Table of Contents

[CHAPTER 22: TRIGGERS 2](#_Toc416108753)

[Theory 2](#_Toc416108754)

[AIM 3](#_Toc416108755)

[Lab Exercise 22: TRIGGERS 4](#_Toc416108756)

[1. Database Trigger 5](#_Toc416108757)

[2. Schema Trigger 12](#_Toc416108758)

[3. Enable or Disable Trigger 18](#_Toc416108759)

[4. Query, Change, and Compile Trigger 21](#_Toc416108760)

[5. Drop Trigger 23](#_Toc416108761)

[SUMMARY 25](#_Toc416108762)

[REFERENCES 26](#_Toc416108763)

[INDEX 27](#_Toc416108764)

# CHAPTER 22: TRIGGERS

## Theory

Triggers are stored subprograms just like stored function and procedures but they fired implicitly and automatically by Oracle. Therefore, triggers need two things to work:

1. **Item** (Level): Oracle allows triggers on different levels, for example
   1. Database

Database triggers are normally created to monitor or track changes on database. For example, if someone changes the database environment variables or logs on/logs out from the database.

* 1. Schema

Schema triggers tack the changes on schemas structure. Mainly, it tracks any DDL statements done on user's object.

* 1. Tables/Views

Table/view trigger is referred as DML trigger because it mainly tracks the changes done on user's object data, not structure.

1. **Event**: For each one of the previous items/levels, Oracle supports different events in which the triggers are fired in consequence. For example, a trigger created on a table may be fired on INSERT, UPDATE, or DELETE event. You also allowed to determine the exact timing point when which the trigger would fire; AFTER or BEFORE.

In addition to creating a trigger, you usually need to change, compile, and debug trigger. Occasionally, you may need to disable a trigger to allow for inapplicable data and then re-enable it again. Oracle supports enabling and disabling triggers.

Trigger and table constraints can enforce conditions on new data inserted or updated on tables. However, Oracle highly recommends using constraints when applicable.

## AIM

The AIM of the following exercise is to create simple triggers.

The steps involved will include:

* Database Trigger
* Schema Trigger
* Enable or Disable Trigger
* Change and Compile Trigger
* Drop Trigger

In general, lab exercises are done in sequential order. Thus, it is assumed that you successfully completed the previous labs. However, not all previous labs are required. Please be sure to run the following lab before proceeding:

* Installing Oracle Database 12c.

Estimated Completion Time:

17 minutes

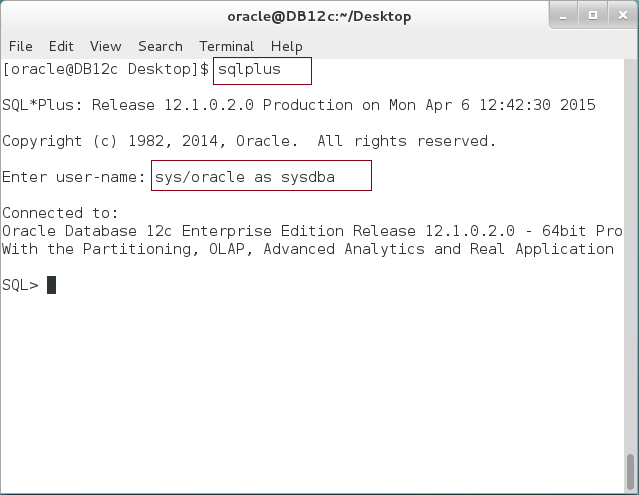
# Lab Exercise 22: TRIGGERS

|  |
| --- |
|  |

## Database Trigger

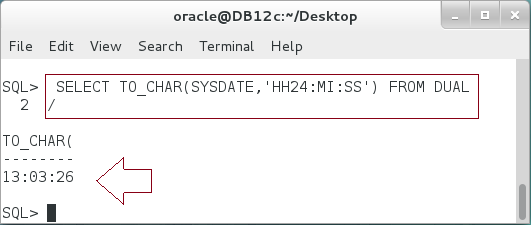
**Step 1:** Open the Terminal, open SQL\*Plus console and connect to hr schema.

|  |  |
| --- | --- |
| Command | Description |
| sqlplus | Open SQL\*Plus console. |
| sys/oracle as sysdba | connect to **sys** schema. |

