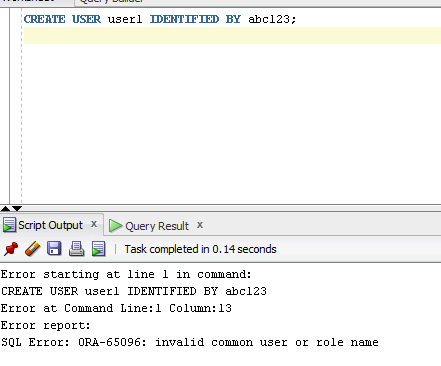
To drop the common user, you have to be in the root



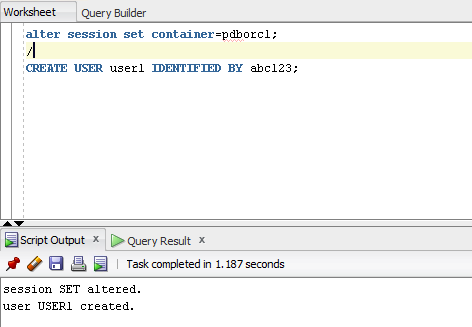
**Local Users**



When you try to create a local user user1 and receive an error message "Invalid common user name or role", it means that you are still in the CDB$ROOT container.



Change the container alter session set container=pdborcl;



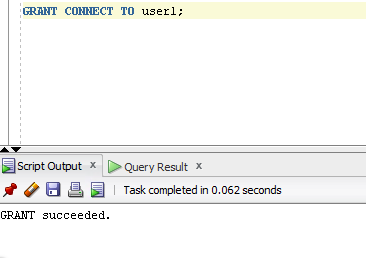
For the local user we can either

GRANT CREATE SESSION TO user1;

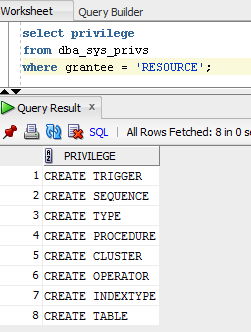
Or

GRANT CONNECT TO user1;

I will grant connect



There is another popular role RESOURCE

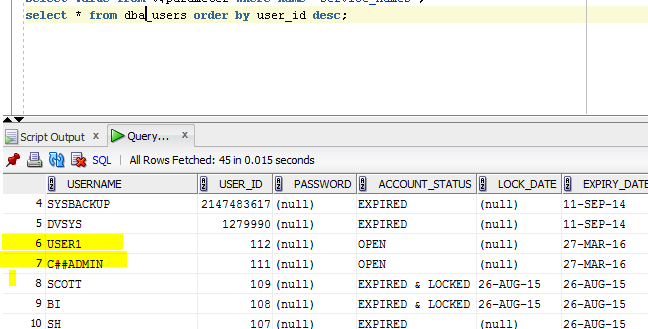


If you would like your user to have all above privileges, you can run:

GRANT RESOURCE TO user1;

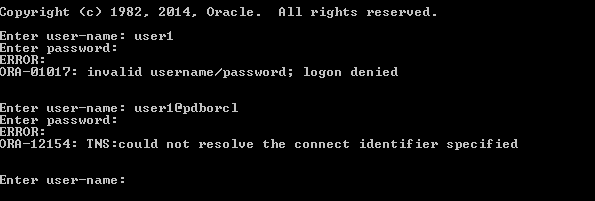
To see all users in the container, use

SELECT \* FROM DBA\_USERS;



Both local user ***user1*** and common user ***c##admin*** are listed.

Use SQL\*Plus to check if user1 can connect

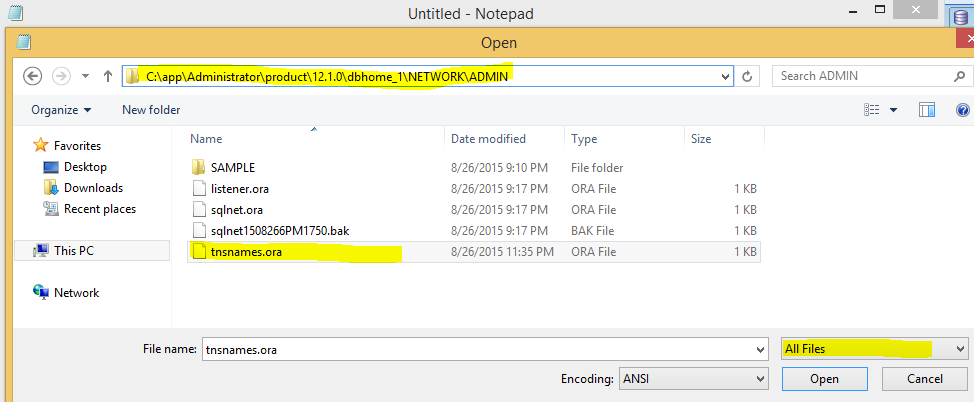


You will not be able to use just **user1** as the username; you have to specify the container PDBORCL.

