



Dr. Vishwanath Karad

**MIT WORLD PEACE
UNIVERSITY** | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

DBMS-Unit II

Database Languages and Programming

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.

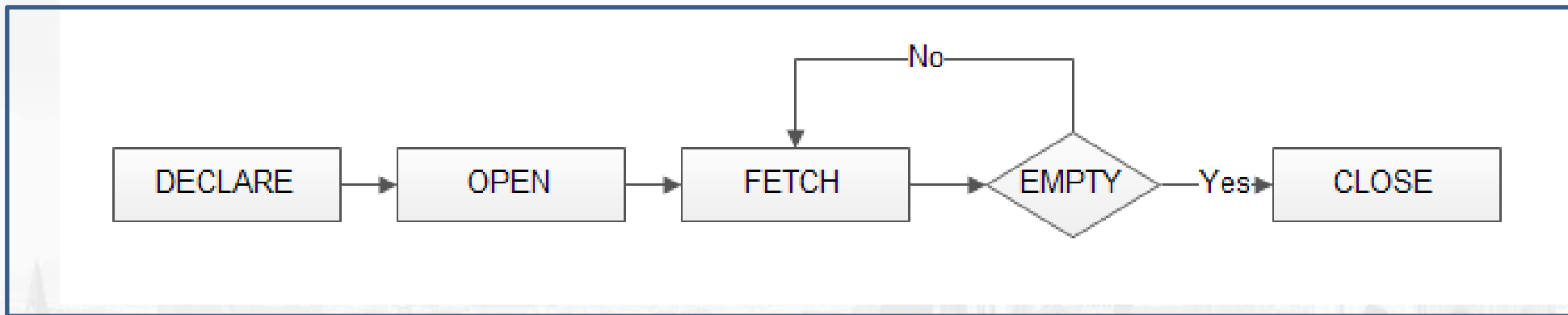
```
FETCH cursor_name INTO variables list;
```

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 1 - Cursors

- 1) Retrieve employees one by one and print out a list of those employees currently working in the DEPARTMENT_ID = 80

```
create procedure p_dept()
begin
declare done int default 0;
declare v_eno int;
declare v_ename varchar(10);
declare v_deptno int;
declare c1 cursor for select eno,ename,deptno from employee where deptno=80;
declare continue handler for not found set done =1;
open c1;
repeat
fetch c1 into v_eno,v_ename,v_deptno;
if done=0 then
select v_eno,v_ename,v_deptno;
end if;
until done end repeat;
close c1;
end;
```

```
mysql> call p_dept//
```

v_eno	v_ename	v_deptno
1	Anil	80

1 row in set (0.00 sec)

Example 2

- 2) Use a cursor to retrieve employee numbers and names from employee table and populate a database table, TEMP_LIST, with this information.

```
create procedure p_emp()
begin
declare done int default 0;
declare v_eno int;
declare v_ename varchar(10);
declare v_deptno int;
declare c1 cursor for select eno,ename,deptno from employee ;
declare continue handler for not found set done =1;
open c1;
repeat
fetch c1 into v_eno,v_ename,v_deptno;
if done=0 then
insert into temp_list values(v_eno,v_ename,v_deptno);
end if;
until done end repeat;
close c1;
end;
```

```
mysql> call p_emp();//
Query OK, 0 rows affected (0.53 sec)
```

```
mysql> select * from temp_list//
```

eno	ename	deptno
1	Anil	80
2	Anita	80
3	Sunita	80
4	Sumita	80
5	Sushmita	10

```
5 rows in set (0.00 sec)
```

Example 3

3. **Create a PL/SQL block that determines the top employees with respect to salaries. Accept a number n from the user where n represents the number of top n earners from the EMPLOYEES table. For example, to view the top five earners, enter 5. Test a variety of special cases, such as $n = 0$ or where n is greater than the number of employees in the EMPLOYEES table. The output shown represents the five highest salaries in the EMPLOYEES table.**

Cont..

```
create procedure p_top(v_n int)
begin
declare done int default 0;
declare v_eno int;
declare v_ename varchar(10);
declare v_deptno int;
declare v_salary int;
declare v_cnt int default 0;
declare c1 cursor for select eid,ename,dno,salary from emp order by salary desc;
declare continue handler for not found set done =1;
open c1;
repeat
fetch c1 into v_eno,v_ename,v_deptno,v_salary;
if done=0 AND v_cnt<v_n then
select v_eno,v_ename,v_salary;
end if;
set v_cnt=v_cnt+1;
until done end repeat;
close c1;
end;
//////////output//////////

mysql> call p_top(3)//
+-----+-----+-----+
| v_eno | v_ename | v_salary |
+-----+-----+-----+
|      3 | Sunita  |    51900 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Example 4

4. 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)
```

Cont..

```
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

Cont..

- 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)
```

Example 5

5. 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)
```


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)
```

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

```
1 CREATE TRIGGER trigger_name trigger_time trigger_event
2   ON table_name
3   FOR EACH ROW
4   BEGIN
5     ...
6   END;
```

Cont..

- 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 1 - Trigger

1. Create a trigger to simulate Recycle Bin for employee table. If any row gets deleted from Emp, same row must get stored in Emp_Bin

Emp(Eno,ENAME,Salary)

Emp_Bin(Eno,ENAME,Salary)

```
create trigger t_bin
before delete on emp for each row
begin
insert into temp_emp values(OLD.eid,OLD.ename,OLD.dno,OLD.salary);
end;
```

```
mysql> delete from emp where eid=1//
Query OK, 1 row affected (0.16 sec)
```

```
mysql> select * from temp_emp//
+-----+-----+-----+-----+
| eid   | ename | dno   | salary |
+-----+-----+-----+-----+
| 1     | Anil  | 10    | 10900  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Example 2

2. 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.

```
1 DELIMITER $$
2 CREATE TRIGGER before_employee_update
3     BEFORE UPDATE ON employees
4     FOR EACH ROW
5 BEGIN
6     INSERT INTO employees_audit
7     SET action = 'update',
8         employeeNumber = OLD.employeeNumber,
9         lastname = OLD.lastname,
10        changedat = NOW();
11 END$$
12 DELIMITER ;
```

Cont..

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

```
1 UPDATE employees
2 SET
3     lastName = 'Phan'
4 WHERE
5     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:

```
1 SELECT
2     *
3 FROM
4     employees_audit;
```


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 3

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

```
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)
```

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

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)
```

← totalsalary increases by 90K

```
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 did not change

Trigger

- To list all the triggers we have created:

mysql> show triggers;

```
1 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>

References

1. MySQL Tutorial

<http://www.mysqltutorial.org/>