TRIGGERS

- Triggers are stored procedures that are ran automatically by the database whenever some event happens.

PL/SQL Triggers

Triggers are basically PL/SQL procedures that are associated with tables, and they are called whenever a modification or event occurs. The modification statements may include

INSERT
UPDATE and
DELETE.

The general structure of triggers is:

```
CREATE [OR REPLACE]
TRIGGER trigger_name
BEFORE (or AFTER)
INSERT OR UPDATE [OF COLUMNS] OR DELETE
ON table_name
[FOR EACH ROW [WHEN (condition)]]
BEGIN
...
END;
```

Explanation:

Create or replace → For creating a trigger.
Before or After → Running the trigger before or after the modification.
Insert, update, delete → Statement that triggers the trigger.
For each row → Types of trigger.

Note:

For creating triggers we need permission to the username.

By dinesh

Creating Triggers

Insert Triggers:

BEFORE INSERT Trigger

AFTER INSERT Trigger

Update Triggers:

BEFORE UPDATE Trigger

AFTER UPDATE Trigger

Delete Triggers:

BEFORE DELETE Trigger

AFTER DELETE Trigger

Drop Triggers:

Drop a Trigger

Disable/Enable Triggers:

Disable a Trigger

Disable all Triggers on a table

Enable a Trigger

Enable all Triggers on a table

By dinesh

BEFORE TRIGGER(Insert)

A BEFORE INSERT Trigger means that Oracle will fire this trigger before the INSERT operation is executed.

Restrictions:

- You can not create a BEFORE trigger on a view.
- You can update the :NEW values.
- You can not update the :OLD values.

Table Used.

Column Name	Col ID	Pk	Data Type	Null?
UNAME	1		VARCHAR2 (30)	Y
UDATE	2		DATE	Y

Output:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

Frequency of polling 5 seconds

```
before insert ofaaa
before insert ofbbb
before insert ofccc
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> select * from usr;

UNAME                          UDATE
------------------------------ ---------
dinesh                         17-JUN-08
aaa                            15-JUL-08
anandakuar.m                   30-JUN-08

SQL> ed
Wrote file afiedt.buf

  1  create or replace trigger t1
  2  before insert
  3  on usr
  4  for each row
  5  begin
  6        dbms_output.put_line('before insert of' || :new.uname);
  7* end;
SQL> /

Trigger created.

SQL> insert into usr values('shovan','15-jul-2008');
before insert ofshovan

1 row created.

SQL>
```

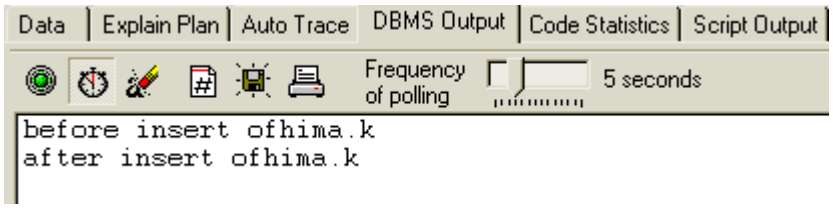
AFTER TRIGGER

```
create or replace trigger t2
after insert
on usr
for each row
begin
     dbms_output.put_line('after insert of' || :new.uname);
end;
```

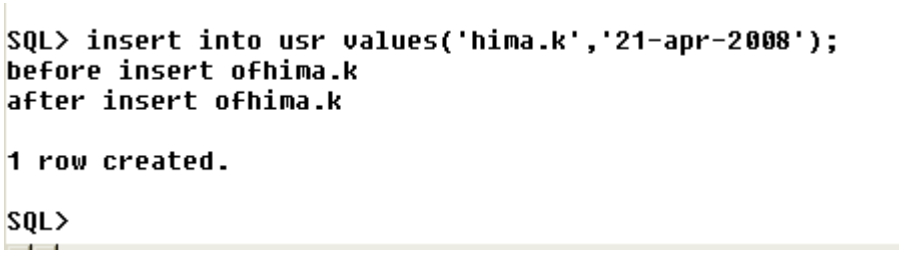
Now inserting the record.

```
insert into usr values('hima.k','21-apr-2008');
```

Output:



Output in SQL Plus:



By dinesh

BEFORE TRIGGER(Update)

```
create or replace trigger t3
before update
on usr
for each row

begin
       dbms_output.put_line('Warning : before update trigger --> date
       changes');
end;
```

Now creating a query to update all the date in the table usr.

```
update usr set udate = sysdate
```

Output

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

Frequency of polling 5 seconds

```
Warning : before update trigger --> date changes
Warning : before update trigger --> date changes
Warning : before update trigger --> date changes
Warning : before update trigger --> date changes
Warning : before update trigger --> date changes
Warning : before update trigger --> date changes
```

All records are update to sysdate.