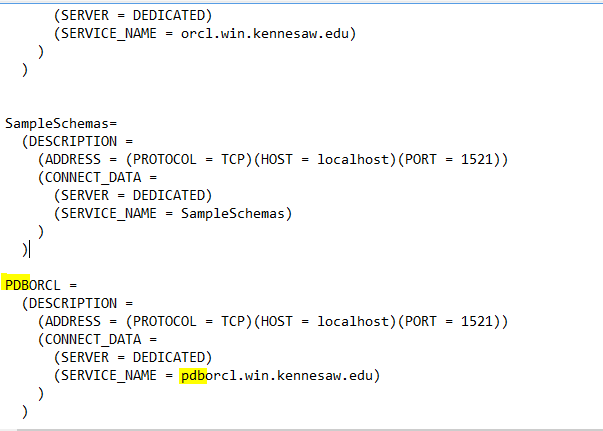
ORA-12154 error states that Oracle does not know where to find the container. ***Tnsnames.ora*** file keeps all Oracle network configurations for connecting clients, it is located at

C:/app/Administrator/product/12.1.0/dbhome\_1/network/admin/

Use notepad to open the file

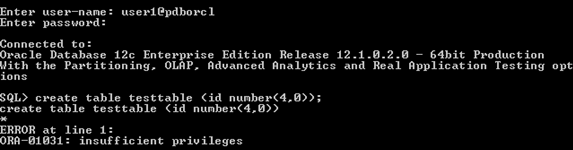


Copy section for ORCL, paste it below ORCL and change ORCL to PDBORCL and orcl.win.kennesaw.edu to pdborcl:



Save the changes.

Login as user1@pdborcl



You will not be able to create a table because

1. no quota a
2. no default tablespace
3. did not grant create table permission

Example :

CREATE USER steve

IDENTIFIED BY abc123

DEFAULT TABLESPACE users

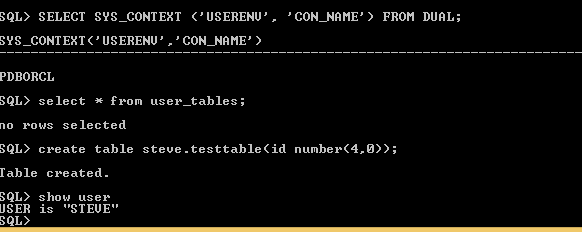
TEMPORARY TABLESPACE temp

QUOTA 15M ON users

PROFILE default;

GRANT connect to steve;

GRANT create table to steve;



**To Remove User**

Use the DROP USER statement to remove a database user and optionally remove the user's objects. Oracle Database does not drop users whose schemas contain objects unless you specify CASCADE or unless you first explicitly drop the user's objects.

If the user's schema contains tables, then Oracle Database drops the tables and automatically drops any referential integrity constraints on tables in other schemas that refer to primary and unique keys on these tables.

DROP USER steve; --drops user whose schema does not contain objects

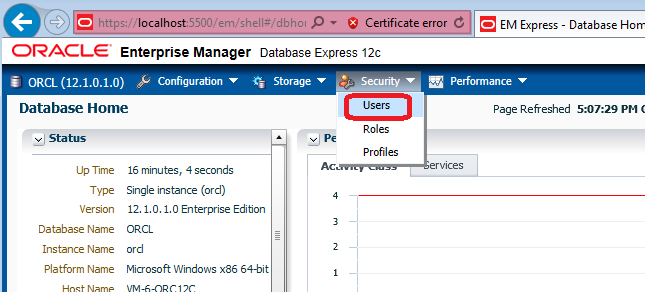
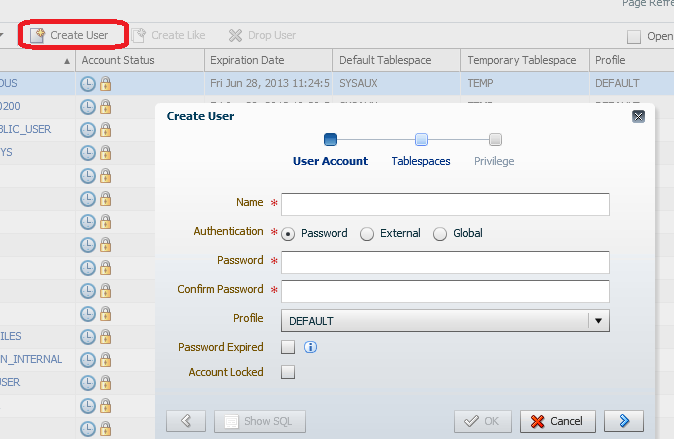
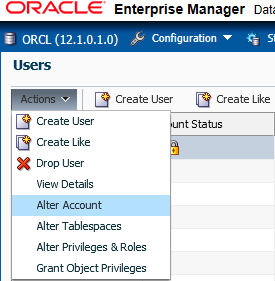
DROP USER steve CASCADE; --drops user and all schema objects

