

Creating Roles

Overview:

This document provides details of how to create, alter and delete Roles within the Oracle database. Roles provide permissions to do what a user needs to do. This enables them to do their daily tasks. Roles are an essential part of database security as they help enable the principle of least privilege. A user should always only be assigned the Role(s) they need to complete their jobs. For example, if a user doesn't typically require access to the data dictionary views, that permission should not be provided in that Role.

Restricted Use:

For security reasons, some permissions have not been provided to students to avoid accidental deletion or modification of other user accounts.

Foundational Definitions:

In order to understand how users receive access to Oracle and other database systems, it is important to understand and differentiate authentication and authorization.

When a system authenticates a user, it verifies the identity of the user attempting to access your database or system. This process establishes a trust relationship for further interactions. When a user is authorized requests to access a particular resource has been granted or possibly denied. Authorization includes the execution of rules that determines what functionality and data users may access. Roles provide the authorization in Oracle.

Authentication may have multiple components including the encryption of passwords, The Oracle Database also encrypts passwords during transmission to ensure the security of network authentication, password hashing, delayed retry login attempts, complexity checking and others.

When creating a user, notice the password is in clear text.

```
CREATE USER user_name IDENTIFIED BY password;
```

There are advanced security options in Oracle such as the Oracle Wallet that can be used to be the creation of the user is using an SSL for the connection. For Enterprise versions of Oracle be sure to implement this feature for all connections such that data is protected in transit.

To slow down password guessers, when a user tries to log in to Oracle Database multiple times using an incorrect password, Oracle Database delays each login by one second. This feature significantly decreases the number of passwords that an intruder would be able to try within a fixed time period when attempting to log in. This feature applies for attempts that are made from different IP addresses or multiple client connections. Oracle Database mitigates concurrent password guessing attacks, but this can simultaneously leave the account vulnerable to denial-of-service (DoS) attacks. However; this may be mitigated with Failed_Login_attempt parameters in the Profile settings.

User login patterns should be monitored and be used to trigger automatic detection of unusual and possibly suspicious patterns. For example, using the following two SQL statements help identify login and current usage.

```
SELECT LAST_LOGIN FROM DBA_USERS WHERE USERNAME = 'SDEV350STUDENT';
```

```
SELECT USERNAME from V$SESSION;
```

A series of security applications can be written using queries to the data dictionary in an attempt to identify possible misuse or even hijacking of an account. Preventing a malicious user from getting into your system is critical but as we all have heard with the hundreds of data breaches of the past years, we must also monitor to detect issues when a breach has already occurred to minimize damages.

If an issue is detected, you can lock a user's account until the matter is resolved using a SQL statement such as:

```
ALTER USER student1 ACCOUNT LOCK;
```

Roles versus Privileges:

Privileges are defined by the Oracle Database and provide the right for a user or role to execute a SQL statement, function or other Database object. A role is a category of a privilege.

Within Oracle, there are multiple categories of privileges. System privileges allow the grantee to perform standard administrator tasks in the database. Roles are several privileges and roles grouped together to allow easy assignment and distribution to users.

In addition, objects, tables, views, procedure and types all have privileges associated with them. For example, a table has both Data Manipulation Language (DML) and Data Definition Language (DDL) privileges associated with it. A view can also have privileges. We have previously utilized some of those privileges when selecting from the data dictionary views.

Thousands of PL/SQL and Java procedures and functions exist within the Oracle. These objects need specific privileges to be executed. For example, you really would not like all users to be able to shut down the database or execute another system critical function.

You can use the ALL_PROCEEDURES Data dictionary view to search for the available functions and procedures within the Oracle database. For example, the following query, groups and counts the number of procedures associated with each database user.

```
select owner, count(*) from all_procedures group by owner;
```

The results of running this query on the Oracle Student database is shown in figure 1.