****

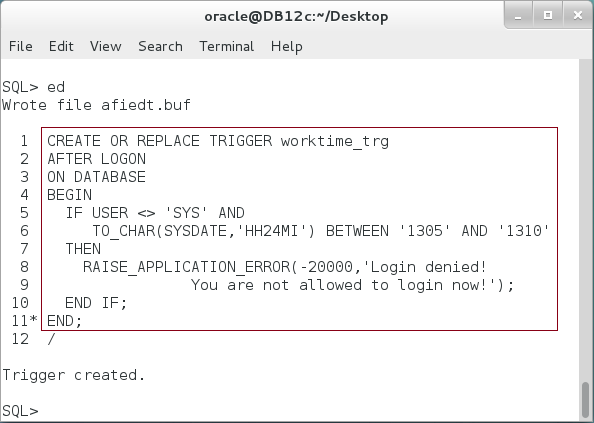
**Step 2:** Execute the following query:

|  |  |
| --- | --- |
| Command | Description |
| SELECT TO\_CHAR(SYSDATE,'HH24:MI:SS') FROM DUAL | Get the current time |
| / |  |

****

**Step 3:** We are going to create a database trigger, which is running after each logon on the database occurs. Execute the following PL/SQL block:

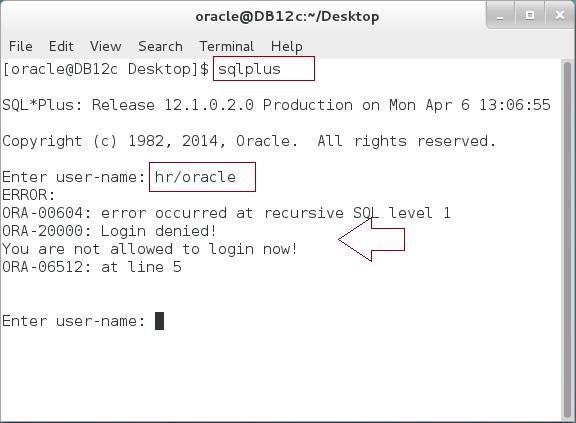
|  |  |
| --- | --- |
| Command | Description |
| **CREATE** OR REPLACE **TRIGGER** worktime\_trg | Create trigger |
| **AFTER LOGON** | Event: Logon |
| **ON DATABASE** | Level: Database |
| BEGIN |  |
| IF USER <> 'SYS' AND | Restrict accessing database after work time. |
| TO\_CHAR(SYSDATE,'HH24MI') BETWEEN **'1245'** AND **'1250'** |
| THEN |
| RAISE\_APPLICATION\_ERROR(-20000,'Login denied! |
| You are not allowed to login now!'); |
| END IF; |
| **END**; |  |
| / |  |

****

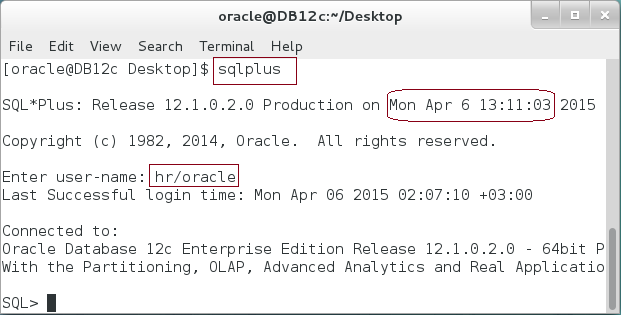
**Please note:** You must specify your own period instead of the period shown above.

**Step 4:** Open new Terminal and try to login with "HR" while the time is in the restriction period specified in the previous trigger:

|  |  |
| --- | --- |
| Command | Description |
| sqlplus | Run SQL\*Plus console |
| hr/oracle | connect as "HR" |

