

AIM

Create a Trigger for employee table it will update another table salary while updating values

OBJECTIVE

To develop and execute a Trigger for After update/Delete/Insert operations on a table

PROCEDURE

step 1: start

step 2: initialize the trigger.

step 3: On update the trigger has to be executed.

step 4: execute the trigger procedure after updation

step 5: carryout the operation on the table to check for trigger execution.

step 6: stop

PROGRAM

Sql>

```
CREATE TABLE `employee` (  
  `emp_id` int(11) NOT NULL,  
  `emp_name` varchar(45) DEFAULT NULL,  
  `dob` date DEFAULT NULL,  
  `address` varchar(45) DEFAULT NULL,  
  `designation` varchar(45) DEFAULT NULL,  
  `mobile_no` int(11) DEFAULT NULL,  
  `dept_no` int(11) DEFAULT NULL,  
  `salary` int(11) DEFAULT NULL,  
  PRIMARY KEY (`emp_id`)  
);
```

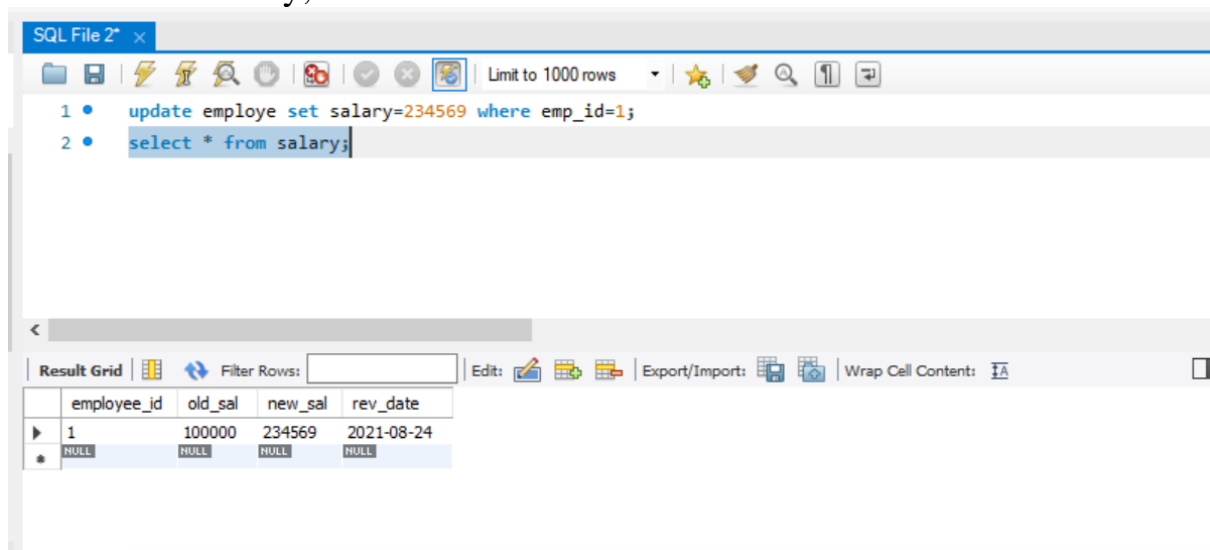
Sql>

```
CREATE TABLE `salary` (  
  `employee_id` int(11) NOT NULL,  
  `old_sal` int(11) DEFAULT NULL,  
  `new_sal` int(11) DEFAULT NULL,  
  `rev_date` date DEFAULT NULL,  
  PRIMARY KEY (`employee_id`)  
);
```

Sql>

```
CREATE DEFINER=`root`@`localhost` TRIGGER  
`employee_db`.`employee_AFTER_UPDATE` AFTER UPDATE ON `employee`  
FOR EACH ROW
```

```
BEGIN
if(new.salary != old.salary)
  then
INSERT INTO salary (employee_id,old_sal,new_sal,rev_date) values
(new.emp_id,old.salary,new.salary,sysdate());
END if;
END
Sql>
update employee set salary=234569 where emp_id=1;
select * from salary;
```



The screenshot shows an SQL IDE window titled "SQL File 2". The toolbar includes icons for file operations, execution, and a "Limit to 1000 rows" dropdown. The SQL editor contains two statements:

- 1 • `update employee set salary=234569 where emp_id=1;`
- 2 • `select * from salary;`

