NAME: JUNAID GIRKAR  
SAP ID: 60004190057

BATCH: SE A-3 Computer

**Experiment No-8**

**Aim:** Create Views and Triggers.

**Theory:** A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depend on the written SQL query to create a view. Views, which are kind of virtual tables, allow users to do the following:

* Structure data in a way that users or classes of users find natural or intuitive.
* Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
* Summarize data from various tables which can be used to generate reports.

**Creating Views:**

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables, or another view. To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic CREATE VIEW syntax is as follows:

**CREATE VIEW view\_name AS**

**SELECT column1, column2.....**

**FROM table\_name**

**WHERE [condition];**

1. **Create a view by fetching few columns from a table with where condition**

create view first as select \* from user WHERE user\_id < 3;

select \* from first;

**+---------+---------------+------------------------+------------------------+**

**| user\_id | Name          | Email                  | Address                |**

**+---------+---------------+------------------------+------------------------+**

**|       1 | Junaid Girkar |**[**junaidgirkar@gmail.com**](mailto:junaidgirkar@gmail.com)**| 221 Bakers Street      |**

**|       2 | Harry Potter  |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**| 4 Private Drive, Surry |**

**+---------+---------------+------------------------+------------------------+**

1. **Create a view by applying aggregate function with group by clause on a single table**

create view second as select avg(user\_id), Count(user\_id), Name, Email from user group by Name;

select \* from second;

**+--------------+----------------+------------------+------------------------+**

**| avg(user\_id) | Count(user\_id) | Name             | Email                  |**

**+--------------+----------------+------------------+------------------------+**

**|       1.0000 |              1 | Junaid Girkar    |**[**junaidgirkar@gmail.com**](mailto:junaidgirkar@gmail.com)**|**

**|       2.0000 |              1 | Harry Potter     |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**|**

**|       3.0000 |              1 | Percy Jackson    |**[**percyjackson@gmail.com**](mailto:percyjackson@gmail.com)**|**

**|       4.0000 |              1 | Severus Snape    |**[**snape@snapey.com**](mailto:snape@snapey.com)**|**

**|       5.0000 |              1 | Albus Dumbledore |**[**APWBD@hogwards.com**](mailto:APWBD@hogwards.com)**|**

**+--------------+----------------+------------------+------------------------+**

1. **Create a view using a subquery**

create view third as select \* from user where user\_id in ( select user\_id from user\_mobile\_no);

select \* from third;

**+---------+---------------+------------------------+------------------------------+**

**| user\_id | Name          | Email                  | Address                      |**

**+---------+---------------+------------------------+------------------------------+**

**|       1 | Junaid Girkar |**[**junaidgirkar@gmail.com**](mailto:junaidgirkar@gmail.com)**| 221 Bakers Street            |**

**|       2 | Harry Potter  |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**| 4 Private Drive, Surry       |**

**|       3 | Percy Jackson |**[**percyjackson@gmail.com**](mailto:percyjackson@gmail.com)**| camp Half Blood, Long Island |**

**+---------+---------------+------------------------+------------------------------+**

1. **Create a view by fetching records from multiple tables using join**

create view fourth as select \* from user\_mobile\_no NATURAL JOIN user;

select \* from fourth;

**+---------+-----------+---------------+------------------------+------------------------------+**

**| user\_id | Mobile\_No | Name          | Email                  | Address                      |**

**+---------+-----------+---------------+------------------------+------------------------------+**

**|       1 |     12345 | Junaid Girkar |**[**junaidgirkar@gmail.com**](mailto:junaidgirkar@gmail.com)**| 221 Bakers Street            |**

**|       2 |     54321 | Harry Potter  |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**| 4 Private Drive, Surry       |**

**|       3 |     98765 | Percy Jackson |**[**percyjackson@gmail.com**](mailto:percyjackson@gmail.com)**| camp Half Blood, Long Island |**

**+---------+-----------+---------------+------------------------+------------------------------+**

1. **Display all views**

ALREADY DONE

1. **Try to perform insert/update/delete**

**INSERT**:

**QUERY: Insert into first values ('0','Lord Voldemort', 'TomRiddle@LV.com','Riddle Manor, Upper Hangleton');**

Select \* from first;

**+---------+----------------+------------------------+-------------------------------+**

**| user\_id | Name           | Email                  | Address                       |**

**+---------+----------------+------------------------+-------------------------------+**

**|       0 | Lord Voldemort |**[**TomRiddle@LV.com**](mailto:TomRiddle@LV.com)**| Riddle Manor, Upper Hangleton |**

**|       1 | Junaid Girkar  |**[**junaidgirkar@gmail.com**](mailto:junaidgirkar@gmail.com)**| 221 Bakers Street             |**