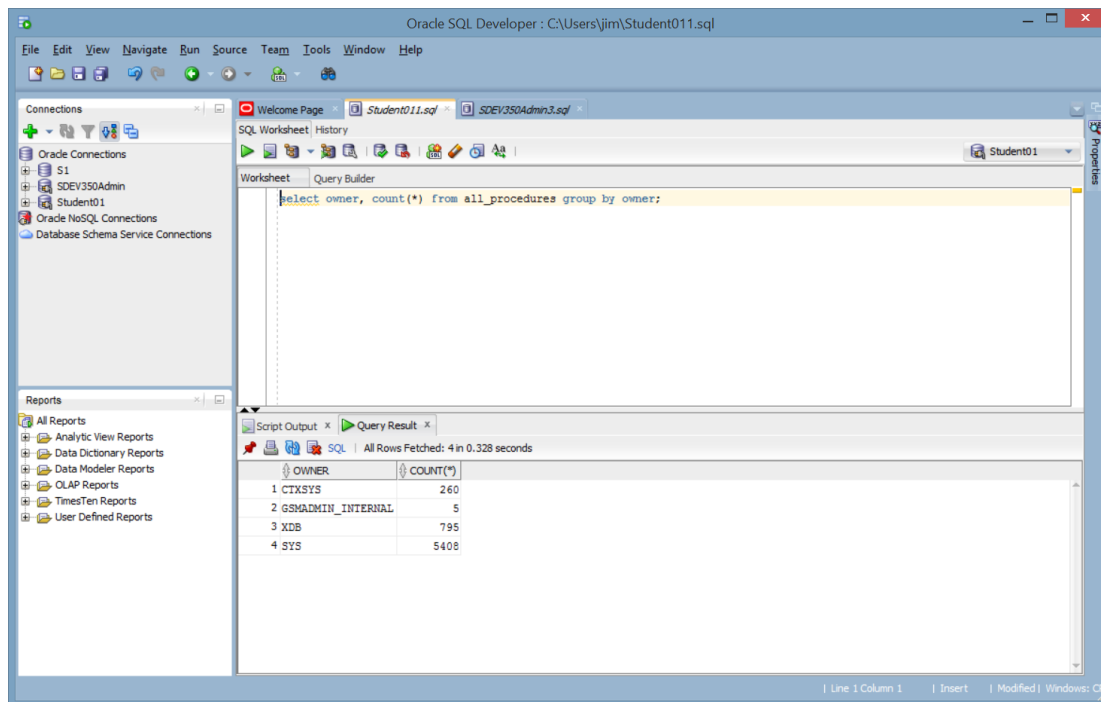


Figure 1 Counting Procedures for each Oracle User

Types Array of values for a column (VARRAYS), Tables within Tables (NestedTables) and other user created types. Privileges are needed to be able to use and work with these special types.

Using Roles:

Roles can be applied to schemas, tables, table rows and other database objects. With level of granularity, you can clearly define and provide access to only the objects and activities a user needs. Roles group together privileges or even other roles to facilitate the granting of multiple privileges to users.

Roles are typically created by administrators or someone with extended permissions and should always be based on business rules or company policies. A formal process should be used to gather requirements gathering based on business rules, job functions and security best practices and standards to design database roles. The roles should be reviewed regularly and updated as appropriate.

System Privilege Examples:

Several Roles exist that can be granted to other users assuming the user has the ability to grant that role. The DBA role is a very powerful role which should always be granted sparingly and with much thought. Common DBA tasks include backup and recovery and encryption key management for Transparent Data Encryption (TDE). Less common tasks but available in DBA privileges include Drop Database and Shut down. Clearly tasks like these need to be given to trusted users. Only grant administrative privileges to trusted (and experience) users.

When privileges are granted there is an option to Grant ANY. Grant ANY sets privileges for an entire category of objects in the database. For example Create Any Table, and Create Any Procedure. Most importantly, when ANY is used, the privilege is not restricted to the schema in which it was created. This

can be very dangerous in the wrong hands. For example, if user Student01 has the CREATE ANY PROCEDURE privilege and creates a procedure in the schema Student02, then the procedure will run as Student02. However, Student02 may not be aware that the procedure Student01 created is running as him.