When you want to prevent a user from accessing a database, but still want to retain the database objects owned by the user, then instead of dropping the user lock his account, or revoke CREATE SESSION privilege from the user.

alter user steve account lock;

REVOKE CONNECT, CREATE SESSION from steve;

**Oracle 12c**

* Enterprise Manger is part of OracleServiceORCL
* In browser enter <https://localhost:5500/em> and logon as ‘system’
* On the home page select Security - Users
* 
* After you created the user, you can modify any values you set (alter account) or grant privileges  
  
* To delete account chose drop user.

# Password Policies & Authentication

Policies

* Enforce password expiration policy to have users change their passwords at regular intervals
* Enforce Password History to prevent users from reusing the old password, or allow reuse only after a certain number of times password has been changed.
* Lock the account after a certain number of invalid log-on attempts are made
* Avoid users from using simple passwords by enforcing password complexity, for example by accepting only a certain combination of characters and numerals
* Passwords should be always stored encrypted format. Whenever a user logs in, the password entered is encrypted initially, then compared to the stored encryption of the password associated with the user's login name. A match succeeds and a mismatch fails

Authentication - How a user will be authenticated

A user must be authenticated before he is allowed to create a session in the database.

In Oracle, user can be authenticated by:

**The Database(IDENTIFIED BY password):** User is authenticated by Oracle, by means of a password specified when the user was created. Passwords are stored in an encrypted format.

**The Operating System(IDENTIFIED EXTERNALLY):** User is authenticated by the Operating System although Oracle maintains the user account. Database password is not used for this type of authentication.

**SSL-Secure Sockets Layer (IDENTIFIED GLOBALLY)** : Users are authenticated and managed outside the database by a directory service

Authorization - The authority level a user is granted

Authority can be set using privileges, roles and profiles**.**

# Oracle Privilege

A privilege is the permission to execute a particular type of SQL statement, or to access a database object.

A DBA should grant a privilege only to a user who absolutely requires that privilege to accomplish necessary work. Excessive granting of unnecessary privileges can compromise security.

GRANT SELECT, INSERT ON emp TO user1, user2,user3;

GRANT SELECT ON emp TO user4,user5;

GRANT SELECT,INSERT,UPDATE,DELETE ON emp to user6;

To Revoke a Privilege:

REVOKE DELETE ON emp FROM user6;

# **Oracle Role**

A role is a convenient way to manage a set of privileges that are granted to groups of users**.** Rather than granting the same set of privileges explicitly to several users, you can grant the privileges for a group of related users to a role, and then only the role needs to be granted to each member of the group. You can selectively enable or disable the roles granted to a user. This allows specific control of a user's privileges in any given situation.

CREATE ROLE SALES\_REP;

CREATE ROLE SALES\_CLERK;

CREATE ROLE SALES\_MANAGER IDENTIFIED BY strong\*password;

COMMIT;

GRANT SELECT, INSERT ON emp TO SALES\_REP;

GRANT SELECT ON emp TO SALES\_CLERK;

GRANT SELECT,INSERT,UPDATE,DELETE ON emp TO SALES\_MANAGER;

COMMIT;

GRANT SALES\_REP TO user1,user2,user3;

GRANT SALES\_CLERK TO user4,user5

GRANT SALES\_MANAGER TO user6;

To Revoke a Privilege

REVOKE DELETE ON emp FROM SALES\_MANAGER;

A user automatically has all object privileges for schema objects contained in his or her schema. A user can grant any object privilege on any schema object he or she owns to any other user or role.

In addition to the above, a privilege can be granted to a user ‘WITH GRANT OPTION’ (used for object privileges like select, update, insert, delete) or ‘WITH ADMIN OPTION’(used for system privileges like create)

GRANT SELECT ON emp TO user1; -- user1 can select rows from emp, but cannot grant SELECT on emp to any other user.