**|       2 | Harry Potter   |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**| 4 Private Drive, Surry        |**

**+---------+----------------+------------------------+-------------------------------+**

**UPDATE**:

**QUERY: update first set email='UpdatedEmail@gmail.com' where user\_id=1;**

Select \* from first;

**+---------+----------------+------------------------+-------------------------------+**

**| user\_id | Name           | Email                  | Address                       |**

**+---------+----------------+------------------------+-------------------------------+**

**|       0 | Lord Voldemort |**[**TomRiddle@LV.com**](mailto:TomRiddle@LV.com)**| Riddle Manor, Upper Hangleton |**

**|       1 | Junaid Girkar  |**[**UpdatedEmail@gmail.com**](mailto:UpdatedEmail@gmail.com)**| 221 Bakers Street             |**

**|       2 | Harry Potter   |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**| 4 Private Drive, Surry        |**

**+---------+----------------+------------------------+-------------------------------+**

**DELETE**:

QUERY: Delete from first where user\_id =0;

Select \* from first;

**+---------+---------------+------------------------+------------------------+**

**| user\_id | Name          | Email                  | Address                |**

**+---------+---------------+------------------------+------------------------+**

**|       1 | Junaid Girkar |**[**UpdatedEmail@gmail.com**](mailto:UpdatedEmail@gmail.com)**| 221 Bakers Street      |**

**|       2 | Harry Potter  |**[**harrypotter@gmail.com**](mailto:harrypotter@gmail.com)**| 4 Private Drive, Surry |**

**+---------+---------------+------------------------+------------------------+**

**TRIGGERS**

A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

A trigger is a named database object that is associated with a table, and it activates when a particular event (e.g. an insert, update or delete) occurs for the table. The statement CREATE TRIGGER creates a new trigger in MySQL.

**Syntax**

CREATE

[DEFINER = { user | CURRENT\_USER }]

TRIGGER trigger\_name

trigger\_time trigger\_event

ON tbl\_name FOR EACH ROW

trigger\_body

trigger\_time: { BEFORE | AFTER }

trigger\_event: { INSERT | UPDATE | DELETE }

**Example**

1) delimiter //

create trigger depocheck

before insert on depositor

FOR EACH ROW

IF NEW.salary is null

THEN

SET NEW.salary = 5000;

END IF;

//

2) delimiter //

create trigger feed\_depositor\_bkp

after insert on depositor

FOR EACH ROW

insert into depositor\_bkp(cust\_name,salary,branch) values

(NEW.cust\_name, new.salary,new.branch);

//

3) delimiter //

create trigger employeetrig

before update on employee

for each row

if new.dno is null then

set new.dno= 01;

end if;

//

4) delimiter //

create trigger total\_sal1

after insert on employee

for each row

if new.dno is not null then

update department

set total\_sal=total\_sal+new.salary

where dnumber=new.dno;

end if;

//

5) drop trigger feed\_depositor\_bkp;

**creating user and Granting privileges to user:**

mysql> create database your\_db\_name;

mysql> grant usage on \*.\* to your\_user@localhost identified by 'your\_user\_password';

mysql> SELECT User FROM mysql.user;

mysql> grant all privileges on your\_db\_name.\* to your\_user@localhost ;

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON your\_db\_name.\*

TO your\_user@localhost ;

mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM your\_user@localhost ;

**MY TRIGGER CODE:**

1. Before updating one table a trigger should fire to insert record in other table

ANS:

CREATE TABLE system\_logs (

    -> id INT AUTO\_INCREMENT PRIMARY KEY,

    -> user\_id INT NOT NULL,

    -> Name VARCHAR(255) NOT NULL,

    -> changedate DATETIME DEFAULT NULL,

    -> action VARCHAR(50) DEFAULT NULL

    -> );

CREATE TRIGGER before\_user\_update

    -> BEFORE UPDATE ON user

    -> FOR EACH ROW

    -> INSERT INTO system\_logs

    -> SET action = 'update',

    -> user\_id = OLD.user\_id,

    -> Name = OLD.Name,

    -> changedate = NOW()

    -> ;

UPDATE user

    -> SET Name = 'APWB Dumbledore'

    -> WHERE

    -> user\_id = 5

    -> ;

**select \* from system\_logs;**

+----+---------+------------------+---------------------+--------+

| id | user\_id | Name             | changedate          | action |

+----+---------+------------------+---------------------+--------+

|  1 |       5 | Albus Dumbledore | 2020-11-03 13:29:49 | update |

+----+---------+------------------+---------------------+--------+

1 row in set (0.00 sec)

1. After Updating record in one table if a specific new value is equal to old value fire a trigger to send a message\_text

create trigger same\_values

    -> after update on user

    -> for each row

    -> begin

    -> if new.Name = old.Name

    -> then signal sqlstate '45000' set message\_text = 'Same Name !';

    -> end if;

    -> end $$

UPDATE user SET Name = 'Junaid Girkar' where Email = 'UpdatedEmail@gmail.com'$$

ERROR: 1644: Same Name!