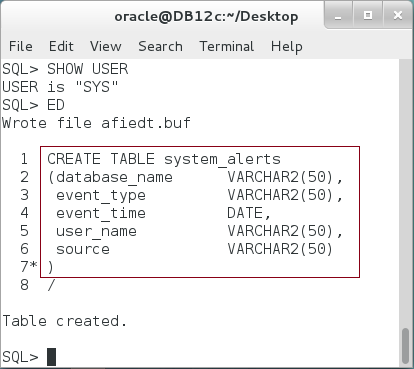


**Step 5:** Try to reconnect again after the time is passing by:



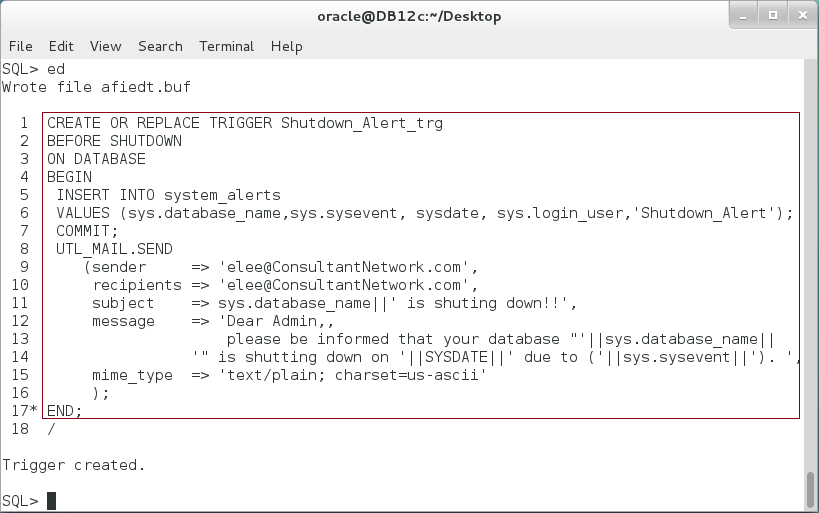
**Step 6:** We are going to create another database trigger to track database SHUTDOWN event. First, we need to create a table to store the alert log. Be sure to connect as sys and execute the following DDL statement:

|  |  |
| --- | --- |
| Command | Description |
| CREATE TABLE system\_alerts |  |
| (database\_name VARCHAR2(50), |  |
| event\_type VARCHAR2(50), |  |
| event\_time DATE, |  |
| user\_name VARCHAR2(50), |  |
| source VARCHAR2(50) |  |
| ) |  |
| / |  |

****

**Step 7:** Create a database trigger "" as shown below:

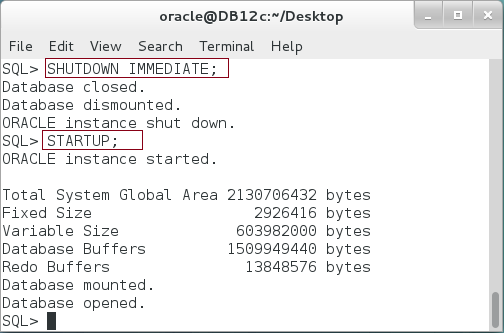
|  |
| --- |
| Command |
| **CREATE** OR REPLACE **TRIGGER** Shutdown\_Alert\_trg |
| **BEFORE SHUTDOWN** |
| **ON DATABASE** |
| BEGIN |
| **INSERT INTO** system\_alerts |
| VALUES (sys.database\_name,sys.sysevent, sysdate, sys.login\_user, 'Shutdown\_Alert'); |
| COMMIT; |
| **UTL\_MAIL.SEND** |
| (sender => 'elee@ConsultantNetwork.com', |
| recipients => 'elee@ConsultantNetwork.com', |
| subject => sys.database\_name||' is shuting down!!', |
| message => 'Dear Admin,, |
| please be informed that your database "'||sys.database\_name|| |
| '" is shutting down on '||SYSDATE||' due to ('||sys.sysevent||'). ', |
| mime\_type => 'text/plain; charset=us-ascii' |
| ); |
| END; |
| / |

****

**Please note:** the trigger does two things: save alert log in "SYSTEM\_ALERTS" table and then send email to the manager to inform him about the event.

