**MySQL-PL - Cursors**

**#1**

Delimiter //

Create procedure abc ()

Begin

Declare a int;

Declare b varchar (15);

Declare c int;

Declare d int;

Declare y int default 0;

Declare c1 cursor for select \* from emp;

Declare continue handler for not found set y = 1;

Open c1;

Cursor\_c1\_loop: loop

Fetch c1 into a, b, c, d;

If y =1 then

Leave cursor\_c1\_loop;

End if;

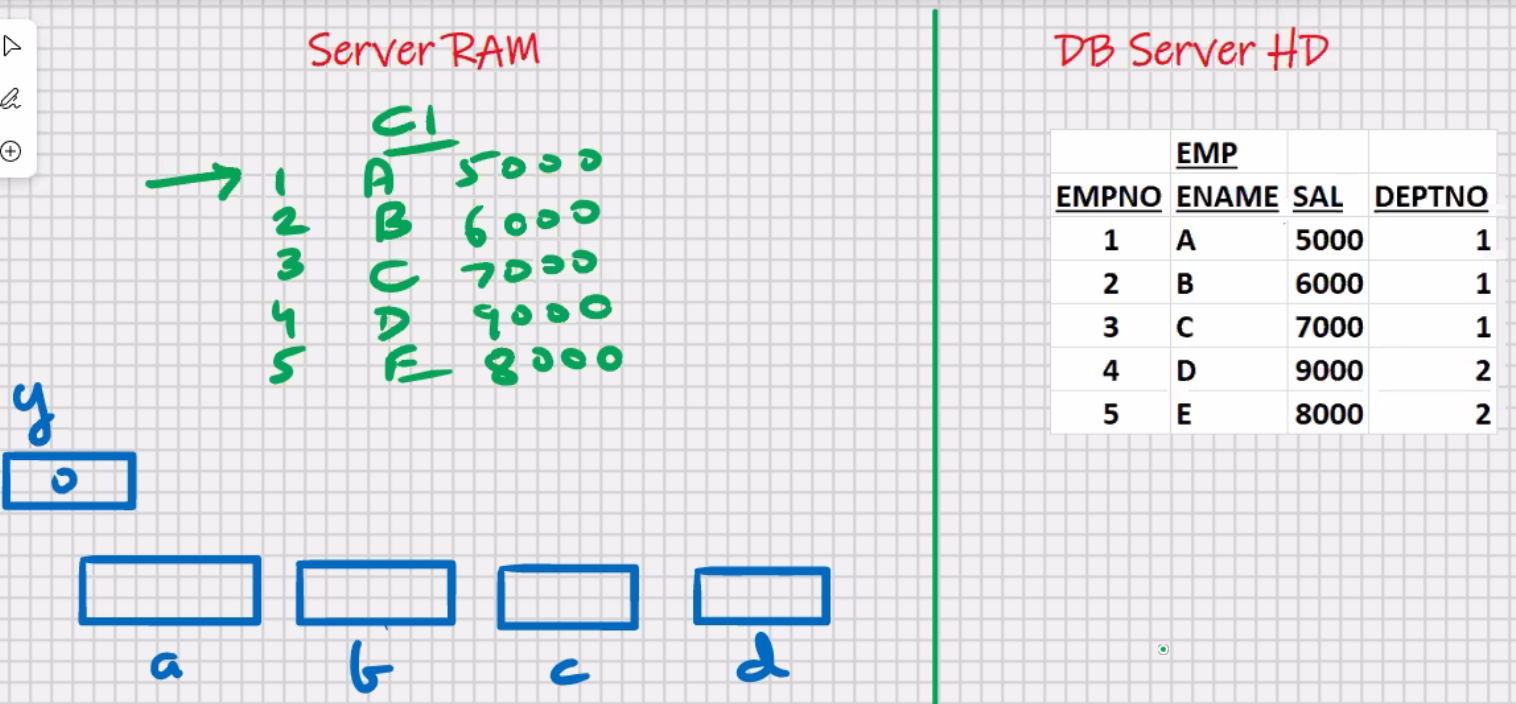
Update emp set sal = sal + 1;

