Day_10

mysql> select * from emp;

- ; is known delimiter
- it indicates end of command

mysql> delimiter .

mysql> select * from emp.

mysql> select sal*0.1 from emp.

mysql> delimiter *

mysql> select * from emp*

mysql> delimiter /

mysql> select sal/10 from emp/

mysql> delimiter //

mysql> select sal/10 from emp//

- changing the delimiter is known as Personalization

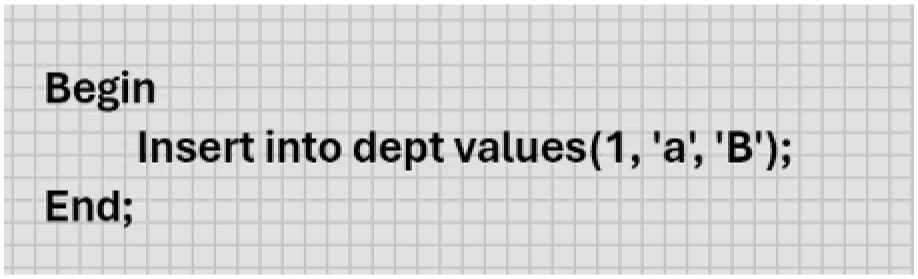
MySQL-PL

- MySQL Programming Language
- programming language from MySQL
- product of MySQL
- used for database programming
  - e.g. HRA_CALC, TAX_CALC, ATTENDANCE_CALC etc.
- used for server-side data processing(convert data into information)
- MySQL-PL program can be called in MySQL Command Line Client

MySQL Workbench, Java, Ms .Net, C++, etc; can be callled through any front-end s/w mysql> call hrc_call();

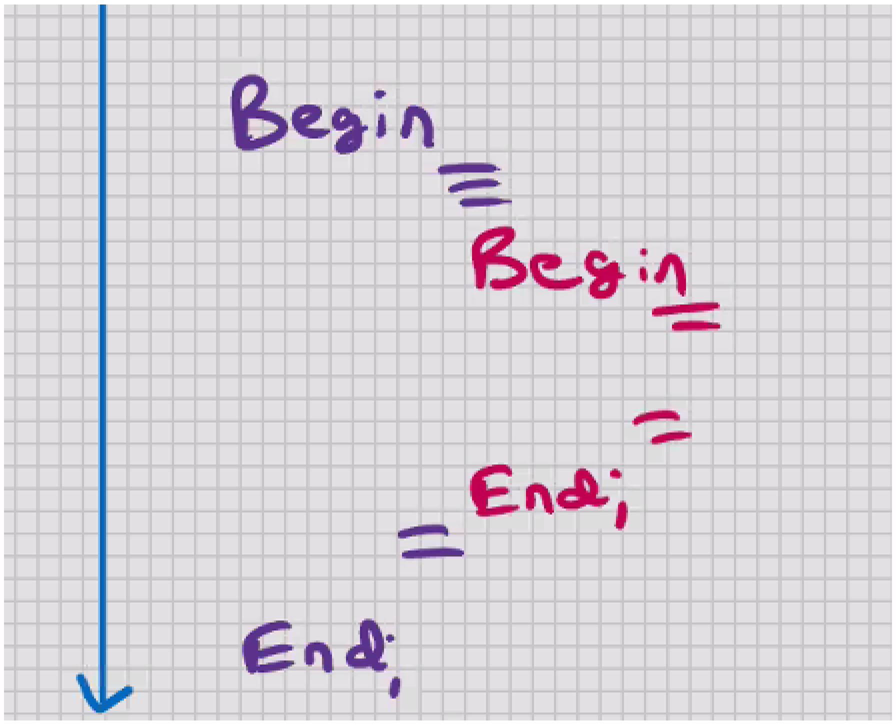
- Every RDBMS has it own native programming language:
    - Oracle(PL/SQL)→Procedural Language SQL(most popular language for commercial RDBMS)(63%)
    - MS SQL Server(T-SQL)→Transact SQL
    - MySQL(MySQL-PL)→MySQL Programming Language(most popular language for open-source RDBMS)(42%)

Typical PL-SQL program consists:



```
Begin
        Insert into dept values(1, 'a', 'B');
End;
```

- MySQL - PL program is commonly referred to MySQL-PL Block



- Block level Language (feature of oops)
- benefits of block level language:
    - Modularity
    - Control scope of variables (form of data hiding)(feature of OOPs)
    - Efficient error management with the help of exception
- Screen input and screen output is not allowed(scanf, printf, etc. not available)
- used ONLY for processing
- can use SELECT statement inside the block but its not recommended
- SQL commands that are allowed inside MySQL-PL block:
    - DDL, DML, DQL(not recommended), DTL/TCL

```
delete from emp
where deptno = (select deptno from emp where ename = 'Thomas');
```

- DCL commands are not allowed inside MySQL-PL program

To store output of MySQL-PL program:

```
create table tempp(
fir int,
sec char(15)
);
```

