#

01/02/2021

EMP

ENAME	SAL	JOB	DeptNO.
SCOTT	3000	Clerk	10
KING	5000	Manager	20

- delimiter //
  create procedure abcc()
  begin
  declare x int;
  select sal into x from emp
  where ename = 'KING';
  /* Processing, e.g. set hra = x*0.4, etc. */
  insert intotempp values cx, 'KING'));
end;//
delimiter ;

- select columnname into variablename from ....
  where ....

---

delimiter //
create procedure abc (y char(5))
begin
declare x int;
select sal into xfrom emp
where ename = y;
insert into tempp values (x, 'KING');
Y
end;//
delimiter ;

call abcc KING);
call abc (SCOTT);

```
delimiter//
Create procedure abcc()
 begin
    declare x int;
    declare y char(15));
    select sal, Job into x, y from emp
    where ename ='KING';
    /* Processing, e.g. set hra = x*0.4, lower(y) etc. */
     insert into tempp values(x,y);
 end;//
delimiter;
```

- declare emp table 1st and then declare x & y accordingly same as emp table.

Decision Making (If statement):-

```
delimiter//
create procedure abcc()
begin
  declare x int;
  select sal into x from emp where ename = 'KING';
   if x>5000 then
      insert into tempp values(x,'High sal');
   end if;
endi;//
delimiter;
```

- if condition then
  - - - - - - ;
  - - - - - - ;

endif;

else is optional.

```
if condition then
- - - - - - ;
else
- - - - - - ;
endif;
```

```
else
   insert into tempp values(x,'Low sal');
endif;
```

(Nested if)

```
delimiter //
create procedure abc()
begin
   declare x int;
   select sal into x from emp where ename = 'KING';
   if x >5000 then
      insert into tempp values (x, 'High sal');
   else
      if x < 5000 then
       insert into tempp values( x, 'Low sal');
      else
       insert into tempp values( x, 'Medium sal');
      endif;
   end if;
end; //
delimiter;
```

```
delimiter //
create procedure abc()
begin
   declare x int;
   select sal into x from emp where ename = 'KING';
   if x > 5000 then
     insert into tempp values( x, 'High sal');
   elseif x < 5000 then
     insert into tempp values("x", 'Low sal');
     else
       insert into tempp values(x, 'medium sal');
     end if
end; //
 delimiter;
```

```
if ----
else if - - .
   elseif - - .
   else - - .
   end if;
```

Snehal Sawant

```
delimiter //
 create procedure abc()
  begin
    declare x boolean default True;
    if x then
      insert into tempp values(1,'Mumbai');
    end if;
 end; //
 delimiter;
```

```
delimiter//
 create procedure abc()
 begin
    declare x boolean default False;
    if not x then
      insert into tempp values( 1,'kolkata');
    endif;
  end;//
  delimiter;
```

Loops (for repetitive / iterative processing).

① While loop

- check for some condition before entering the loop.

```
WHILE expression DO
        ------;
        ------;
END WHILE;
```

```
delimiter//
 create procedure abcc()
 begin
     declare x int default 1;
     while x< 10 do
       insert into tempp values(x,'in while loop');
       set x=x+1;    /(x-1)
     end while;
  end;//
 delimiter;
```

Snehal Sawant

```
delimiter //
create procedure abcc()
begin
    declare x int default 1;
    declare y int default 1;
    while x<10 do
        while y<10 do
            insert into tempp values(y,'in y loop');
            set y=y+1
        end while;
        insert into tempp values(x,'in x loop');
        set x = x+1;
    end while;
end; //
delimiter;
```

Repeat loop (similar to Do-while loop)

- There is no condition to enter the loop, but there is a condition to exit the loop.
- it will execute at least once (e.g. outer join/ Master-data)

```
Repeate
        ------;
        ------;
UNTIL expression_is_not_satisfied.
END Repeat;
delimiter //
Create procedure abcc()
begin
    declare x int default 1;
    repeat
        insert into tempp values(x,'inloop');
        set x=x+1;
    until x>5
    end repeat;
end; //
delimiter;
```

Snehal Sawant

- Loop, leave and iterate statements:-
  - leave statement allows you to exit the loop (similar to 'break' statement of 'C' programming)
  - iterate statement allows you to skip the entire code under it and start a new iteration (similar to 'continue's statement of 'C' programming)
  - Loop statement executes a block of code repeatedly with an additional flexibility of using loop label.

```
delimiter//
Create procedure abc(c)
 begin
     declare x int default 1;
     pqr_loop : loop.
            if x>10 then
                  leave pqr-loop;
            endif;
     set x = x+1;
     if mod(x,2) !=0 then
          iterate pqr-loop;
     else
          insert into tempp values(x,'inside loop');
      endif;
 endloop;
 end;//
 delimiter;
```

Global variables:-

set @x = 10;

- @x remains in the RAM till you exit (end of session)
- can be accessed in all procedures.

