DBMS- Lab 6

PL SQL Cursors and Triggers

School of Computer Engineering and technology

Cursors

Cursors

- To handle a result set inside a stored procedure, we use a cursor.
- A cursor allows us to iterate a set of rows returned by a query and process each row accordingly.
- The set of rows the cursor holds is referred to as the **active set**.

1. We can declare a cursor by using the DECLARE statement:

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

- The cursor declaration must be after any variable declaration.
- A cursor must always be associated with a SELECT statement.

Cursors

2. Next, open the cursor by using the OPEN statement.

```
OPEN cursor_name;
```

3. Then, use the FETCH statement to retrieve the next row pointed by the cursor and move the cursor to the next row in the result set.

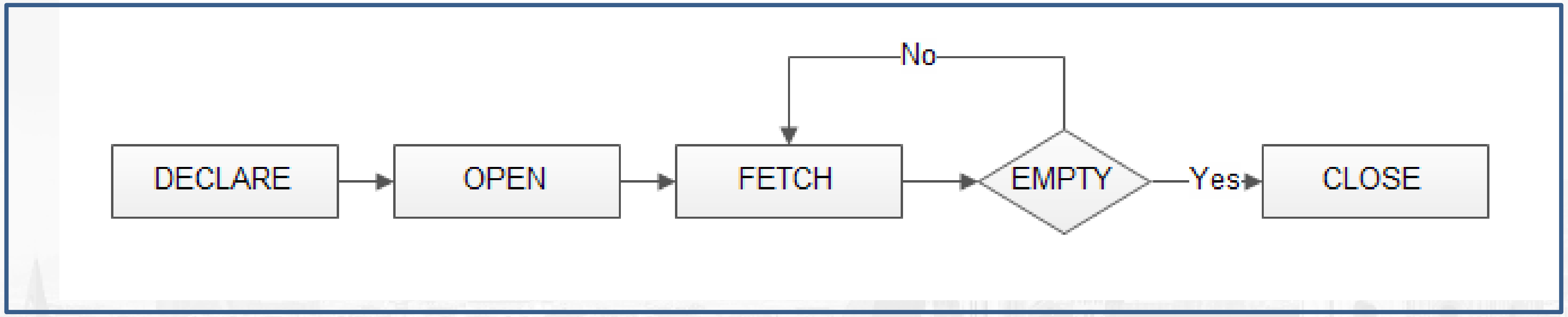
```
FETCH cursor_name INTO variables list;
```

4. Finally, call the CLOSE statement to deactivate the cursor and release the memory associated with it as follows:

```
CLOSE cursor_name;
```

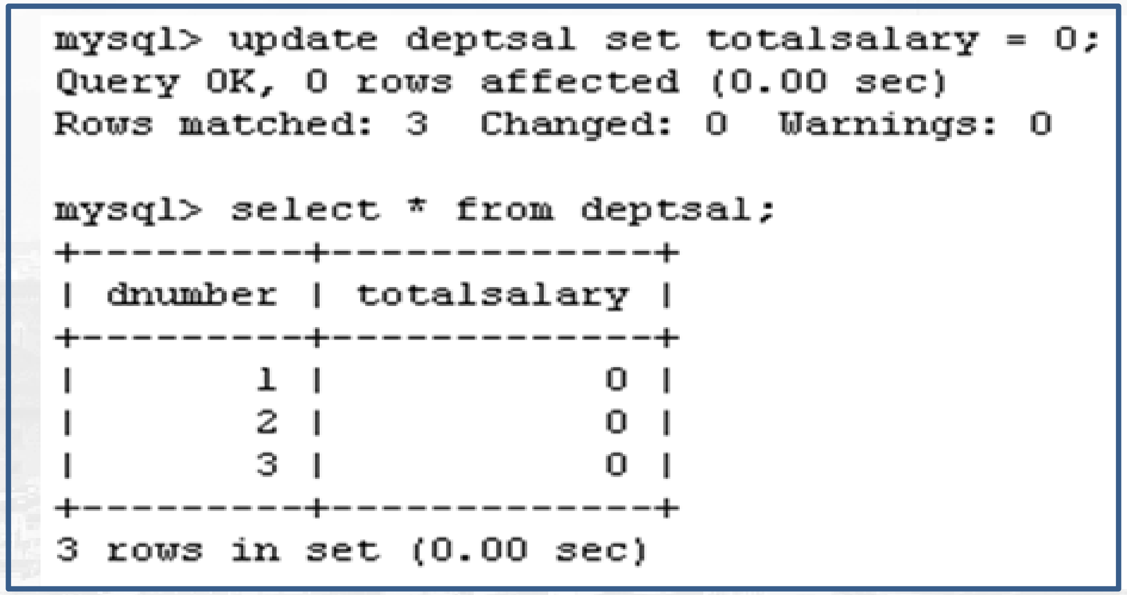
Cursors

The following diagram illustrates how MySQL cursor works.