The Public Role is also worth discussing. Public is a Role available in Oracle. By default, it has no privileges associated with it. However; you can grant privileges to the PUBLIC role. When you do, this makes the privileges available to every user in the Oracle database.

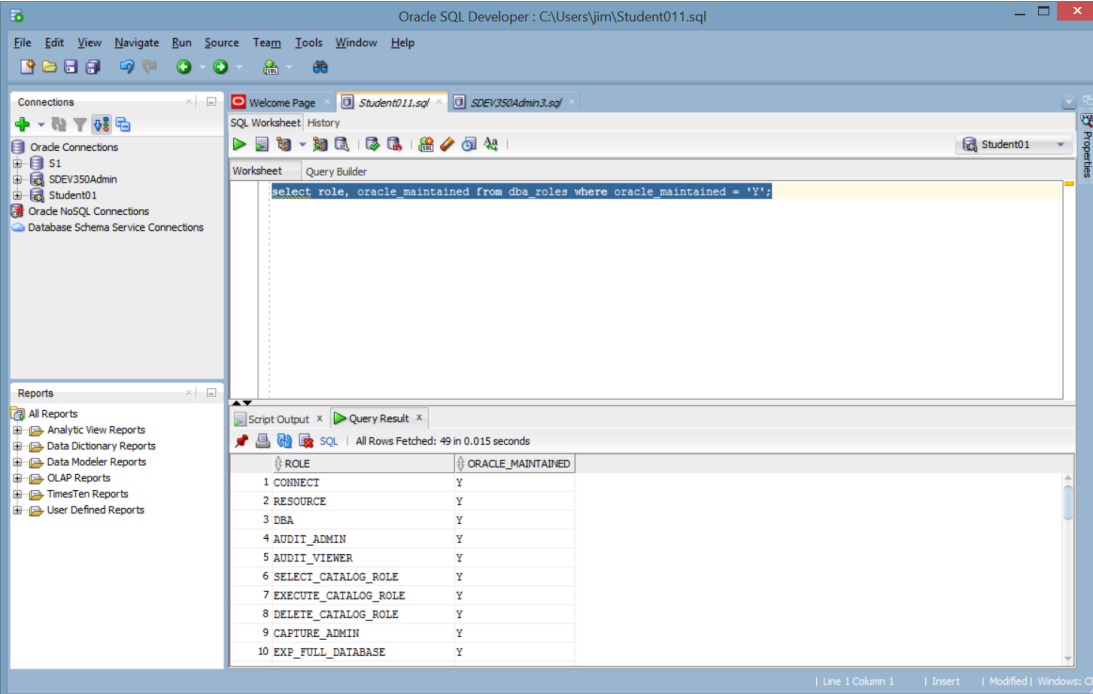
This is another privilege that should be used with great caution. For example, if Student01 has the CREATE PUBLIC SYNONYM system privilege, he/she could redefine an interface that everyone else uses (e.g. application login). At some point the PUBLIC SYNONYM could be pointed to replace the old interface. In this manner, instead of accessing the correct interface, users would access the interface of Student01, which could possibly perform illegal activities such as stealing the login credentials of users.

Oracle Maintained Roles:

Several roles are available within the database and are maintained and created by Oracle. You can use the DBA_ROLES data dictionary view to Roles and additional information. For example, the following query will retrieve Oracle maintained roles.

```
select role, oracle_maintained from dba_roles where oracle_maintained = 'Y';
```

Figure 2 shows the results of running this query from the student account.



The screenshot shows the Oracle SQL Developer interface. The 'Query Result' window displays the results of the query 'select role, oracle_maintained from dba_roles where oracle_maintained = 'Y';'. The results are shown in a table with two columns: 'ROLE' and 'ORACLE_MAINTAINED'. There are 10 rows of data, all with 'ORACLE_MAINTAINED' set to 'Y'.