- select @x from dual;
  - you will have to create and initialize at the same time.
  - stores as character string.

Snehal Sawant

```
set @x=10;
select @x from dual;
set @x=@x+1;
select @x from dual;
```

Cursor - V.V.V.IMP:

EMPNO.	ENAME	SAL	DEPTNO.
1	A	5000	1
2	B	6000	1
3	C	7000	1
4	D	9000	2
5	E	8000	2
int	varchar(15)	int	int

DB server HD.

TEMPP.

FIR SEC.

IO Server RAM

5000

hra

- Cursors presents in all RDBMS and some of DBMS, and some the front-ends also.
- types of variables
- cursor can store multiple rows.

HRA_Calc → 40% sal.

```
declare x int;
declare hra float;
select
```

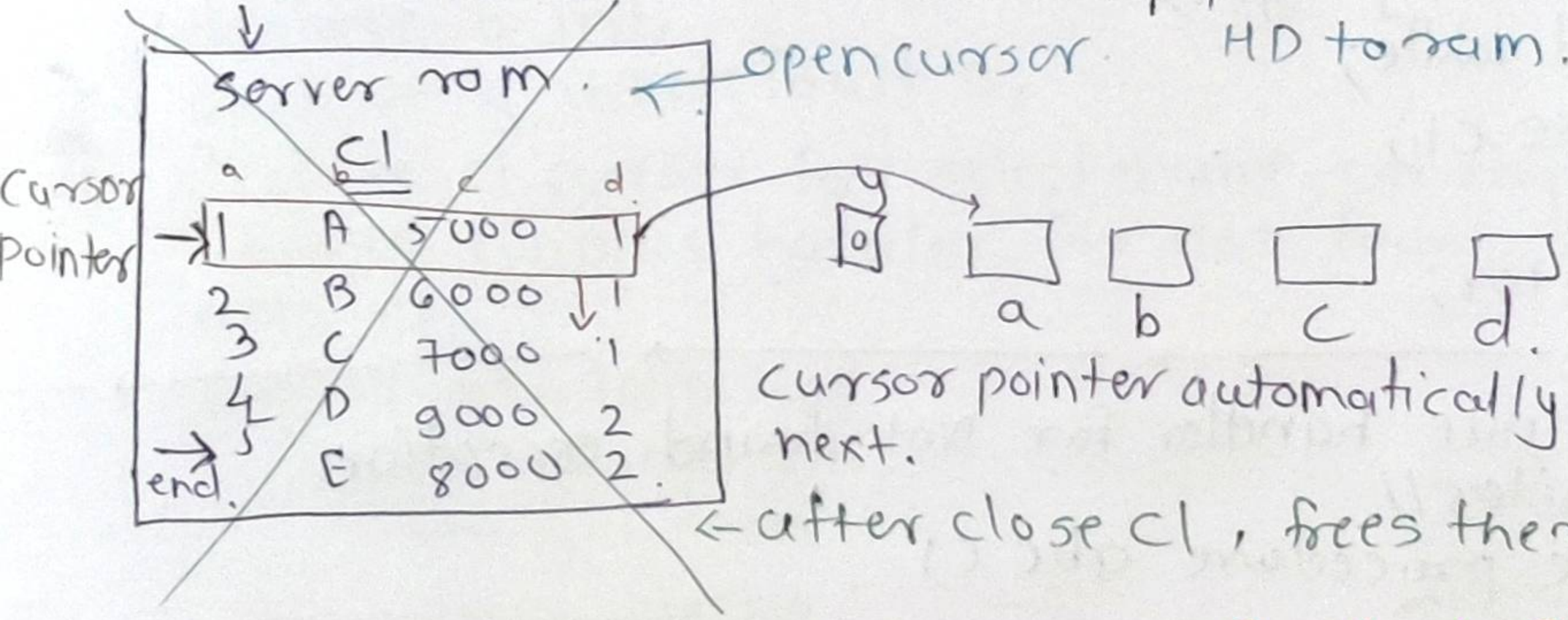
- similar to 2D array.
- cursors are used for storing multiple rows.
- cursors are used for processing multiple rows.
- cursors are used for handling multiple rows.
- cursors are used for storing the data temporarily