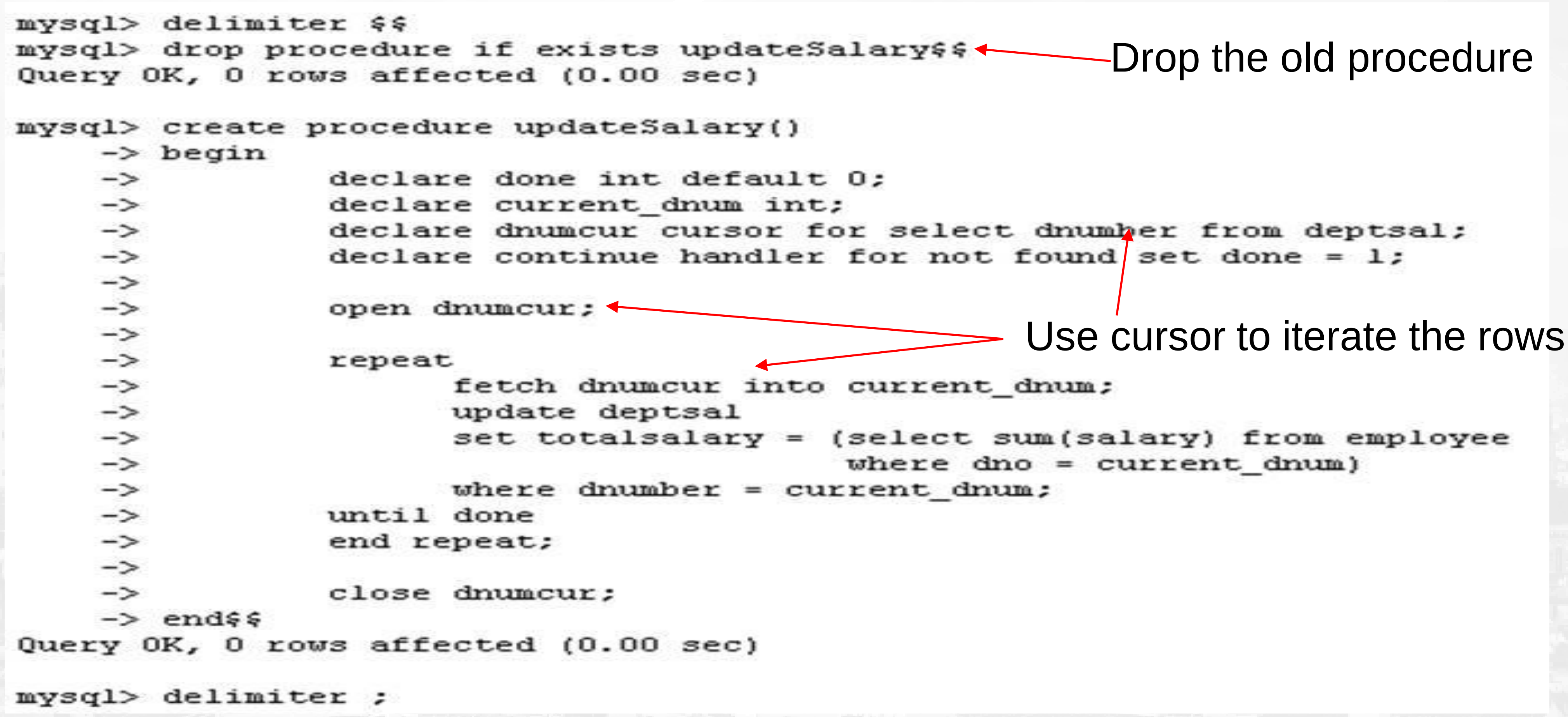
Example using Cursors

- Example 2 in stored procedure updates one row in deptsal table based on input parameter.
- Suppose we want to update all the rows in deptsal simultaneously.
  - First, let's reset the totalsalary in deptsal to zero.



```
mysql> update deptsal set totalsalary = 0;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 3  Changed: 0  Warnings: 0

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |           0 |
|       2 |           0 |
|       3 |           0 |
+---------+-------------+
3 rows in set (0.00 sec)
```

Example using Cursors



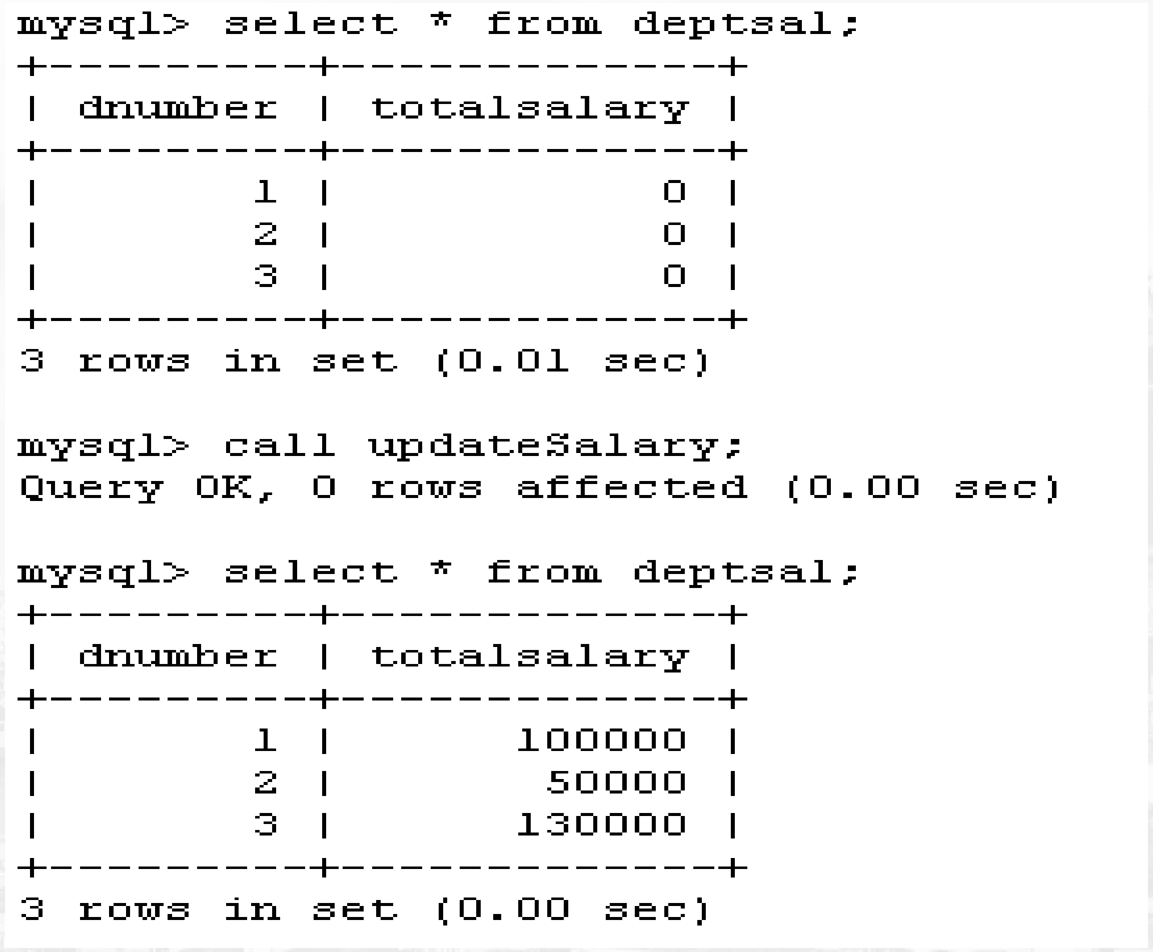
```
mysql> delimiter $$
mysql> drop procedure if exists updateSalary$$
Query OK, 0 rows affected (0.00 sec)

mysql> create procedure updateSalary()
    -> begin
    ->          declare done int default 0;