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

UNAME	UDATE
dinesh	6/18/2008 3:42:49 PM
aaa	6/18/2008 3:42:49 PM
bbb	6/18/2008 3:42:49 PM
ccc	6/18/2008 3:42:49 PM
anandakuar.m	6/18/2008 3:42:49 PM
shovan	6/18/2008 3:42:49 PM

By dinesh

BEFORE & AFTER TRIGGER (Delete)

Before:

```
create or replace trigger t4
before delete
on usr

begin
        dbms_output.put_line('Warning : before delete trigger --> data
going to be lost');
end;
```

After:

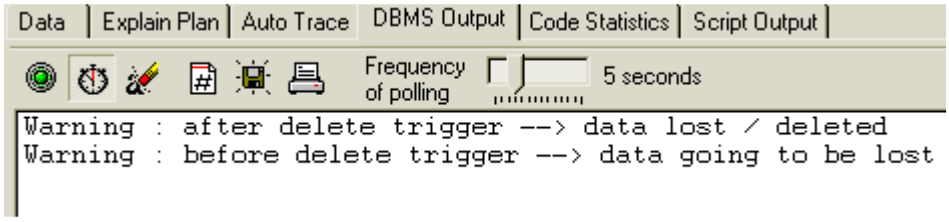
```
create or replace trigger t5
before delete
on usr

begin
        dbms_output.put_line('Warning : after delete trigger --> data
lost / deleted');
end;
```

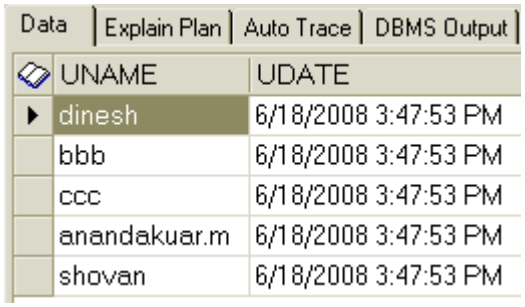
Now write query as follows.

```
delete from usr where uname = 'aaa'
```

Output:



Record successfully deleted.



UNAME	UDATE
dinesh	6/18/2008 3:47:53 PM
bbb	6/18/2008 3:47:53 PM
ccc	6/18/2008 3:47:53 PM
anandakuar.m	6/18/2008 3:47:53 PM
shovan	6/18/2008 3:47:53 PM

By dinesh

SPECIAL IF Statements

```
create or replace trigger t_all
before insert or update or delete on usr1
for each row
begin
if inserting then
dbms_output.put_line('inserting person: ' );
elsif updating then
dbms_output.put_line('updating person: ' );
elsif deleting then
dbms_output.put_line('deleting person: ' );
end if;
end;
```

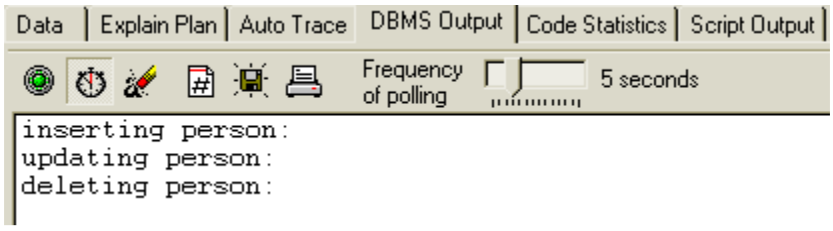
Execute the below queries.

```
insert into usr1 values('mani','14-feb-2008')

update usr1 set uname = 'mani.s' where uname='mani'

delete from usr1 where uname ='mani.s'
```

Output:



By dinesh

Working with VIEWS

- A view is a predefined query on one or more tables.
- Retrieving information from a view is done in the same manner as retrieving from a table.
- With some views you can also perform DML operations (delete, insert, update) on the base tables.
- Views don't store data, they only access rows in the base tables.
- user_tables, user_sequences, and user_indexes are all views.
- View Only allows a user to retrieve data.
- view can hide the underlying base tables.
- By writing complex queries as a view, we can hide complexity from an end user.
- View only allows a user to access certain rows in the base tables.