- declare pqr cursor for select * from emp;
  - cursor is based on select statement.
- declare pqr cursor for select * from emp where deptno = 1;

Snehal Sawant

```
delimiter //
Create procedure abc()
begin
    declare a int;
    declare b varchar(15);
    declare c int;
    declare d int;
    declare y int default 0;                    ← No data in cursor
    declare c1 cursor for select * from emp;  ←
    open c1;                                    Cursor declaration Definition
    while y<5 do  (try y<3)
        fetch c1 into a, b, c, d; ← fetches new row.
        /* processing, eg. set hra = c*0.4, etc. */
        insert into tempp values (a, b);
        set y = y+1;
    end while;
    closec1; ← close to cursor c1 & free the ram.
end; //
delimiter;
```

- open c1; ← This will open the cursor c1, it will execute the select statement, and it will popwlate the cursor c1



- closing c1 is optional.

Snehal Sawant

- Cursor is a read only variable
- data that is present in cursor, it cannot be manipulated.
- you will have to fetch 1 row at a time into some intermediate variables, and do your processing with those variables.
- you can only fetch backward in A MySQL cursor.

- you can fetch sequentially (top to bottom).
- you can only fetch 1 row at a time.

```
delimiter//
create procedure abc()
begin
   declare a int;
   declare b varchar(15);
   declare c int;
   declare d int;
   declare y int default 0;
   declare z int;
   declare c1 cursor from select * from emp;
   select count(*) into z from emp;
   open c1;
   while y<z do
      fetch c1 into a,b,c,d;
      /* Processing, e.g. set hra = c*0.4, etc */
      insert into tempp values(a,b);
      set y = y+1;
   end while;
   close c1;
end;//
delimiter;
```

```
# continue handler for Not found exception.
  delimiter//
  create procedure abc()
  begin
     declare a int;
     declare b varchar(15);
     declare c int;
     declare d int;
     declare y int default 0;
     declare cicursor for select * from emp;
     declare continue handler for not found set y=1;
```

Snehal Sawant

```
open c1;
cursor_c1_loop: loop
    fetch c1 into a, b, c, d;
    if y = 1 then
        leave cursor_c1_loop;
    end if;
    insert into tempp values(a, b);
end loop cursor_c1_loop;
close c1;
end; //
delimiter;
```

- not found is a cursor attribute, it returns a boolean True value if the last fetch was unsuccessful, and false value if the last fetch was successful.

```
delimiter //
create procedure abc()
begin
declare a varchar(15);
declare b int;
declare y int default 0;
declare c1 cursor for select ename, sal from emp;
declare continue handler for not found set y = 1;
open c1;
cursor_c1_loop: loop
    fetch c1 into a, b;
    if you y = 1 then
        leave cursor_c1_loop;
    end if;
    /* processing, e.g. set hra = b * 0.4, etc */
    insert into tempp values(b, a);
end loop cursor_c1_loop;
close c1;
end; //
delimiter;
```

Snehal Sawant

```
delimiter //
create procedure abcc()
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 wher deptno=1;
   declare ~~condition~~ handler for not found set y=1;
        continue
open c1;
cursor_c1_loop:loop
   fetch c1 into a,b,c,d;
   if y=1 then
      leave cursor_c1_loop;
   end if;

      insert into tempp values (c,d);
   end loop cursor_c1_loop;
   close c1;
end;//
delimiter;
```

To reset cursor :-

```
  closec1;
  opencl;
```

Snehal Sawant

```
delimiter//
create procedure abc()
begin.
        declare c1 cursor for select * from dept;
        declare c2 cursor for select * from emp;
        open c1;
        open c2;
end;//
```

- There is no upper ~~curr~~ limit on no of cursor that you can declare in one procedure.
- there is no upper limit on the number of cursors that you can open at a time.
(the only restriction would be the size of server RAM; it should be large enough to manage so much data).
- If the cursor are large, and the server RAM is insufficient, then MySQL will use virtual memory (but then it will be very slow).

• declare c1 cursor for select dname, ename from emp, dept where dept.deptno = emp.empno;