    ->          declare current_dnum int;
    ->          declare dnumcur cursor for select dnumber from deptsal;
    ->          declare continue handler for not found set done = 1;
    ->
    ->          open dnumcur;
    ->
    ->          repeat
    ->                fetch dnumcur into current_dnum;
    ->                update deptsal
    ->                set totalsalary = (select sum(salary) from employee
    ->                                   where dno = current_dnum)
    ->                where dnumber = current_dnum;
    ->          until done
    ->          end repeat;
    ->
    ->          close dnumcur;
    -> end$$
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Drop the old procedure

Use cursor to iterate the rows

Example using Cursors

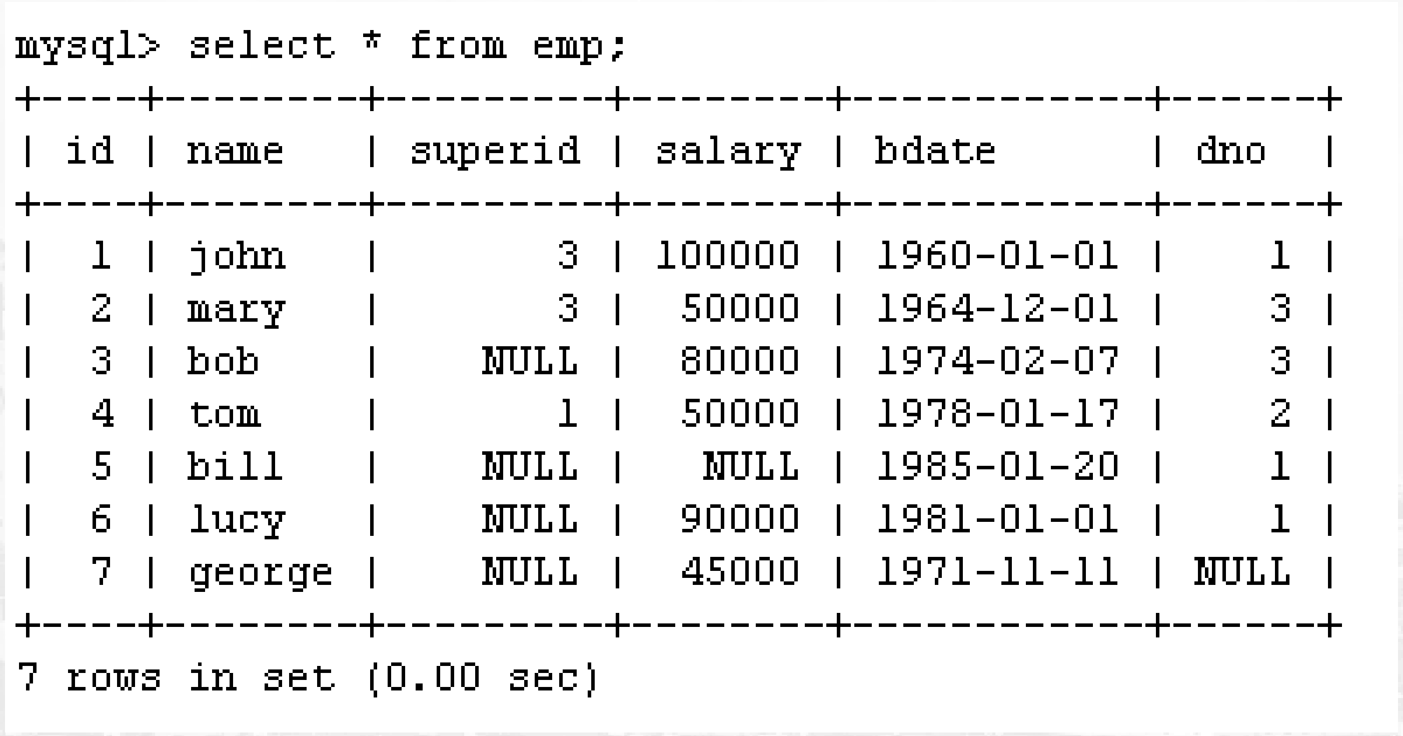
- Call procedure :