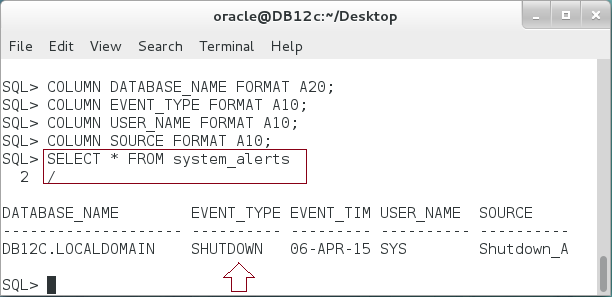
**Step 8:** We generate a SHUTDOWN event by restarting database using the following SQL\*Plus commands:

|  |  |
| --- | --- |
| Command | Description |
| SHUTDOWN IMMEDIATE; | Shutdown database |
| STARTUP; | Startup database |

****

**Step 9:** Query "SYSTEM\_ALERTS" table to see if the trigger was fired after shutting the database down as shown:

|  |  |
| --- | --- |
| Command | Description |
| COLUMN DATABASE\_NAME FORMAT A20; |  |
| COLUMN EVENT\_TYPE FORMAT A10; |  |
| COLUMN USER\_NAME FORMAT A10; |  |
| COLUMN SOURCE FORMAT A10; |  |
| **SELECT \* FROM system\_alerts** |  |
| / |  |

****

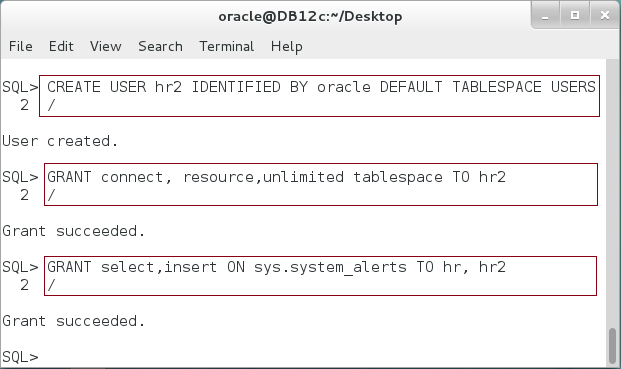
## Schema Trigger

**Step 1:** Oracle supports trigger on schema level in which you specify one or more of the following events:

|  |  |
| --- | --- |
| Command | Description |
| ALTER | * Some events include others. For example, DDL event includes ALTER, CREATE, GRANT, and other events. * All events, except SUSPEND, can be applied with AFTER or BEFORE timing clause. SUSPEND event works with "AFTER" timing only. * All schema events can also work with database level. In this case, Oracle track your selected events for all database schemas, not just one schema. |
| ANALYZE |
| ASSOCIATE STATISTICS |
| AUDIT |
| COMMENT |
| CREATE |
| DDL |
| DISASSOCIATE STATISTICS |
| DROP |
| GRANT |
| NOAUDIT |
| RENAME |
| REVOKE |
| TRUNCATE |
| SUSPEND |  |

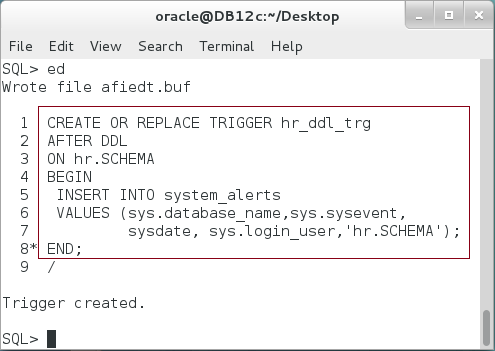
**Step 2:** We are going to start by create a new schema "HR2" and grant it all required privileges as shown below:

|  |
| --- |
| Command |
| CREATE USER **hr2** IDENTIFIED BY oracle DEFAULT TABLESPACE USERS |
| / |
| GRANT connect, resource,unlimited tablespace TO hr2 |
| / |
| GRANT select,insert ON sys.system\_alerts TO hr, hr2 |
| / |



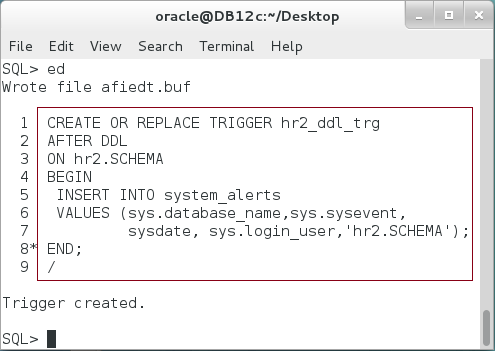
**Step 3:** Create a schema trigger to track DDL statements on HR objects as shown below:

|  |  |
| --- | --- |
| Command | Description |
| **CREATE** OR REPLACE **TRIGGER** hr\_ddl\_trg | Create trigger |
| **AFTER DDL** | Event: After DDL |
| **ON hr.SCHEMA** | Item: all "HR" objects. |
| BEGIN |  |
| INSERT INTO system\_alerts | Save a record for each DDL statement done on HR schema. |
| VALUES (sys.database\_name,sys.sysevent, |
| sysdate, sys.login\_user,**'hr.SCHEMA'**); |
| END; |  |
| / |  |