1. before inserting a record in a table check if a column is null ,insert some default value through trigger

ANS:

CREATE TRIGGER null\_selector BEFORE INSERT ON user FOR EACH ROW IF NEW.Email is NULL THEN SET NEW.Email ='Default@email.com'; END IF; $$

INSERT INTO user (user\_id, Name,Address) VALUES (9, 'Ariana Dumbledore', 'Godric Hallow')

                                          -> $$

**select \* from user $$**

+---------+--------------------+---------------------------+------------------------+

| user\_id | Name               | Email                     | Address                |

+---------+--------------------+---------------------------+------------------------+

|       1 | Junaid Girkar      | [junaidgirkar@gmail.com](mailto:junaidgirkar@gmail.com)    | Goregaon               |

|       2 | Harry Potter       | [harrypotter@gmail.com](mailto:harrypotter@gmail.com)     | 4 Private Drive, Surry |

|       3 | Hermoine Granger   | [hermoinegranger@gmail.com](mailto:hermoinegranger@gmail.com) | London                 |

|       4 | Ron Weasley        | [ronweasley@gmail.com](mailto:ronweasley@gmail.com)      | The Burrow             |

|       5 | Luna Lovegood      | [luna@gmail.com](mailto:luna@gmail.com)            | The Rookery            |

|       6 | Neville Longbottom | [neville@gmail.com](mailto:neville@gmail.com)         | Longbottom Manor       |

|       7 | Lord Voldemort     | [tomriddle@gmail.com](mailto:tomriddle@gmail.com)       | Riddle Manor           |

|       8 | APWB Dumbledore    | [albusdumbledore@gmail.com](mailto:albusdumbledore@gmail.com) | Hogwarts               |

|       9 | Ariana Dumbledore  | [Default@email.com](mailto:Default@email.com)         | Godric Hallow          |

+---------+--------------------+---------------------------+------------------------+

1. **After insert in a table ,create a backup table through trigger**

create table backup\_user

-> (

-> user\_id int not null,

-> Name varchar(60),

-> Email varchar(50),

-> Address varchar(100)

-> );

create trigger user\_backup

-> after insert on user

-> for each row

-> insert into backup\_user values (new.user\_id, new.Name, new.Email, new.Address);

INSERT INTO user (user\_id, Name, Email, Address)

-> VALUES

-> (9, 'Ariana Dumbledore', 'ariana@gmail.com', 'Godric Hollow');

**select \* from backup\_user;**

**+---------+-------------------+------------------+---------------+**

**| user\_id | Name | Email | Address |**

**+---------+-------------------+------------------+---------------+**

**| 9 | Ariana Dumbledore | ariana@gmail.com | Godric Hollow |**

**+---------+-------------------+------------------+---------------+**

1. **before deletion archive data in another table**

create table before\_deletion\_records

-> (user\_id int, Name varchar(30), Email varchar(50), Address varchar(100));

Query OK, 0 rows affected (0.06 sec)

create trigger previous\_records

-> before delete on user

-> for each row

-> insert into before\_deletion\_records values (old.user\_id, old.Name, old.Email, old.Address);

delete from user where user\_id=4;

select \* from before\_deletion\_records;

+---------+---------------+----------------------+------------+

| user\_id | Name | Email | Address |

+---------+---------------+----------------------+------------+

| 4 | Severus Snape | snape@snapey.com | Hogwards |

| 6 | Ron Weasley | ronweasley@gmail.com | The Burrow |

+---------+---------------+----------------------+------------+

1. **After deletion archive data in another table**

create table after\_deletion\_records(user\_id int,name varchar(30),email varchar(50),address varchar(100));

create trigger deleted\_records

-> AFTER DELETE ON user

-> for each row

-> insert into after\_deletion\_records values

-> (OLD.user\_id, OLD.name, old.Email, OLD.Address);

delete from user where user\_id = 6

-> ;

select \* from after\_deletion\_records;

+---------+-------------+----------------------+------------+

| user\_id | name | email | address |

+---------+-------------+----------------------+------------+

| 6 | Ron Weasley | ronweasley@gmail.com | The Burrow |

+---------+-------------+----------------------+------------+

**CONCLUSION**: We have learnt the use of Views and Triggers and their applications in Relational Database Management Systems.