# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

### Experiment 8

###### Aim: Write a PL/SQL block to create Triggers for update, delete and insert Resources required: P-IV and above, Oracle

**Theory:**

**View**

A **view** is a virtual table, which provides access to a subset of column from one or more table. A **view** can derive its data from one or more table. An output of query can be stored as a **view**.

... A **view in oracle** is nothing but a stored sql scripts.

###### create view v11 as (select empid, ename ,dname from emp,dept where emp.dno=dept.dno);

Views may be created for the following reasons:

* 1. The DBA stores the views as a definition only. Hence there is no duplication of data.
  2. Simplifies Queries.
  3. Can be Queried as a base table itself.
  4. Provides data security.
  5. Avoids data redundancy. **Creation of Views:- Syntax:-**

CREATE VIEW viewname AS SELECT columnname,columnname FROM tablename

WHERE columnname=expression\_list; **Renaming the columns of a view:- Syntax:-**

CREATE VIEW viewname AS SELECT newcolumnname…. FROM tablename

WHERE columnname=expression\_list;

###### Selecting a data set from a view-

**SɅRɅSWɅTI**

**Department of Computer Engineering**

**College of Engineering**

SɅRɅSWɅTI

**Syntax:-**

SELECT columnname, columnname FROM viewname

WHERE search condition; **Destroying a view- Syntax:-**

DROP VIEW viewname;

**Triggers**

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

* A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
* A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
* A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

#### Benefits of Triggers

Triggers can be written for the following purposes −

* Generating some derived column values automatically
* Enforcing referential integrity
* Event logging and storing information on table access
* Auditing
* Synchronous replication of tables
* Imposing security authorizations
* Preventing invalid transactions

###### The syntax for creating a trigger is −

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

ON table\_name

[REFERENCING OLD AS o NEW AS n] [FOR EACH ROW]

WHEN (condition) DECLARE

Declaration-statements BEGIN

Executable-statements EXCEPTION

Exception-handling-statements END;

Where,

* CREATE [OR REPLACE] TRIGGER trigger\_name − Creates or replaces an existing trigger with the *trigger\_name*.
* {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
* {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
* [OF col\_name] − This specifies the column name that will be updated.
* [ON table\_name] − This specifies the name of the table associated with the trigger.
* [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
* [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
* WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

SQL>create table customer(id number(2), name varchar2(15), age number(2), address varchar2(20), salary number(9,2));

# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

SQL>create table customer\_log(id number(2), name varchar2(15), age number(2), address varchar2(20), salary number(9,2), usern varchar2(10), dt date, oper varchar2(10));

create or replace trigger cust\_log

after update or delete or insert on customer1 for each row

declare

op varchar2(10); begin

if updating then op:='update';

insert into customer1\_log(id,name,age,address,salary,dt,oper) values(:old.id, :old.name,

:old.age, :old.address, :old.salary,current\_timestamp, op); end if;

if deleting then op:='delete';

insert into customer1\_log(id,name,age,address,salary,dt,oper) values(:old.id, :old.name,

:old.age, :old.address, :old.salary,current\_timestamp, op); end if;

if inserting then op:='insert';

insert into customer1\_log(id,name,age,address,salary,dt,oper) values(:new.id, :new.name,

:new.age, :new.address, :new.salary,current\_timestamp, op); end if;

end;

SQL> insert into customer1(id,salary) values(9,10000); 1 row created.

# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

###### Conclusion: We have successfully Implemented Views and Triggers in MySQL Command Line.

###### Code and Output:

###### Views:

###### 

###### 

###### Triggers:

###### 

###### 