Below the editor is a "Result Grid" with a "Filter Rows" input and a "Wrap Cell Content" checkbox. The grid displays the following data:

	employee_id	old_sal	new_sal	rev_date
▶	1	100000	234569	2021-08-24
*	NULL	NULL	NULL	NULL

AIM

Create a Trigger for employee table it will update another table personal_updates while updating values

OBJECTIVE

To develop and execute a Trigger for Before and After update/Delete/Insert operations on a table

PROCEDURE

step 1: start

step 2: initialize the trigger.

step 3: On update the trigger has to be executed.

step 4: execute the trigger procedure after updation

step 5: carryout the operation on the table to check for trigger execution.

step 6: stop

PROGRAM

sql>

```
CREATE TABLE `employee` (  
  `emp_id` int(11) NOT NULL,  
  `emp_name` varchar(45) DEFAULT NULL,  
  `dob` date DEFAULT NULL,  
  `address` varchar(45) DEFAULT NULL,  
  `designation` varchar(45) DEFAULT NULL,  
  `mobile_no` int(11) DEFAULT NULL,  
  `dept_no` int(11) DEFAULT NULL,  
  `salary` int(11) DEFAULT NULL,  
  PRIMARY KEY (`emp_id`)  
);
```

Sql>

```
CREATE TABLE `personal_updates` (  
  `emp_id` int(11) NOT NULL,  
  `old_phoneno` int(11) DEFAULT NULL,  
  `new_phoneno` int(11) DEFAULT NULL,  
  `rev_date` date DEFAULT NULL,  
  PRIMARY KEY (`emp_id`)  
);
```

Sql>

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`employee_AFTER_UPDATE_1` AFTER UPDATE ON `employee` FOR EACH
ROW BEGIN
if(new.mobile_no != old.mobile_no)
then
INSERT INTO personal_updates
(emp_id,old_phoneno,new_phoneno,rev_date) values
(new.emp_id,new.mobile_no,old.mobile_no,sysdate());
END if;
END
```

sql>

```
update employee set mobile_no=34566 where emp_id=4 ;
```

```
select * from personal_updates;
```

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, including 'college_management_db', 'db_student', and 'employee_db'. The 'employee_db' database is selected, showing its tables, views, stored procedures, functions, and triggers. The 'employee' table is highlighted. On the right, the 'SQL File 2' pane shows the SQL code for the trigger and the execution results. The SQL code is as follows:

```
1 • update employee set mobile_no=34566 where emp_id=1;
2 select * from personal_updates;
3
```

The 'Result Grid' pane shows the results of the SQL execution. It contains a table with the following data:

emp_id	old_phoneno	new_phoneno	rev_date
1	34566	755982243	2021-08-24

AIM

Create a Trigger for employe table it will update another table promotions while updating values

OBJECTIVE

To develop and execute a Trigger for Before and After update/Delete/Insert operations on a table

PROCEDURE

step 1: start

step 2: initialize the trigger.

step 3: On update the trigger has to be executed.

step 4: execute the trigger procedure after updation

step 5: carryout the operation on the table to check for trigger execution.

step 6: stop

PROGRAM

sql>

```
CREATE TABLE `employe` (  
  `emp_id` int(11) NOT NULL,  
  `emp_name` varchar(45) DEFAULT NULL,  
  `dob` date DEFAULT NULL,  
  `address` varchar(45) DEFAULT NULL,  
  `designation` varchar(45) DEFAULT NULL,  
  `mobile_no` int(11) DEFAULT NULL,  
  `dept_no` int(11) DEFAULT NULL,  
  `salary` int(11) DEFAULT NULL,  
  PRIMARY KEY (`emp_id`)  
);
```

Sql>

```
CREATE TABLE `promotions` (  
  `emp_id` int(11) NOT NULL,  
  `old_designation` varchar(11) DEFAULT NULL,  
  `new_designation` varchar(11) DEFAULT NULL,  
  `rev_date` date DEFAULT NULL,  
  PRIMARY KEY (`emp_id`)  
);
```

sql>

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`employee_AFTER_UPDATE_2` AFTER UPDATE ON `employee` FOR EACH
ROW BEGIN
if(new.designation != old.designation)
then
INSERT INTO promotions (emp_id,old_designation,new_designation,rev_date)
values (new.emp_id,new.designation,old.designation,sysdate());
END if;
END
```

sql>

```
update employee set designation='clk' where emp_id=1;
```

```
select * from promotions;
```