```
mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |           0 |
|       2 |           0 |
|       3 |           0 |
+---------+-------------+
3 rows in set (0.01 sec)

mysql> call updateSalary;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      100000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)
```

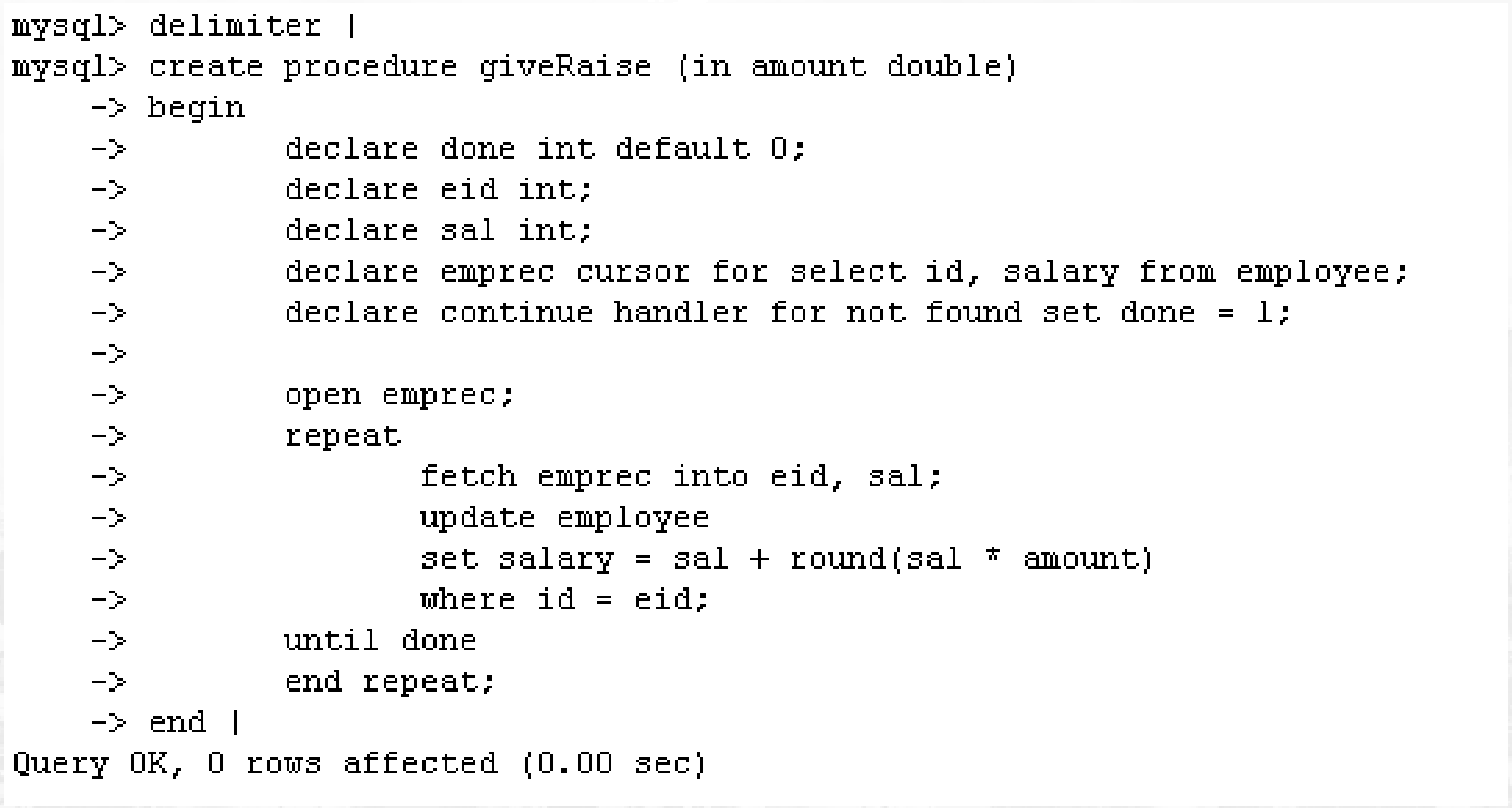
Another Example

- Create a procedure to give a raise to all employees



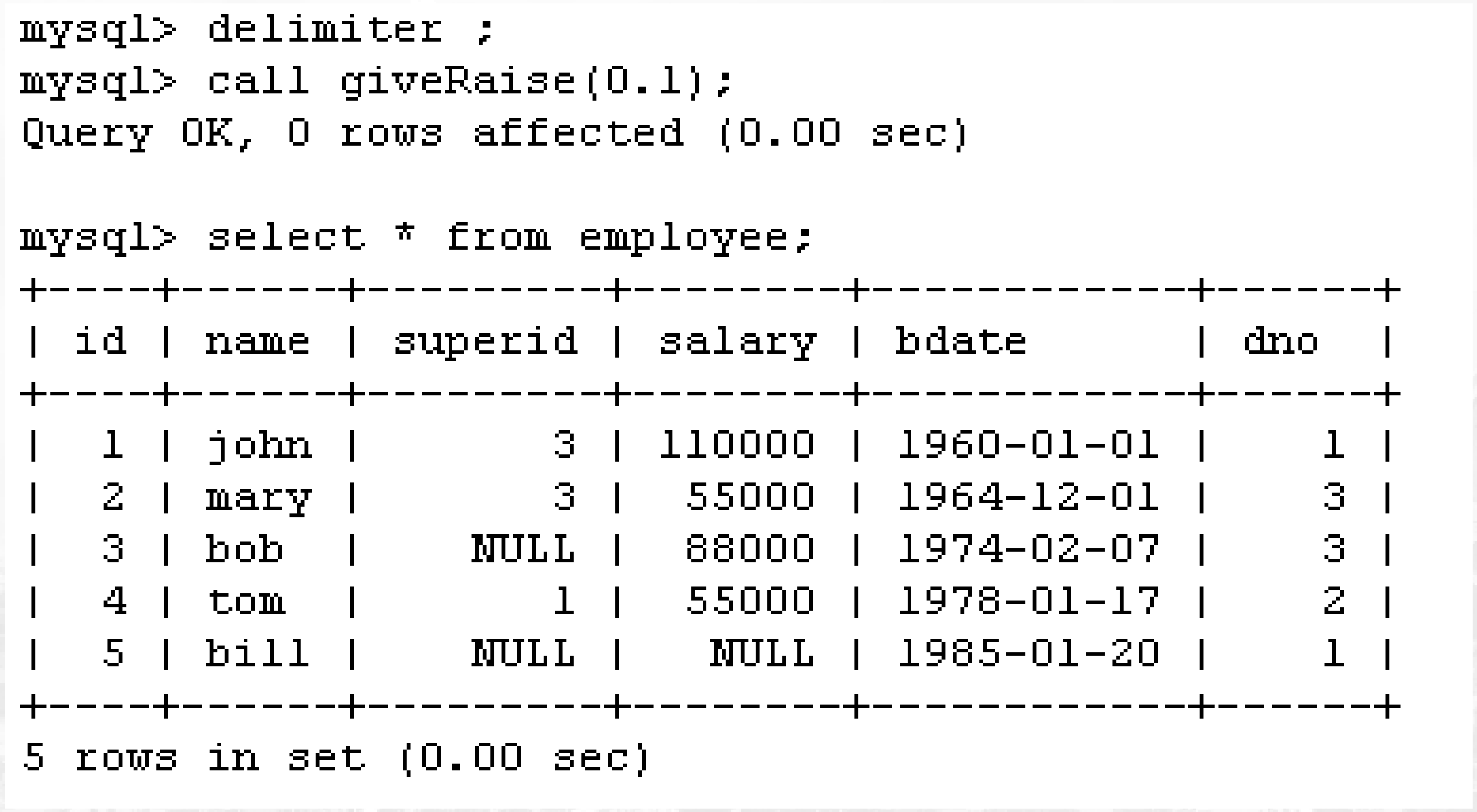
```
mysql> select * from emp;
+----+--------+---------+--------+------------+------+
| id | name   | superid | salary | bdate      | dno  |
+----+--------+---------+--------+------------+------+
|  1 | john   |       3 | 100000 | 1960-01-01 |    1 |
|  2 | mary   |       3 |  50000 | 1964-12-01 |    3 |
|  3 | bob    |    NULL |  80000 | 1974-02-07 |    3 |
|  4 | tom    |       1 |  50000 | 1978-01-17 |    2 |
|  5 | bill   |    NULL |   NULL | 1985-01-20 |    1 |
|  6 | lucy   |    NULL |  90000 | 1981-01-01 |    1 |
|  7 | george |    NULL |  45000 | 1971-11-11 | NULL |
+----+--------+---------+--------+------------+------+
7 rows in set (0.00 sec)
```

Another Example (Cont..)