Create a view:

```
create or replace view v1 as
select * from shipment2;
```

See the data's in view:

```
select * from v1
```

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

SHIPMENT_ID	CUST_ID	WEIGHT	TRUCK_ID	DESTINATION	SHIP_DATE
100	100	500	100	london	
101	101	100	102	paris	
102	101	300	103	london	
103	101	10	102	panamacity	12/12/2003
104	101	20	101	losangles	
105	102	200	102	rome	
106	100	50	101	siouxcity	9/18/2003
107	104	500	100	manhattan	

By dinesh

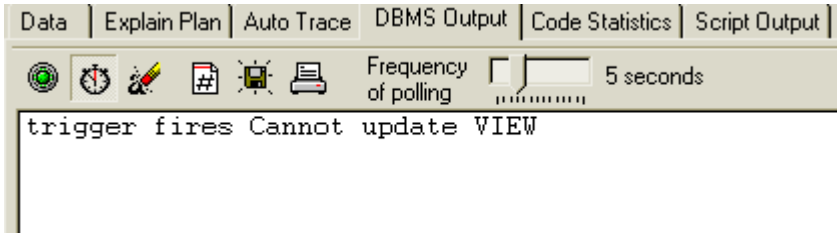
Creating a Trigger:

```
create or replace trigger v_t1
instead of insert on v1
for each row
begin
      dbms_output.put_line('trigger fired Cannot update VIEW');
end;
```

Inserting the value into VIEW:

```
insert into v1(shipment_id) values(156)
```

Output:



Note:

When we try to insert a value into the view, it raises a trigger that we cannot insert anything into the view.

By dinesh

TRIGGER EXCEPTION

Sometimes, an error (or exception) is raised for a good reason.

For example, to prevent some action that improperly modifies the database.

```
CREATE OR REPLACE
TRIGGER t11
BEFORE UPDATE OF udate ON usr1
FOR EACH ROW
BEGIN
            RAISE_APPLICATION_ERROR(-20000,'CANNOT CHANGE DATE');
END;
```

Note:

The next thing you should notice is the procedure call RAISE APPLICATION ERROR, which accepts an error code, and an explanation string. This effectively halts our trigger execution, and raises an error, preventing our UDATE from being modified.

VIEWING TRIGGERS

```
select trigger_name from user_triggers
```

Output:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

TRIGGER_NAME
AUDIT_TRIAL
EMP_BSL_TR
MANAGER_INFO_INSERT
ORDERS_BEFORE_INSERT
T1
T10
T11
T2
T3
T4
T5
T6
TEST_BANK_TRIG
T_ALL
VB_CV2
V_T1

By dinesh

DROPPING TRIGGER

See the above output. Now I am going to delete the trigger **T_ALL.**

```
drop trigger t_all
```

Output:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

TRIGGER_NAME
AUDIT_TRIAL
EMP_BSL_TR
MANAGER_INFO_INSERT
ORDERS_BEFORE_INSERT
T1
T10
T11
T2
T3
T4
T5
T6
TEST_BANK_TRIG
VB_CV2
V_T1

By dinesh

ALTERING TRIGGER

If a trigger seems to be getting in the way, and you don't want to drop it, just disable it for a little while, you can alter it to disable it. Note that this is not the same as dropping a trigger; after you drop a trigger, it is gone.

The general format of an alter would be something like this:

ALTER TRIGGER trigger_name [ENABLE|DISABLE];

Consider Trigger

```
create or replace trigger t12
before insert
on usr3
for each row
begin
     dbms_output.put_line('before insert of' || :new.uname);
end;
```

Disable a Trigger:

```
alter trigger t12 DISABLE
```

Output :

Data | Explain Plan | Auto Trace | DBMS Output

UNAME	UDATE
dinesh	6/18/2008 3:47:53 PM
bbb	6/18/2008 3:47:53 PM
ccc	6/18/2008 3:47:53 PM
anandakuar.m	6/18/2008 3:47:53 PM
shovan	6/18/2008 3:47:53 PM
hima.k	4/21/2008
rahul	1/1/1008
xxx	12/10/2009

After disabling the trigger, the trigger event is not activated.

By dinesh

Disable all triggers on Table:

```
alter table usr3 DISABLE ALL TRIGGERS
```

Enable a Trigger:

```
alter trigger t12 ENABLE
```

Enable all triggers on Table:

```
alter table usr3 ENABLE ALL TRIGGERS
```

By dinesh