****

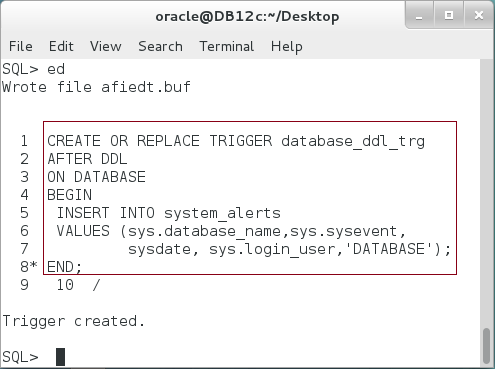
**Step 4:** Create another trigger but to track changes done in HR2 schema this time.

|  |  |
| --- | --- |
| Command | Description |
| **CREATE** OR REPLACE **TRIGGER** hr2\_ddl\_trg | Create trigger |
| **AFTER DDL** | Event: After DDL |
| **ON hr2.SCHEMA** | Item: all "HR2" objects. |
| BEGIN |  |
| INSERT INTO system\_alerts | Save a record for each DDL statement done on HR2 schema. |
| VALUES (sys.database\_name,sys.sysevent, |
| sysdate, sys.login\_user,**'hr2.SCHEMA'**); |
| END; |  |
| / |  |



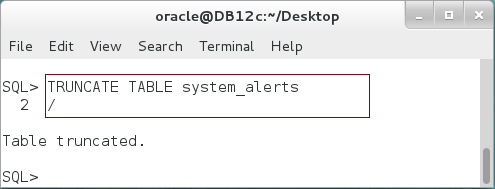
**Step 5:** Finally, we are going a third trigger to track DDL changes but on database level as shown below:

|  |  |
| --- | --- |
| Line |  |
| **CREATE** OR REPLACE **TRIGGER** database\_ddl\_trg | Create trigger |
| **AFTER DDL** | Event: After DDL |
| **ON DATABASE** | Item: all database objects. |
| BEGIN |  |
| INSERT INTO system\_alerts | Save record for each DDL statement done on any database object. |
| VALUES (sys.database\_name,sys.sysevent, |
| sysdate, sys.login\_user,**'DATABASE'**); |
| END; |
| / |  |



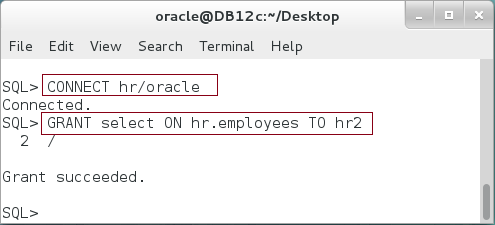
**Step 6:** While connecting to SYS user, execute the first DDL statement as shown:

|  |  |
| --- | --- |
| Line |  |
| TRUNCATE TABLE system\_alerts |  |
| / |  |



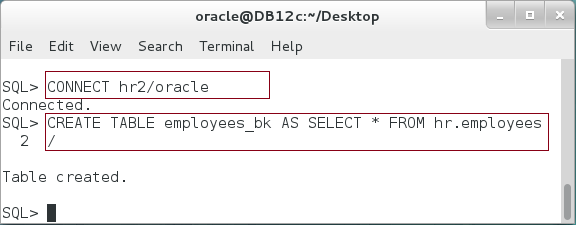
**Step 7:** Connect to "HR" and execute this DDL statement:

|  |  |
| --- | --- |
| Line | Description |
| CONNECT hr/oracle | login as HR |
| GRANT select ON hr.employees TO hr2 |  |
| / |  |