```
mysql> delimiter |
mysql> create procedure giveRaise (in amount double)
    -> begin
    ->         declare done int default 0;
    ->         declare eid int;
    ->         declare sal int;
    ->         declare emprec cursor for select id, salary from employee;
    ->         declare continue handler for not found set done = 1;
    ->
    ->         open emprec;
    ->         repeat
    ->                 fetch emprec into eid, sal;
    ->                 update employee
    ->                 set salary = sal + round(sal * amount)
    ->                 where id = eid;
    ->         until done
    ->         end repeat;
    -> end |
Query OK, 0 rows affected (0.00 sec)
```

Another Example (Cont..)



```
mysql> delimiter ;
mysql> call giveRaise(0.1);
Query OK, 0 rows affected (0.00 sec)

mysql> select * from employee;
+----+------+---------+--------+------------+------+
| id | name | superid | salary | bdate      | dno  |
+----+------+---------+--------+------------+------+
|  1 | john |       3 | 110000 | 1960-01-01 |    1 |
|  2 | mary |       3 |  55000 | 1964-12-01 |    3 |
|  3 | bob  |    NULL |  88000 | 1974-02-07 |    3 |
|  4 | tom  |       1 |  55000 | 1978-01-17 |    2 |
|  5 | bill |    NULL |   NULL | 1985-01-20 |    1 |
+----+------+---------+--------+------------+------+
5 rows in set (0.00 sec)
```

Triggers

Triggers

- A **trigger** is a statement that is executed automatically by the system as a side effect of a modification to the database i.e. when changes are made to the table.
- To monitor a database and take a corrective action when a condition occurs

Examples:

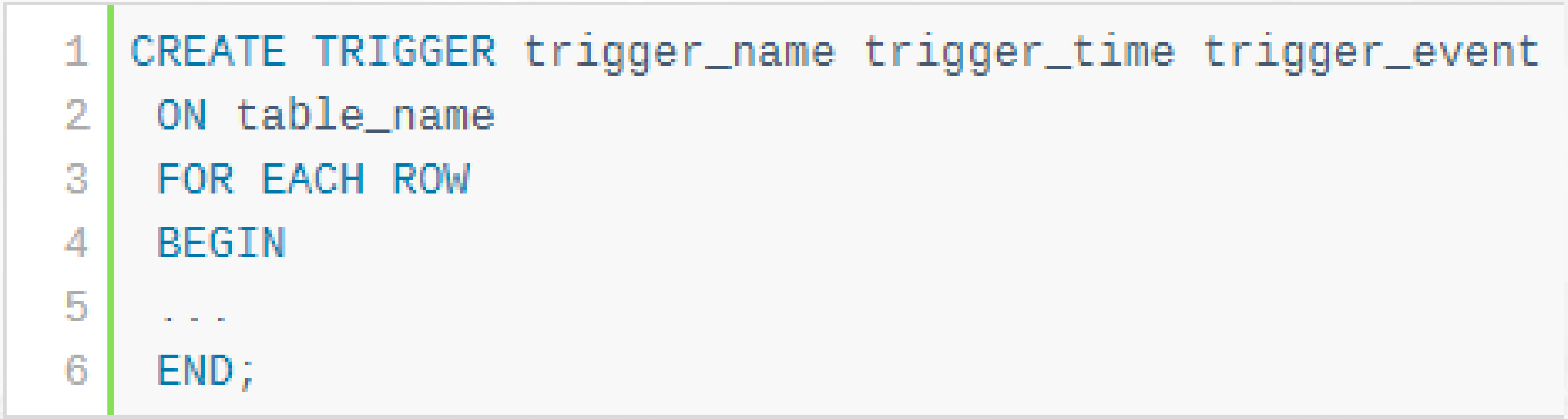
  - Charge $10 overdraft fee if the balance of an account after a withdrawal transaction is less than $500
  - Limit the salary increase of an employee to no more than 5% raise

- SQL triggers provide an alternative way to check the integrity of data.

Triggering Events and Actions in SQL

- A trigger can be defined to be invoked either before or after the data is changed by **INSERT, UPDATE or DELETE .**
- MySQL allows you to define maximum six triggers for each table.
  - BEFORE INSERT – activated before data is inserted into the table.
  - AFTER INSERT- activated after data is inserted into the table.
  - BEFORE UPDATE – activated before data in the table is updated.
  - AFTER UPDATE - activated after data in the table is updated.
  - BEFORE DELETE – activated before data is removed from the table.
  - AFTER DELETE – activated after data is removed from the table.

MySQL Trigger Syntax

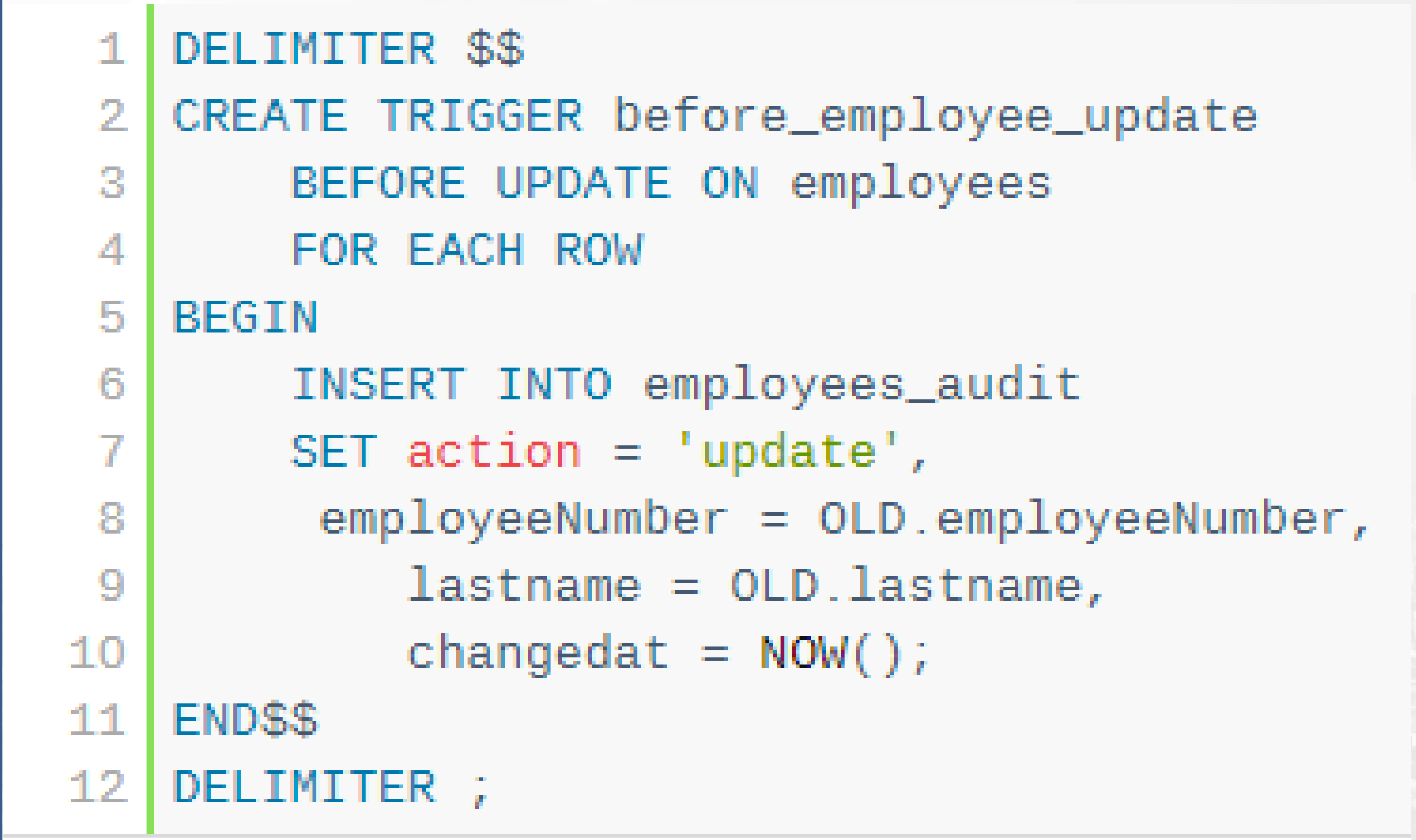


```
CREATE TRIGGER trigger_name trigger_time trigger_event
 ON table_name
 FOR EACH ROW
 BEGIN
 ...
 END;
```

MySQL Trigger Example 1

https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx

- Create a BEFORE UPDATE trigger that is invoked before a change is made to the employees table.
- we used the OLD keyword to access employeeNumber and lastname column of the row affected by the trigger.



```
DELIMITER $$
CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW
BEGIN
    INSERT INTO employees_audit
    SET action = 'update',
     employeeNumber = OLD.employeeNumber,
        lastname = OLD.lastname,
        changedat = NOW();
END$$
DELIMITER ;
```

Continued…

- In a trigger defined for INSERT, you can use NEW keyword only. You cannot use the OLD keyword.
- However, in the trigger defined for DELETE, there is no new row so you can use the OLD keyword only.
- In the UPDATE trigger, OLD refers to the row before it is updated and NEW refers to the row after it is updated.

Example (Cont..)

- Update the employees table to check whether the trigger is invoked.

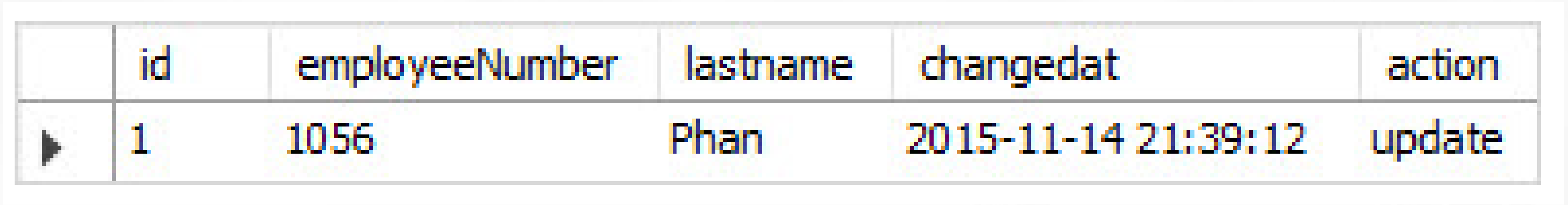
```sql
UPDATE employees
SET
    lastName = 'Phan'
WHERE
    employeeNumber = 1056;
```

- Finally, to check if the trigger was invoked by the UPDATE statement, we can query the employees_audit table using the following query:

```sql
SELECT
    *
FROM
    employees_audit;
```

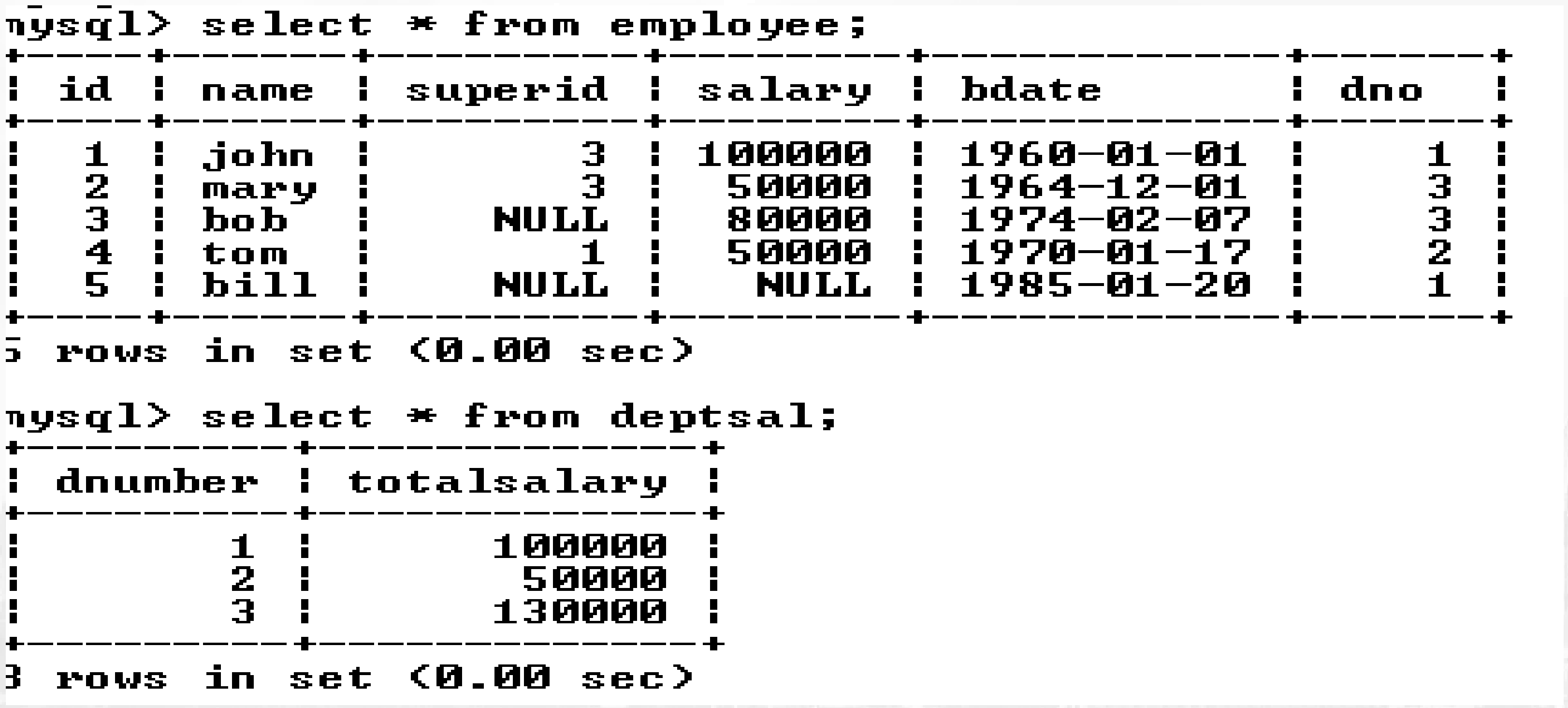
Example (Cont..)

- The following is the output of the query:



	id	employeeNumber	lastname	changedat	action
▶	1	1056	Phan	2015-11-14 21:39:12	update

Example 2

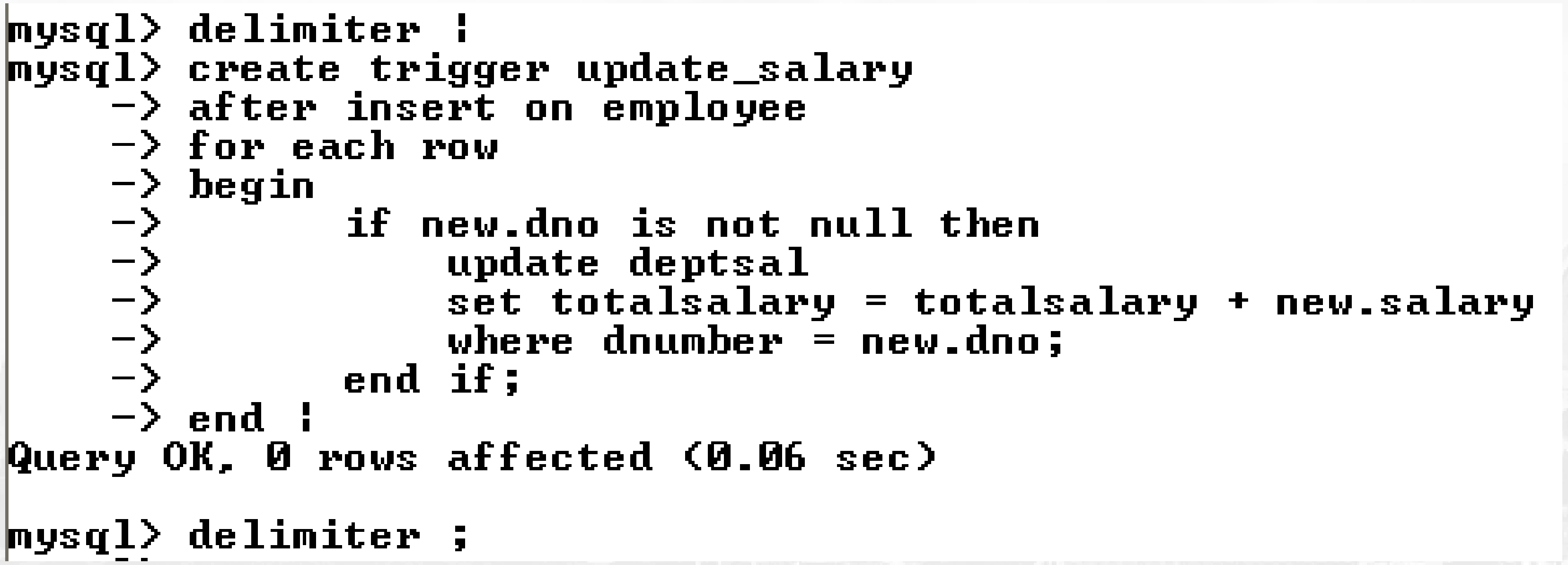


```
mysql> select * from employee;
+----+------+---------+--------+------------+------+
| id | name | superid | salary | bdate      | dno  |
+----+------+---------+--------+------------+------+
|  1 | john |       3 | 100000 | 1960-01-01 |    1 |
|  2 | mary |       3 |  50000 | 1964-12-01 |    3 |
|  3 | bob  |    NULL |  80000 | 1974-02-07 |    3 |
|  4 | tom  |       1 |  50000 | 1970-01-17 |    2 |
|  5 | bill |    NULL |   NULL | 1985-01-20 |    1 |
+----+------+---------+--------+------------+------+
5 rows in set (0.00 sec)