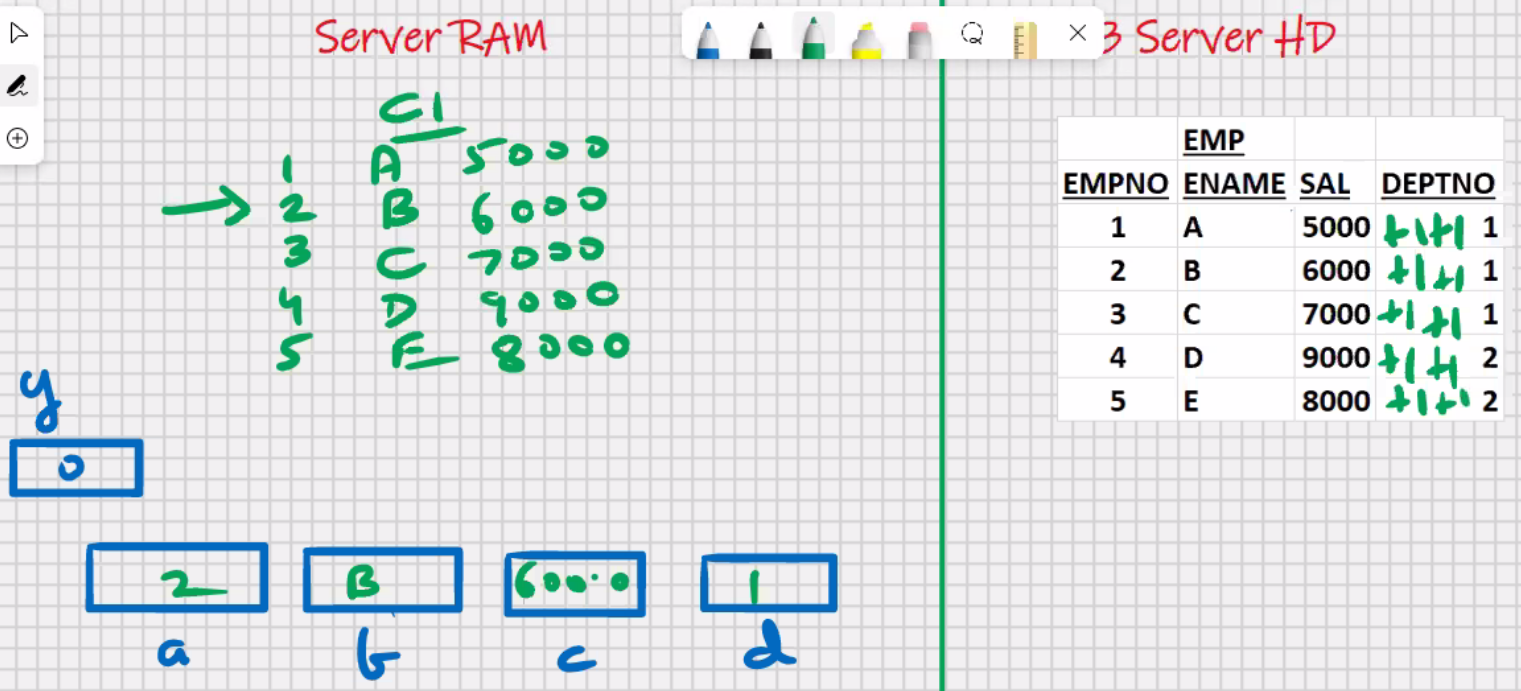
End loop cursor\_c1\_loop;

Close c1;

End; //

Delimiter;





**#2**

Delimiter //

Create procedure abc ()

Begin

Declare a int;

Declare b varchar (15);

Declare c int;

Declare d int;

Declare y int default 0;

Declare c1 cursor for select \* from emp;

Declare continue handler for not found set y = 1;

Open c1;

Cursor\_c1\_loop: loop

Fetch c1 into a, b, c, d;

If y =1 then

Leave cursor\_c1\_loop;

End if;

If c>7000 then

Update emp set sal = sal + 1;

End if;

End loop cursor\_c1\_loop;

Close c1;

End; //

Delimiter;

**#3**

Delimiter //

Create procedure abc ()

Begin

Declare a int;

Declare b varchar (15);

Declare c int;

Declare d int;

Declare y int default 0;

Declare c1 cursor for select \* from emp **for update**;

Declare continue handler for not found set y = 1;

Open c1;

Cursor\_c1\_loop: loop

Fetch c1 into a, b, c, d;

If y =1 then

Leave cursor\_c1\_loop;

End if;

If c>7000 then

Update emp set sal = sal + 1 **where empno = a;**

End if;

End loop cursor\_c1\_loop;

Close c1;

**Commit;**

End; //

Delimiter;

**#4**

Delimiter //

Create procedure abc ()

Begin

Declare a int;

Declare b varchar (15);

Declare c int;

Declare d int;

Declare y int default 0;

Declare c1 cursor for select \* from emp **for update**;

Declare continue handler for not found set y = 1;

Open c1;

Cursor\_c1\_loop: loop

Fetch c1 into a, b, c, d;

If y =1 then

Leave cursor\_c1\_loop;