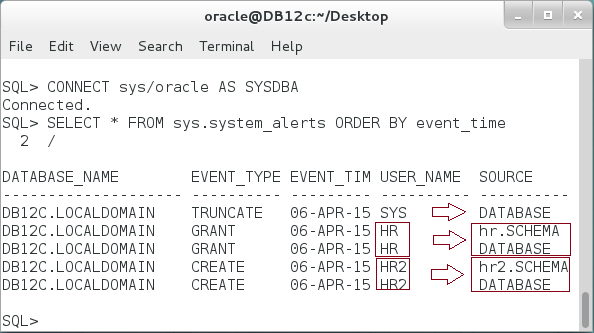
**Step 8:** Finally, logon as "HR2" and execute the following DDL statement:

|  |  |
| --- | --- |
| Line |  |
| CONNECT hr2/oracle |  |
| CREATE TABLE employees\_bk AS SELECT \* FROM hr.employees |  |
| / |  |



**Step 9:** Now, logon as "SYS" again and query "SYSTEM\_ALERT" table as shown:

|  |  |
| --- | --- |
| Line |  |
| CONNECT sys/oracle AS SYSDBA |  |
| SELECT \* FROM sys.system\_alerts ORDER BY event\_time |  |
| / |  |

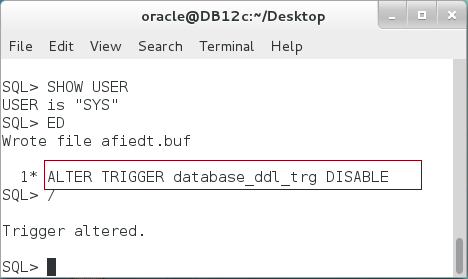


**Explain the output.**

## Enable or Disable Trigger

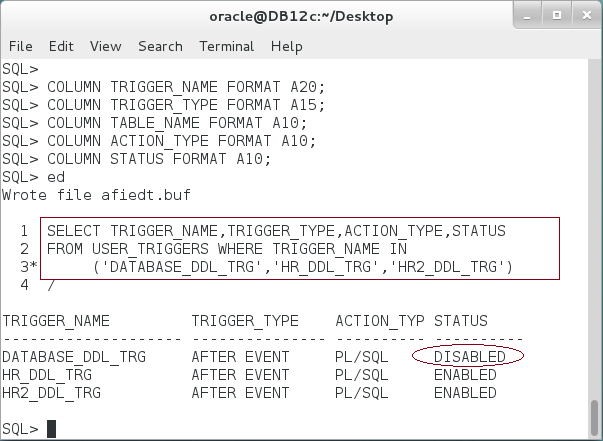
**Step 1:** Use the DISABLE command to temporarily stop firing a trigger. Execute the following command to disable "database\_ddl\_trg" while connecting to "SYS" user:

|  |  |
| --- | --- |
| Line | Description |
| ALTER TRIGGER database\_ddl\_trg **DISABLE** |  |
| / |  |



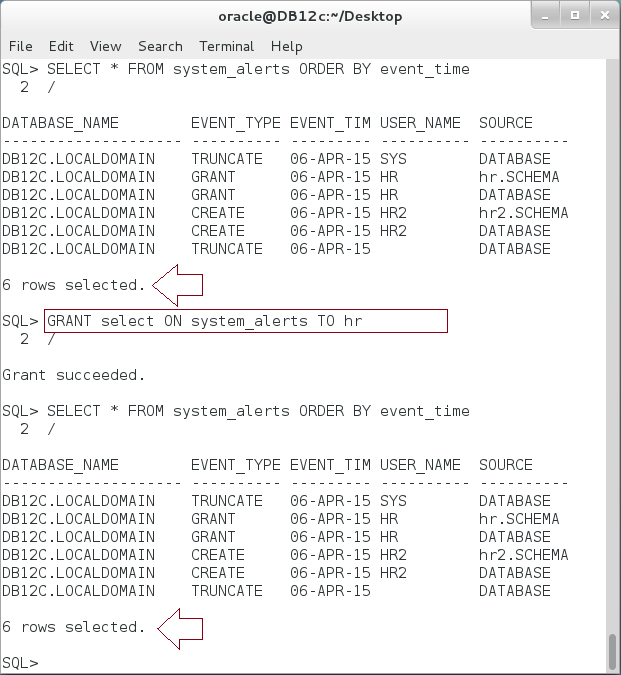
**Step 2:** To view your own triggers and their statuses, use the following command:

|  |  |
| --- | --- |
| Line | Description |
| COLUMN TRIGGER\_NAME FORMAT A20; |  |
| COLUMN TRIGGER\_TYPE FORMAT A15; |  |
| COLUMN TABLE\_NAME FORMAT A10; |  |
| COLUMN ACTION\_TYPE FORMAT A10; |  |
| COLUMN STATUS FORMAT A10; |  |
|  |  |
| **SELECT** TRIGGER\_NAME,TRIGGER\_TYPE,ACTION\_TYPE,STATUS | Query "USER\_TRIGGERS" view. |
| **FROM** USER\_TRIGGERS WHERE TRIGGER\_NAME IN |
| ('DATABASE\_DDL\_TRG','HR\_DDL\_TRG','HR2\_DDL\_TRG') |
| / |  |

****

**Step 3:** Disabled trigger is no longer executed. Execute new DDL statement and view if there is any change occurring in "SYSTEM\_ALERTS" table as shown below:

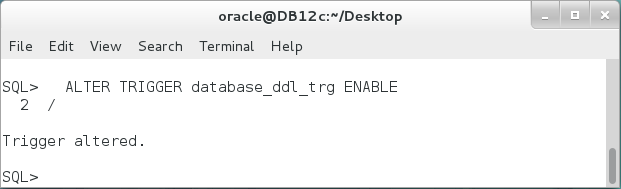
|  |  |
| --- | --- |
| Line | Description |
| SELECT \* FROM system\_alerts ORDER BY event\_time |  |
| / |  |
| GRANT select ON system\_alerts TO hr |  |
| / |  |
| SELECT \* FROM system\_alerts ORDER BY event\_time |  |
| / |  |

****

**Please note:** The number of rows remains the same before and after executing a DDL statement, GRANT.

**Step 4:** You can also enable the trigger again using the following command:

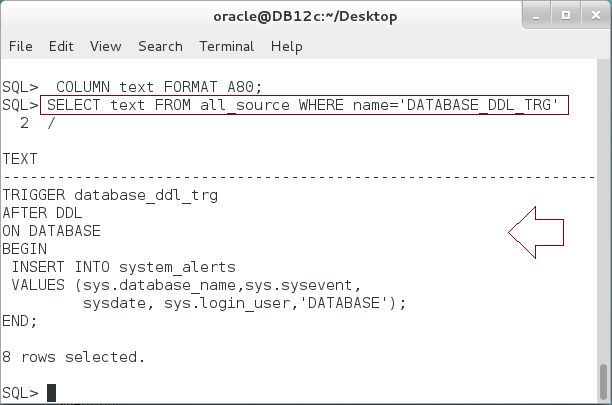
|  |
| --- |
| Line |
| **ALTER TRIGGER** database\_ddl\_trg **ENABLE** |
| / |



## Query, Change, and Compile Trigger

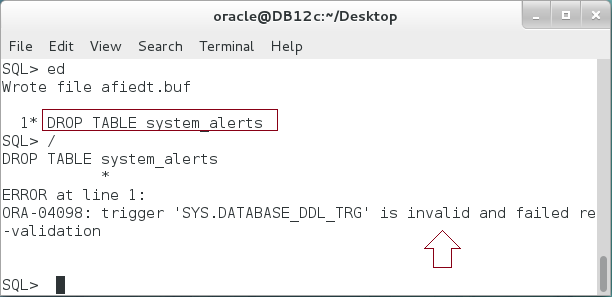
**Step 1:** View the text of Trigger using the following query:

|  |  |
| --- | --- |
| Line | Description |
| COLUMN text FORMAT A80; |  |
| **SELECT** **text** **FROM all\_source** WHERE name='DATABASE\_DDL\_TRG' |  |
| / |  |