ROLE	ORACLE_MAINTAINED
1 CONNECT	Y
2 RESOURCE	Y
3 DBA	Y
4 AUDIT_ADMIN	Y
5 AUDIT_VIEWER	Y
6 SELECT_CATALOG_ROLE	Y
7 EXECUTE_CATALOG_ROLE	Y
8 DELETE_CATALOG_ROLE	Y
9 CAPTURE_ADMIN	Y
10 EXP_FULL_DATABASE	Y

Figure 2 DBA Roles

Descriptions of the Predefined in Oracle 12C database are listed and described in more detail in this URL:

<https://docs.oracle.com/database/121/DBSEG/authorization.htm#GUID-A5B26A03-32CF-4F5D-A6BE-F2452AD8CB8A>

The predefined roles are handy to use but can be dangerous to assign as often more privileges than needed are provided. Before granting these roles, be sure you clearly know each privilege that is associated with the role and that a user needs those privileges. In most cases, creating your own custom roles is better.

Creating Roles:

You must have the Create Role privilege to be able to Create roles.

The simplest form of Create follows this syntax example where a role named devops is created.

```
CREATE ROLE devops;
```

Figure 3 shows the additional options available when creating a role.

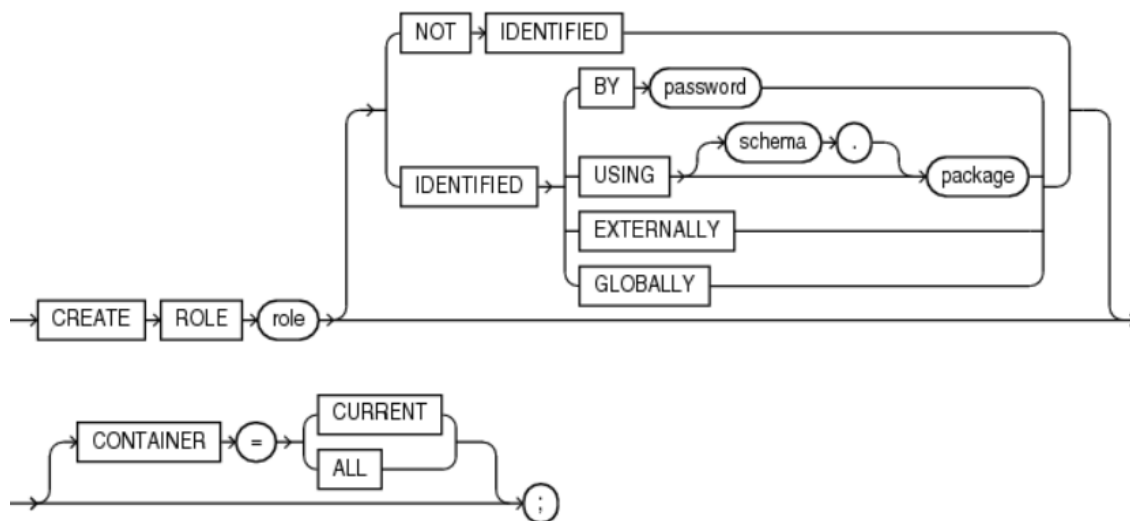


Figure 3 Create Role Syntax

Similar to the Create User syntax, options are available for using password or external credentials. In addition, Oracle 12C (and above) provide containers for isolating databases in the cloud. Although interesting and an important feature, we won't be using the containers portion of 12C for this class.

Privileges can be granted directly to a user or to a role. The role can then granted to one or more users. It is security best practice to grant privileges to roles and not to specific users.

For example, the following grant statements show granting a privilege to the alld devs role and then granting that role to a specific user named Joe.

```
Grant INSERT on Students to alld devs;
```

```
Grant alld devs to Joe;
```

Privileges should be very granular and reach the object level. For example, If a user only needs to read from the Students table and no other tables, the following syntax would suffice:

```
GRANT Select on Students to alldevs;
```

If the Role needed to Insert and Delete from Students, those privileges could be added as well.

```
GRANT Insert on Students to alldevs;
```

```
GRANT Delete on Students to alldevs;
```

To grant the alldevs role to Student01, the following SQL statement would work:

```
Grant alldevs to Student01;
```

All privileges associated with the alldevs role are now available to Student01.