End if;

If c>7000 then

**delete** emp **where empno = a;**

End if;

End loop cursor\_c1\_loop;

Close c1;

**Commit;**

End; //

Delimiter;

**Cursors uses**

* Storing/processing multiple rows
* **USED TO LOCK THE ROWS MANUALLY**

**#5**

Delimiter //

Create procedure abc()

begin

Declare c1 cursor for Select \* from emp for update;

Open c1;

Close c1;

End; //

Delimiter ;

Call abc();

* **LOCKS ARE AUTOMATICALLY RELEASED WHEN YOU ROLLBACK OR COMMIT**

**#6**

Delimiter //

Create procedure abc ()

begin

Declare c1 cursor for Select \* from emp

Where deptno = 1 for update;

Open c1;

Close c1;

End; //

Delimiter ;

Call abc();

**Parameters are of 3 types**

1. **In (by default)**

* Read only
* Can pass a constant, variable and expression
* Call by value
* Fastest in terms of processing speed
* If you don’t want to return a value, then use IN parameter

**#7**

Delimiter //

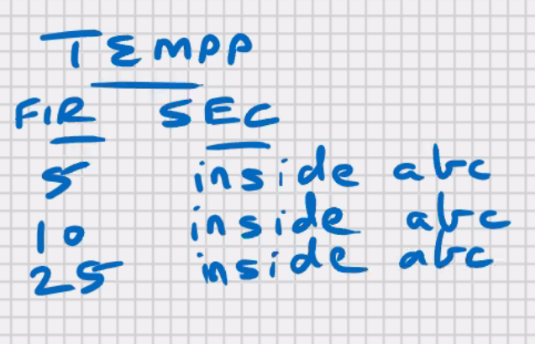
Create procedure abc (in y int)

Begin

--Set y = 100; <- ERROR

Set x = y + 10 ; <- Allowed

Insert into tempp values (y, ‘inside abc’);

End ; //

Delimiter ;

Call abc (5);

Set @x = 10;

Call abc(@x);

Set @x = 10;

Call abc(2\*@x+5);

--------------------------------------------------------------------------------------------

**#8**

Delimiter //

Create procedure abc (in y int)

Begin

Insert into tempp values (y, ‘inside abc’);

End ; //

Delimiter ;

Delimiter //

Create procedure pqr ()

Begin

Declare x int default 10;

Call abc(5) ;

Call abc(x);

Call abc(2\*x+5);

End ; //

Delimiter ;

Call pqr();

========================================================

1. **Out**

* Write only
* Can pass variables only
* Call by reference
* Procedure can return a value indirectly if you call by reference
* **MOST SECURE**
* if you are working on a public network, e.g. Internet

(e.g. username, password, OTP, etc.)

**#9**

Delimiter //

Create procedure abc (out y int)

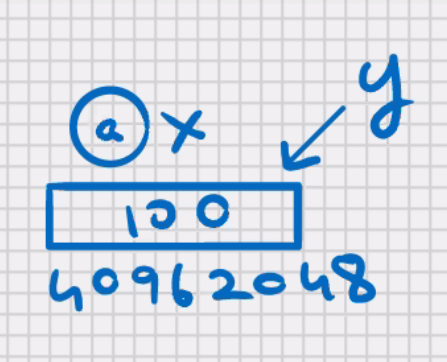
Begin

-- Set x = y ; -> ERROR cant read from y

Set y = 100;

End ; //

Delimiter ;

Set @x = 10 ;

Select @x from dual;

10

**Call abc (@x);**

Select @x from dual;

100

**#10**

Delimiter //

Create procedure abc (out y int)

Begin

set y = 100;

End ; //

Delimiter ;

Create procedure pqr ()

Begin

Declare x int default 10;

Insert into tempp values (x, ‘before abc’);