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      100000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)
```

- We want to create a trigger to update the total salary of a department when a new employee is hired

Example 2 (Cont..)

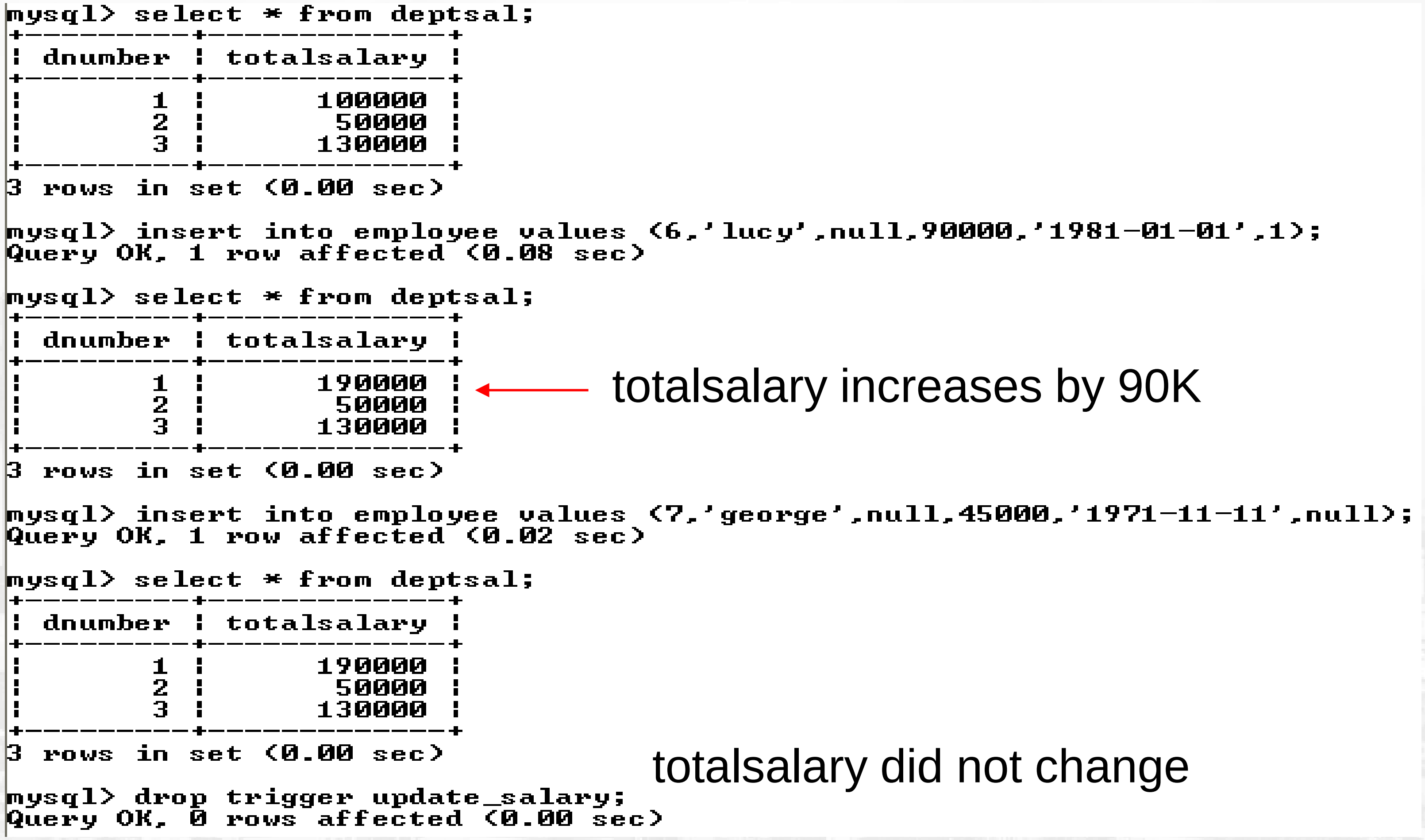
Create a trigger to update the total salary of a department when a new employee is hired:



```
mysql> delimiter |
mysql> create trigger update_salary
    -> after insert on employee
    -> for each row
    -> begin
    ->       if new.dno is not null then
    ->           update deptsal
    ->           set totalsalary = totalsalary + new.salary
    ->           where dnumber = new.dno;
    ->       end if;
    -> end |
Query OK, 0 rows affected (0.06 sec)

mysql> delimiter ;
```

- The keyword "new" refers to the new row inserted

Example 2 (Cont..)



```
mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      100000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)

mysql> insert into employee values (6,'lucy',null,90000,'1981-01-01',1);
Query OK, 1 row affected (0.08 sec)

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      190000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)

mysql> insert into employee values (7,'george',null,45000,'1971-11-11',null);
Query OK, 1 row affected (0.02 sec)

mysql> select * from deptsal;
+---------+-------------+
| dnumber | totalsalary |
+---------+-------------+
|       1 |      190000 |
|       2 |       50000 |
|       3 |      130000 |
+---------+-------------+
3 rows in set (0.00 sec)

mysql> drop trigger update_salary;
Query OK, 0 rows affected (0.00 sec)
```

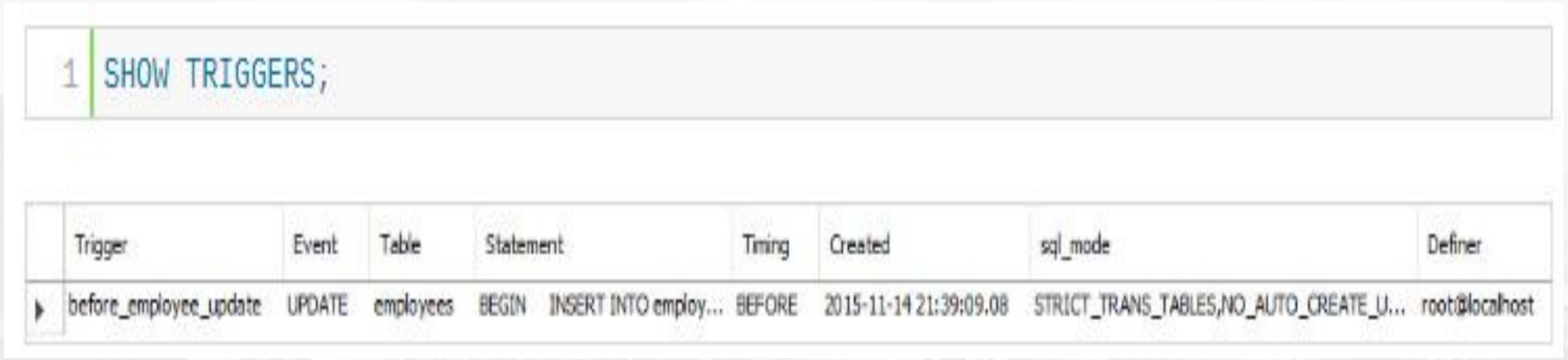
totalsalary increases by 90K

totalsalary did not change

MySQL Trigger

- To list all the triggers we have created:

mysql> show triggers;