To test this privilege, you would need to use the schema name concatenated with the table name. For example, if Student01 logged in and wanted to select from the Students table from the creator of the Students table, it would need to be to know the user (schema) of the creator. If Student02 create the students table and grant alldevs role to Student01, then Student01 would execute this statement to select from the Students table.

```
Select * from Student02.Students;
```

This is an important concept to understand from both a security and user functionality standpoint. With multiple users, each possible with a different schema, the Students table could exist in each of those schemas. You would need to use the schema.Tablename to properly access it.

Public Synonyms could be used as well but again, this should be used cautiously because of the possible security issues associated with making a table or database object available to all users.

The lab this week will allow you to become very comfortable and confident using Roles and assigning privileges to different users and then testing those privileges.

Altering a Role:

The Alter Role statement can be used to alter an existing Role.

You do need Alter Role permissions for a successful Alter call. If provided with the Alter Role privileges, running this statement would yield a successful response as shown in figure 4.

```
ALTER ROLE devops IDENTIFIED BY "devUser2_1";
```

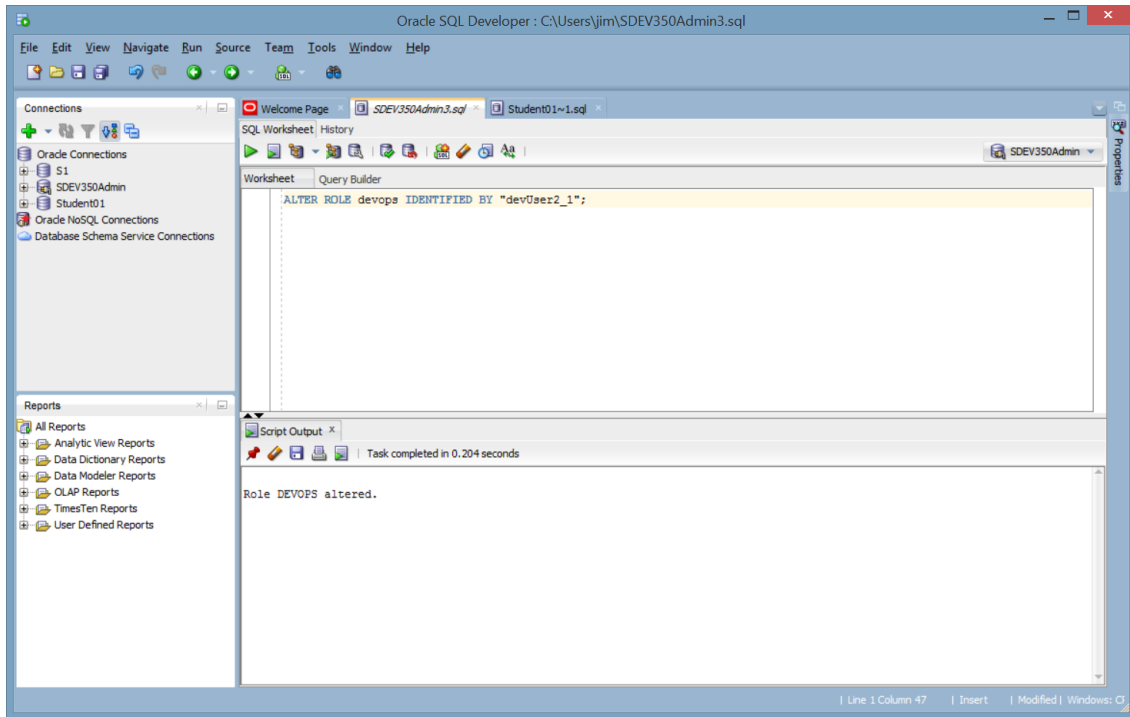


Figure 4 Altering a Role to add Credentials

Additional options available for the Alter Role syntax are shown in figure 5.