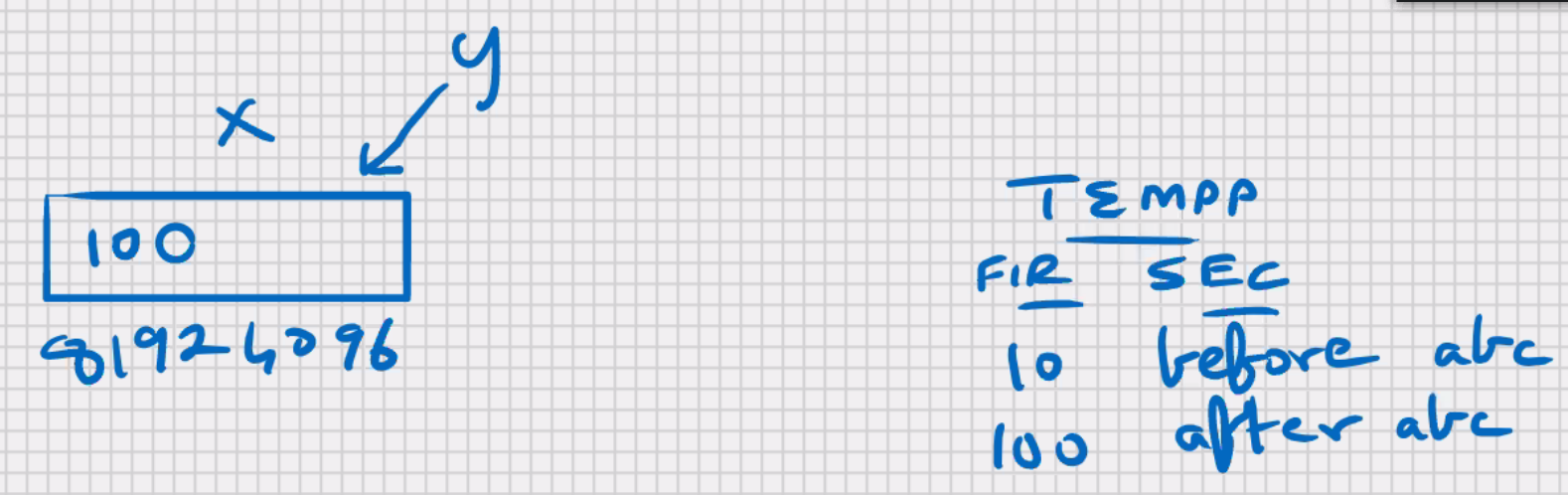
Call abc(x);

Insert into tempp values(x, ‘after abc’);

End ; //

Delimiter ;

Call pqr();



========================================================

1. **inout**

* Read and Write
* Can pass variables only
* Call by reference
* Procedure can return a value indirectly if you call by reference
* **MOST POWERFUL**
* if you are working on a local network

**#11**

Delimiter //

Create procedure abc (inout y int)

Begin

Set y = y \* y \* y;

End ; //

Delimiter ;

Set @x = 10 ;

Select @x from dual;

10

**Call abc (@x);**

Select @x from dual;

100

**#12**

Delimiter //

Create procedure abc (out y int)

Begin

set y = y\*y\*y;

End ; //

Delimiter ;

Create procedure pqr ()

Begin

Declare x int default 10;

Insert into tempp values (x, ‘before abc’);

Call abc(x);

Insert into tempp values(x, ‘after abc’);

End ; //

Delimiter ;

Call pqr();

**Stored objects**

* objects that are stored in the database
* e.g. create ……. Tables, indexes, views, procedures
* anything that you do create command is a stored object

**STORED FUNCTIONS**

* Routine that **returns a value** directly and compulsorily
* Global functions
* Can be called through MySQL command line client, MySQL Workbench, MySQL- PL, Java, MS.NET ,etc
* Can be called through any front-end s/w;
* Unlike a procedure, a function cannot be called by itself, because a function returns a value, and that value has to be stored somewhere
* Therefore a function has to be equated with a variable, or it has to be a part of some expression
* Etc. points and benefits are same as procedures.
* In parameter only

Call y = abc();

**Functions are of 2 types:**

1.Deterministic

2.Not Deterministic

* for the same input parameters, if the stored function returns the same result, it is considered deterministic, and otherwise the stored function is not deterministic
* you have to decide whether a stored function is deterministic or not
* if you declare it incorrectly, the stored function may produce an unexpected result, or the available optimization is not

used which degrades the performance

* only the current date and time are used inside the body of the function, then it will be not deterministic, else mostly it is going to be deterministic

**#13**

delimiter //

create function abc ()

returns int

deterministic

begin

return 10;

end; //

delimiter ;

Function created.

delimiter //

create procedure pqr ()

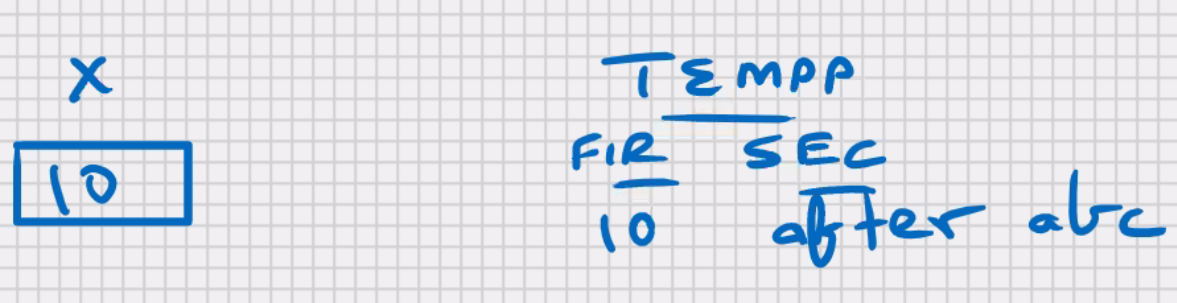
begin

declare x int;

set x = abc (); -- 10

insert into tempp values (x, 'after abc');

end; //

delimiter ;

call pqr();