```
SHOW TRIGGERS;
```

	Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer
▶	before_employee_update	UPDATE	employees	BEGIN INSERT INTO employ...	BEFORE	2015-11-14 21:39:09.08	STRICT_TRANS_TABLES,NO_AUTO_CREATE_U...	root@localhost

- To drop a trigger

  mysql> drop trigger <trigger name>

Batch 1 Exercise 1 (on trigger)

Consider following three table and create trigger for below exercise

Highschooler(ID int, name text, grade int);
Friend(ID1 int, ID2 int);
Likes(ID1 int, ID2 int);

1. Write one or more triggers to maintain symmetry in friend relationships. Specifically, if (A,B) is deleted from Friend, then (B,A) should be deleted too. If (A,B) is inserted into Friend then (B,A) should be inserted too. Don't worry about updates to the Friend table

Batch 1 Exercise 2

- Create table EMPLOYEE with 5 columns: ENo Number (Employee Number) FName Varchar2 (First name) Age Number (Age of Employee) Grade Varchar2 (Grade of Employee such Asst. prof, Associate Prof. or Professor) Salary Number (Salary of the Empoyee)
- Create a Cursor Emp_Cursor that fetches the record of employee and their salary are incremented according to grade such as for Asst prof, the salary increment is 10000, for Associate Prof 20,000 and for Professor 30,000.

Batch 2 Exercise 1 (on trigger)

- **Consider the following relational schema: BOOK (Isbn, Title, SoldCopies) WRITING (Isbn, Name) AUTHOR (Name, SoldCopies)**
- Define a set of triggers for keeping SoldCopies in AUTHOR updated with respect to: updates on SoldCopies in BOOK insertion of new tuples in the WRITING relation

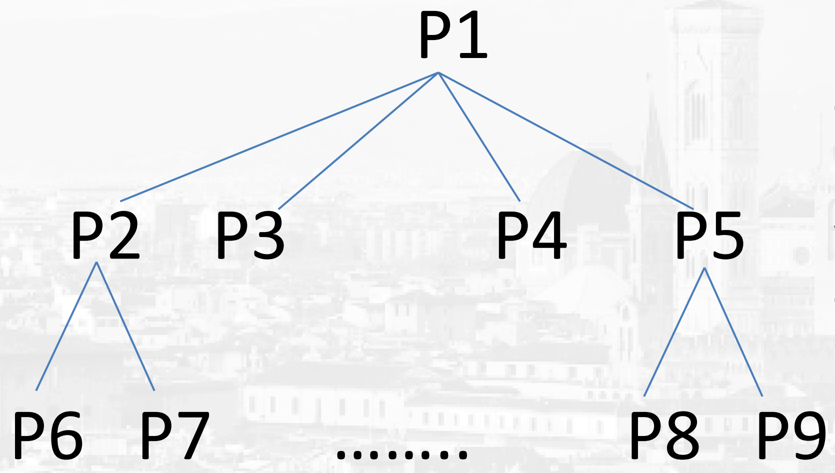
Batch 2 Exercise 2

Use a cursor to calculate compound interest for each customer and insert customer id and simple interest in another table named TEMPLIST.

Customer(cust_id, Prinicipal_amount, Rate_of_interest, No. of Years)

Batch 3 Exercise 1(on trigger)

- The following schema describes a hierarchy of products: Product(Code, Name, Description, SuperProduct, Level) Level describes the depth of the product positioning in the hierarchy tree. Products that are not included into any other product are considered as Level=0 products, with SuperProduct = NULL.
- Hierarchy example:



Define a trigger for the following task:

When a product is deleted, all the sub-products (at any level) must be deleted as well.

Batch 3 Exercise 2

Different states in the US charge different rates for sales tax. If you sell goods to people from different states, you must charge

tax using the rate appropriate for customer state of residence. To handle tax computations, use a table that lists the sales tax rate

for each state

To set up the sales_tax_rate table has two columns: state (a two-letter abbreviation), and tax_rate (a DECIMAL value rather than a FLOAT, to preserve accuracy). and table customer with columns: c_name,product_purchased,price_of_product,city,state

Write a cursor to find sales tax paid by each customer with his city and state of residence?

Batch 4 Exercise 1(On Trigger)

Consider the following relational schema:

STUDENT (ID, Name, Address, Phone, Faculty, Year, Campus, EarnedCredits)

ENROLLMENT (StudID, CourseCode, Year, Date)

COURSEYEAR (CourseCode, Year, Teacher, Quarter, #Students, #ExternalStuds)

COURSE (coursecode, Name, Credits, Campus)

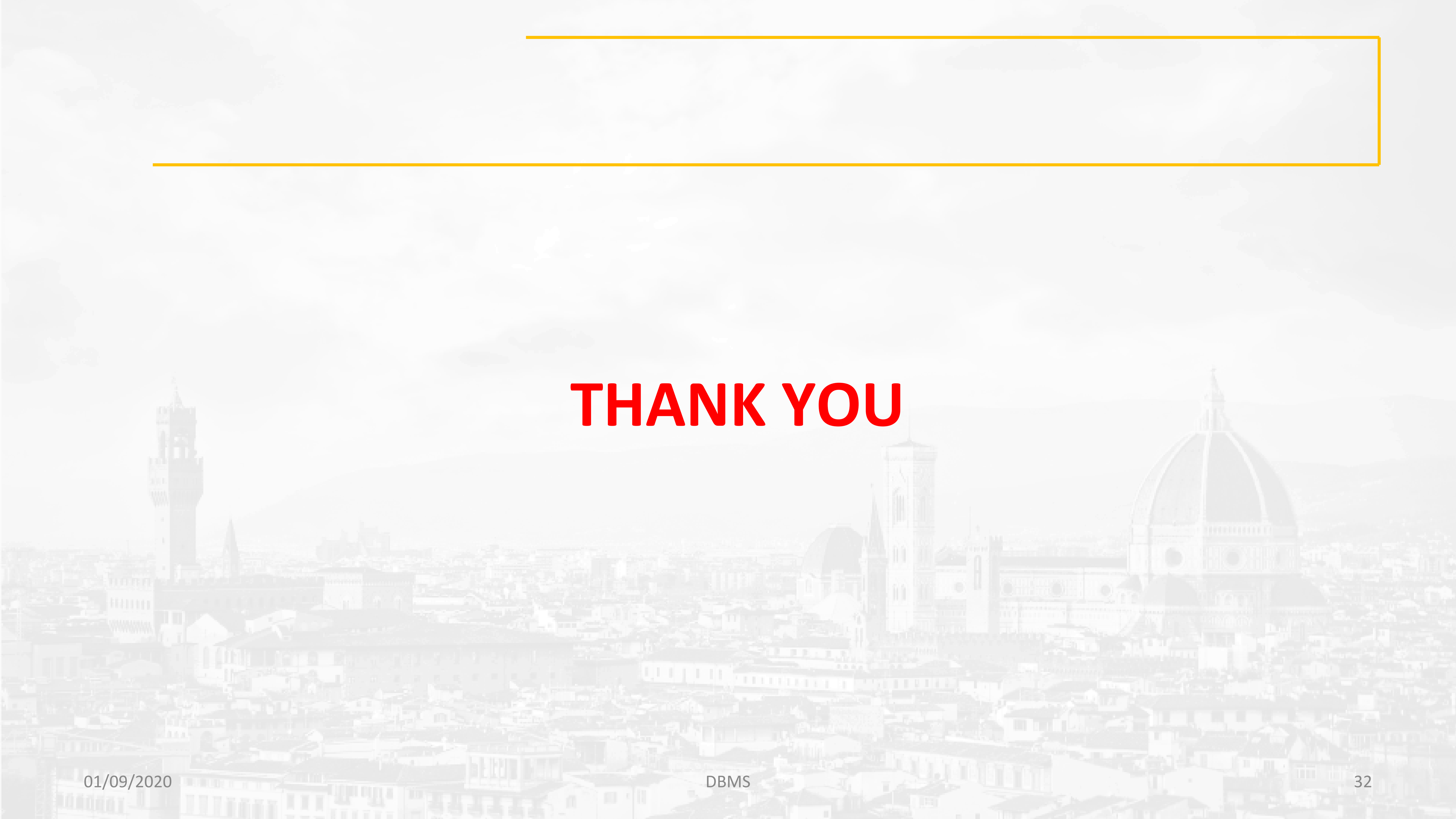
Write a trigger that rolls back the creation of a new Enrollment if the referenced Student and/or COURSEYEAR does not exist in the corresponding table

Batch 4 Exercise 2

- consider the following tables

drive(drive_name,directory_name,file_name)
file(file_name,time_of_creation,time_of_last_access,size_of_file,file_format)

- write cursor to input file _name and if the file is present in drive then makes it's entry in log table with all it's attributes.



THANK YOU