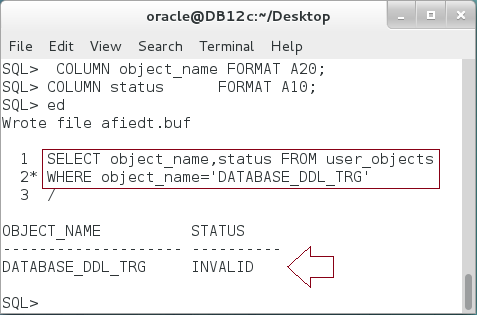
**Step 2:** We are going to make "DATABASE\_DDL\_TRG" invalid by trying to drop "SYSTEM\_ALERTS" table. Execute the following DDL command:

|  |  |
| --- | --- |
| Line | Description |
| DROP TABLE system\_alerts |  |
| / |  |



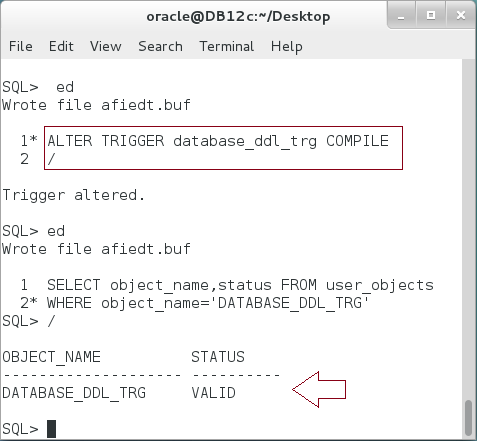
**Step 3:** View the status of the trigger:

|  |  |
| --- | --- |
| Line | Description |
| COLUMN object\_name FORMAT A20; |  |
| COLUMN status FORMAT A10; |  |
| **SELECT** object\_name,status **FROM** **user\_objects** |  |
| WHERE object\_name='DATABASE\_DDL\_TRG' |  |

****

**Step 4:** Compile the invalid triggers using the following command:

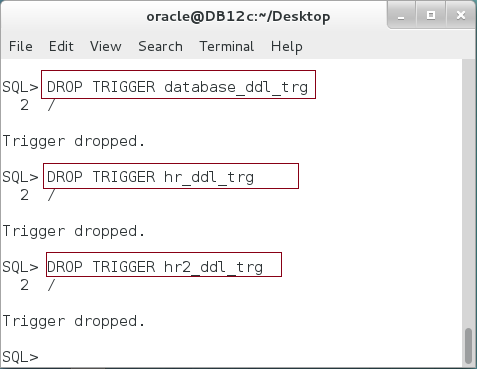
|  |  |
| --- | --- |
| Line | Description |
| ALTER TRIGGER database\_ddl\_trg COMPILE |  |
| / |  |
| SELECT object\_name,status FROM user\_objects |  |
| WHERE object\_name='DATABASE\_DDL\_TRG' |  |
| / |  |

****

## Drop Trigger

**Step 1:** Drop triggers using the following simple commands:

|  |  |
| --- | --- |
| Line | Description |
| **DROP** **TRIGGER** database\_ddl\_trg |  |
| / |  |
| **DROP** **TRIGGER** hr\_ddl\_trg |  |
| / |  |
| **DROP** **TRIGGER** hr2\_ddl\_trg |  |
| / |  |

****

# SUMMARY

Triggers and constraints can enforce business rules. Constraints are simpler and easier to write. They also applied for current table data as well as to new inserted data. However, they are limited and can't apply all business constraints. For that reasons, triggers are coming up. Triggers can handle complex business rules because you are able to write any code you want to enforce your company's rules. A trigger has three main things you should take care of: (1) level or item, (2) event, and (3) trigger body. There are three item categories of levels: Database, Schema and Schema Objects. For each item or level, there are some of events that cause the trigger to be fired. For example, Database trigger may be fired due to SHUTDOWN, STARTUP, LOGON and LOGOUT. Triggers can be disabled and enabled using simple ALTER command. Moreover, triggers have status which may become invalid due to some changes on the underlying objects. You can compile and recreate the trigger in such cases.

After completing this lab exercise, you should be able to build your own database and schema level trigger.

# REFERENCES

* http://www.java2s.com/Tutorial/Oracle/0560\_\_Trigger/DatabaseleveltriggerBEFORESHUTDOWNONDATABASE.htm
* https://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11\_QUESTION\_ID:49818662859946
* http://dba.stackexchange.com/questions/29252/what-is-the-difference-between-on-schema-and-on-database-triggers

# INDEX

trigger 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 20, 22, 26