**#14**

delimiter //

create function abc (y int)

returns int

deterministic

begin

return y\*y;

end; //

delimiter ;

delimiter //

create procedure pqr ()

begin

declare x int;

set x = abc (10);

insert into tempp values (x, 'after abc');

end; //

delimiter ;

call pqr ();

difference between stored function and stored procudures

* stored function can be called in select statement

select abc(sal) from emp;

* stored function can be called in sql commands

select abc (10) from dual;

delete from emp where abc(sal) = 1000000;

**#15**

delimiter //

create function abc (y int)

returns boolean

deterministic

begin

if y > 5000 then

return TRUE;

else

return FALSE;

end if;

end; //

delimiter ;

delimiter //

create procedure pqr ()

begin

declare x int;

select sal into x from emp where ename = 'KING';

if abc(x) then

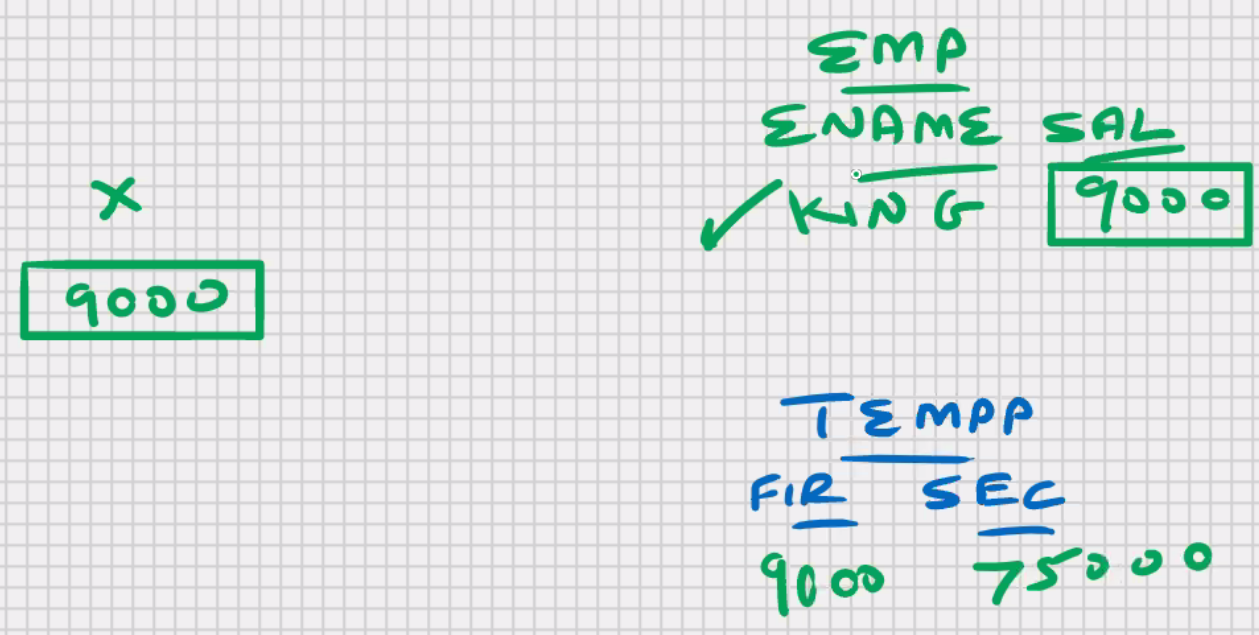
insert into tempp values (x, '> 5000');

else

insert into tempp values (x, '<= 5000');

end if;

end; //

delimiter ;

call pqr ();

**to drop the function: -**

drop function abc;

**to see which all functions are created: -**

show function status; -> shows all functions in all schemas

show function status where db = 'cdac';

show function status where name like 'a%';

**to view the source code of stored function:** -

show create function abc;

**to share the function with n other users: -**

edac\_mysql> grant execute on function abc to scott;

scott\_mysql> select cdac.abc() from dual;

edac\_mysql> revoke execute on function abc from scott;