GRANT SELECT ON emp TO user1 WITH GRANT OPTION; --user1 can select rows from emp, as well as he can grant SELECT privilege on emp to any user.

To access information about grants of privileges and roles, you can query the following data dictionary views:

| **View** | **Description** |
| --- | --- |
| DBA\_COL\_PRIVS  ALL\_COL\_PRIVS  USER\_COL\_PRIVS | DBA view describes all column object grants in the database. ALL view describes all column object grants for which the current user or PUBLIC is the object owner, grantor, or grantee. USER view describes column object grants for which the current user is the object owner, grantor, or grantee. |
| DBA\_TAB\_PRIVS  ALL\_TAB\_PRIVS  USER\_TAB\_PRIVS | DBA view lists all grants on all objects in the database. ALL view lists the grants on objects where the user or PUBLIC is the grantee. USER view lists grants on all objects where the current user is the grantee. |
| ROLE\_SYS\_PRIVS | This view contains information about system privileges granted to roles. Information is provided only about roles to which the user has access. |
| ROLE\_TAB\_PRIVS | This view contains information about object privileges granted to roles. Information is provided only about roles to which the user has access. |

# Oracle Profile

A profile is a named set of resource limits that restrict a user's database usage and instance resources, as well as password practices. For profiles to take effect, resource limits must be turned on for the database as a whole. You can assign a profile to each user, and a default profile to all others. Each user can have only one profile; creating a new one supersedes any earlier one.

CREATE PROFILE user1 LIMIT

SESSIONS\_PER\_USER UNLIMITED --Number of concurrent sessions

CPU\_PER\_SESSION UNLIMITED --CPU time limit for a session, expressed in hundredth of seconds.

CPU\_PER\_CALL 3000 --CPU time limit for a call (a parse, execute, or fetch), expressed in hundredths of seconds.

CONNECT\_TIME 45 --total elapsed time limit for a session, expressed in minutes

IDLE\_TIME 20 --permitted periods of continuous inactive time during a session, expressed in minutes

LOGICAL\_READS\_PER\_SESSION DEFAULT --number of data blocks read in a session

LOGICAL\_READS\_PER\_CALL 1000 --number of data blocks read for a call to process a SQL statement

PRIVATE\_SGA 15K --amount of private space a session can allocate in the shared pool of the system global area (SGA), expressed in bytes

CREATE PROFILE user2 LIMIT

FAILED\_LOGIN\_ATTEMPTS 5 --number of failed attempts to log in to the user account before the account is locked

PASSWORD\_LIFE\_TIME 60 --number of days the same password can be used for authentication

PASSWORD\_REUSE\_TIME 60 --the number of days before which a password cannot be reused

PASSWORD\_REUSE\_MAX 5 --number of password changes required before the current password can be reused

PASSWORD\_VERIFY\_FUNCTION verify\_function --lets a PL/SQL password complexity verification script be passed as an argument to the CREATE PROFILE statement

PASSWORD\_LOCK\_TIME 1/24 --the number of days an account will be locked after the specified number of consecutive failed login attempts

PASSWORD\_GRACE\_TIME 10; --number of days after the grace period begins during which a warning is issued and login is allowed

PROFILE can be assigned to user when creating the user, or later by using ALTER USER statement.

Profiles only take effect when resource limits are "turned on" for the database as a whole

ALTER SYSTEM SET RESOURCE\_LIMIT = TRUE;

# Using Views for Security

Views can serve as security mechanisms by restricting the data available to users. For example, suppose the SSN column in a table contains confidential information, but the rest of the columns contain information that should be available to all users. You can define a view that includes all of the columns in the table with the exception of the sensitive SSN column. As long as table and view have the same owner, granting SELECT permissions on the view allows the user to see non confidential columns in the view without having any permissions on the table itself. Permission to access the subset of data in a view must be granted, denied, or revoked, regardless of the set of permissions in force on the underlying table(s)

CREATE view emp\_all AS

SELECT first\_name, last\_name, middle\_initial,

street\_address, state, Zip\_code

FROM employee; -- select only the columns which should be viewed by your users, this is vertical restriction

If you want your users to view only information corresponding to a particular department, then

CREAT view emp\_dept20

AS

SELECT first\_name, last\_name, middle\_initial,

street\_address, state, zip\_code

FROM employee

WHERE deptno = 20 – select only the rows where deptno=20, this is horizontal restriction

Then GRANT SELECT privilege on emp\_all view to appropriate users.