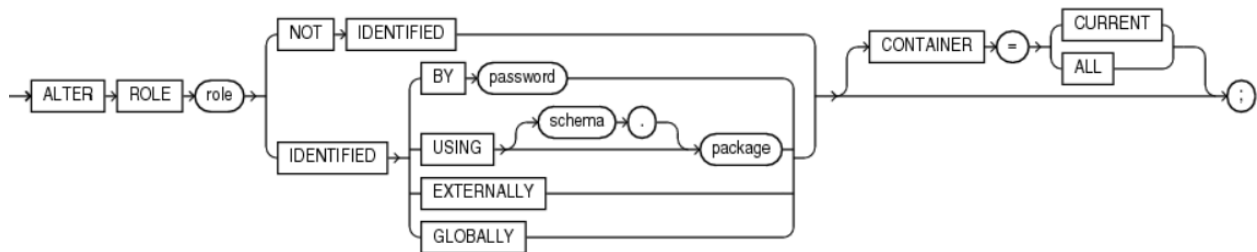


Figure 5 Alter Role Syntax

Drop a Role

With the proper Drop Role permissions, a role can be dropped (deleted). You do need the Drop role permissions for a successful Drop call.

Figure 6 shows the drop role options.

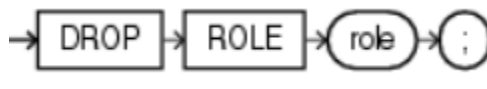


Figure 6 Drop Tablespace Syntax

Executing the following SQL statement will drop the devops Role (See figure 7).

```
Drop Role devops;
```

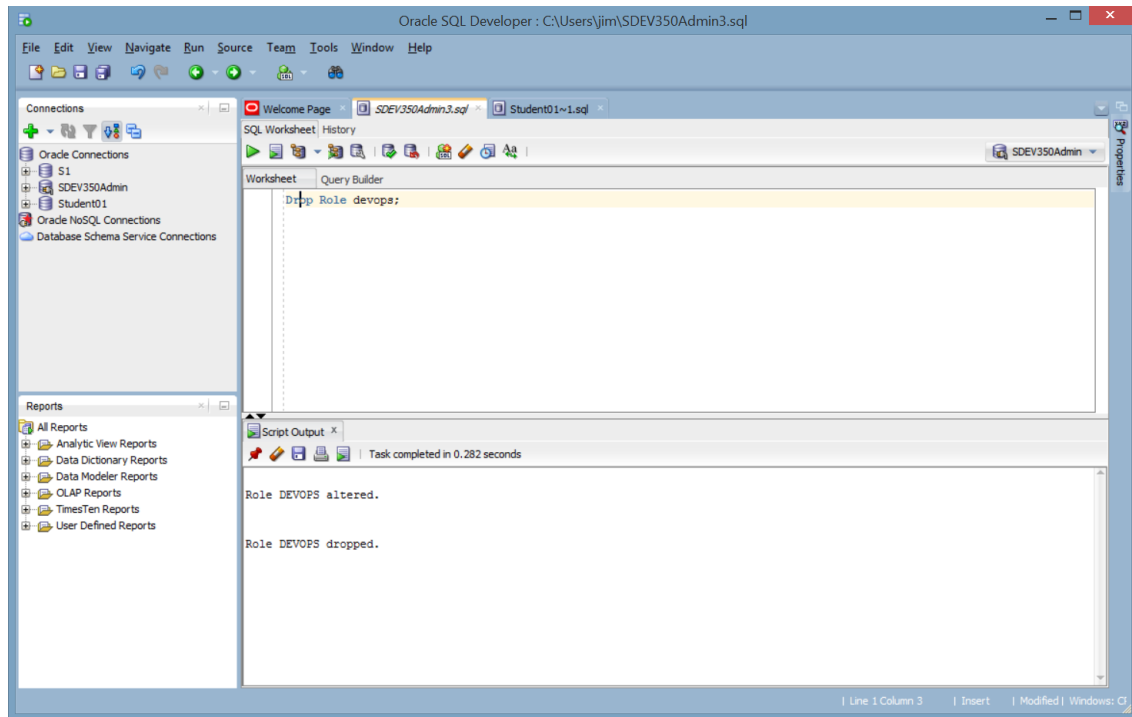


Figure 7 Dropping a Role