# Using Stored Procedures for Security

Stored Procedures offer an opportunity to improve your database security. If you do not GRANT read (SELECT) and write (INSERT, UPDATE, and DELETE) access to all database tables and provide only EXECUTE privileges to your stored procedures, you can ensure that your data access is more controlled. Ad hoc SQL no longer will be able to access your data.

For example, suppose every Clerk in an organization needs to run the query:

SELECT first\_name, last\_name, middle\_initial, street\_address, state, zip\_code FROM employee WHERE deptno = '20'

This means you have to specifically GRANT SELECT permission on Inventory to all Clerks, and then Clerks have read access to all columns of employee table.But the employee table also contains SSN and Salary information, which you do not want to be disclosed to the Clerks.

We can achieve this through the use of a stored procedure. Let's create a procedure called uspGetEmp in *SQL Server*, that retrieves the employee information for a given dept. Here's the SQL code:

CREATE PROCEDURE uspGetEmp  
@dept int  
AS  
SELECT first\_name, last\_name, middle\_initial, street\_address, state, zip\_code  
FROM employee  
WHERE deptno = @dept;

Now, we will not GRANT any permission on employee table to Clerks. Instead we will GRANT EXECUTE on the stored procedure uspGetEmp, so they can do exactly what we want them to do.

GRANT EXECUTE ON OBJECT : : uspGetEmp to Clerk;

To get information of employees of deptno 20, the Clerk can execute the procedure as follows

EXECUTE uspGetEmp '20';

We can even remove input parameter, but check what is the department ID of the clerk executing uspGetEmp procedure and use his/her department instead. SESSION\_USERID is a database user name by which the current user is authenticated (Oracle)

SELECT sys\_context('USERENV', 'SESSION\_USER') FROM dual;.

It might be different than SELECT USER FROM DUAL; see <http://www.red-database-security.com/exploits/oracle_exploit_dbms_scheduler_select_user.html> and <http://forums.oracle.com/forums/thread.jspa?threadID=901895>

# Database Application Security

Most modern applications are database driven. That means application logic is built on top of database that manages the data users create, delete, modify, and query. Depending on how the application code is designed and written, the security of the underlying database may be compromised. SQL Injection involves carefully passing crafted user input to an application. The idea is to convince the application to run SQL code that was not intended. For example, you have an account with some website, and this website allows you to view your details when you enter your email id in a given text box.

The application in turn might run a query like this, to fetch your details:

***Select \* from users where email =’jdow@kennesaw.edu’*** *-- jdow@kennesaw.edu* is the string yousupplied.

If a user types the following string instead of his email id jdow@kennesaw.edu***’ or ‘1’=’1***

As a result, the SQL statement that will be executed will look like this:

***Select \* from users where email =’jdow@kennesaw.edu’ or ‘1’=’1’;***

This statement will let the user see records of all users that exist in the users table.

The user might guess the table name and try to drop the table.

**jdow@kennesaw.edu’ or ‘1’=’1’; DROP TABLE users; --**

SQL Injections can be prevented if the application uses parameterized query or Prepared Statements for querying the database. See <http://ksuweb.kennesaw.edu/~speltsve/files/sql_files/prepared_statement.doc>

The roles assigned to application users and privileges assigned to roles should be thoroughly checked to ensure privileges just enough to get the user’s work done are granted.

# Lab

**Objectives**

Learn how to administer users in Oracle

Lab overview

In this lab you will learn how to use DDL and Oracle Enterprise server to create user accounts, modify, monitor and delete them. You will take the first look at object permissions in Oracle.

(Questions 1-4, 6-8 one point, question 5 three points for each correct answer: code and screenshot)

1. Create a common user c##admin

query you used to create the user

screenshots to prove that user was created, can connect and can create tables and insert data.

1. Use system account to create a table system.TEST\_TABLE\_SYSTEM with at least one column and add at least one row. Do not forget to commit.

query you used to create the table

screenshots to show the content

1. As c##admin try to select data from the table you created in #2.

What is the result? Explain.

1. Fix the error you received in #3

query you used

screenshots to show results

1. Use Oracle data dictionary to check what permissions you granted to the user c##admin.

query you used to find system and table permissions

screenshots to show the results

1. Alter the common user c##admin

query you used to alter the user

screenshots to show the results

1. Drop the common user c##admin

query you used to drop the user

screenshots to prove that user was dropped

1. What procedures would you follow if you were told that an employee was terminated? Hint: This is a tricky question and includes not only Oracle commands.

Explanations

SQL queries you plan to use

**Feedback:**

Difficulty (-2 - too easy ... 0 - just right ... 2 - too hard)

Interest level (-2 - low interest ... 0 - just right ... 2 - high interest)

Time to complete (min)

Make a suggestion to improve