- MySQL-Pl programs are written in the form of stored procedures

---

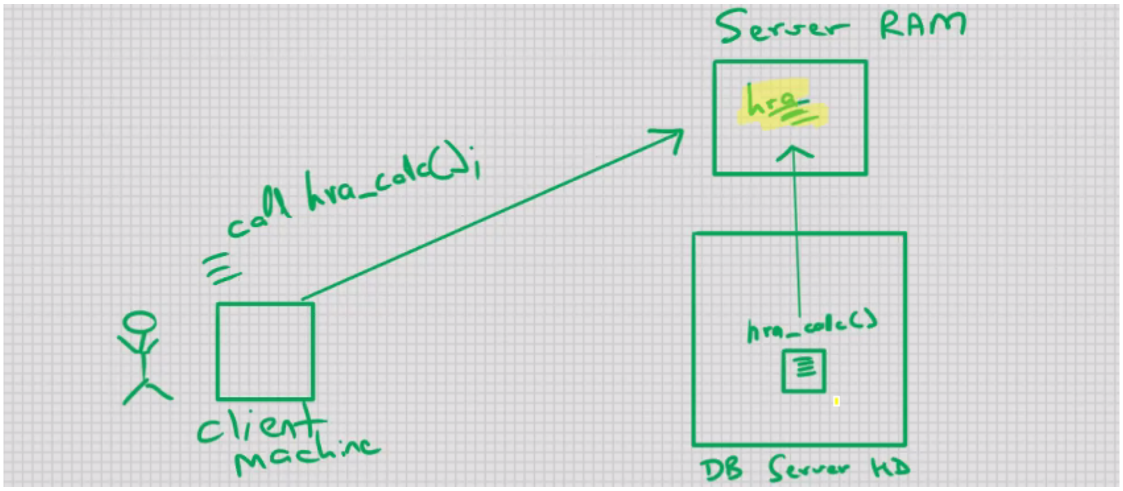
MySQL - STORED Objects

- objects that are stored in the database
- create ... .tables, indexes, views
- anything that you do with CREATE command is a stored object

---

STORED PROCEDURES

- Routine(set of commands) that has called explicitly
- global procedures
- stored in the database
- can be called through MySQL Command Line Client, MySQL Workbench, Java, MS .Net, etc; can be called through any front-end s/w
- stored in the database in the COMPILED FORMAT
- hence the execution is very fast, hiding the source code from end user
- execution takes place in SERVER RAM
- therefore ideally suited for Server-side data processing

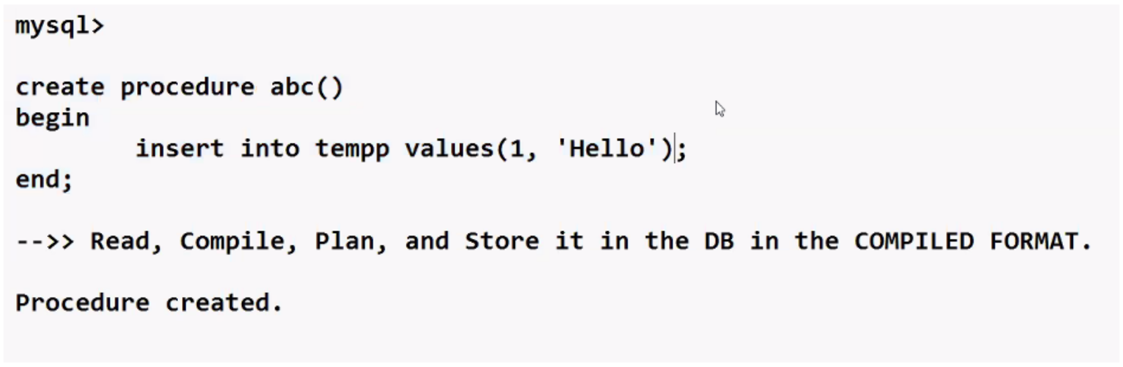


mysql> call hra_calc(); ← calling through CLC

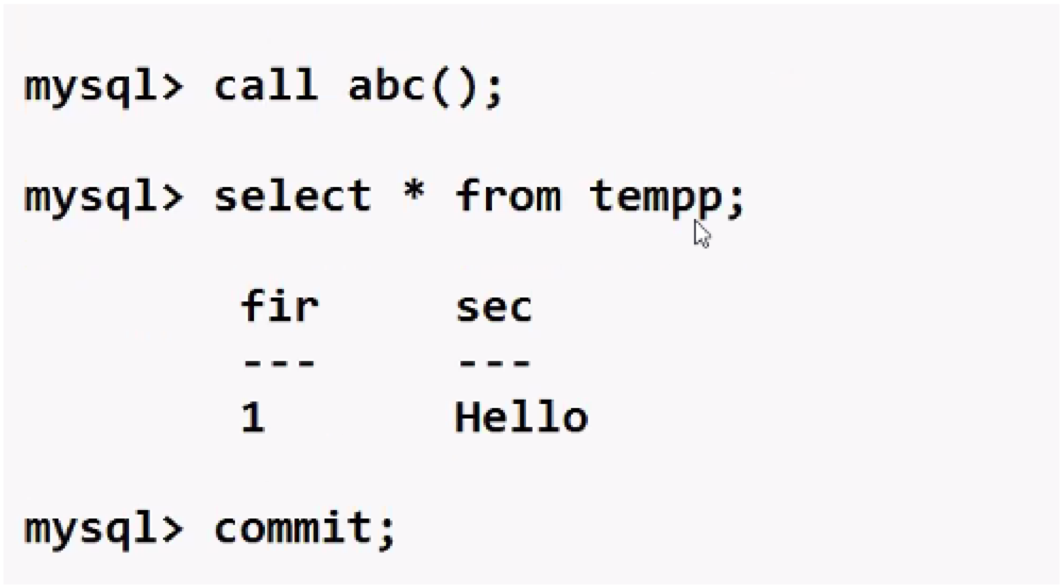
- Procedure can have LOCAL variables
- within the procedure you can have any processing, all MySQL-PL statements allowed, eg. IF Statement, loops, cursors, etc.
- one procedure can call another procedure
- procedure can call itself(known as Recursion)(e.g. self join)
- to make it flexible, you can pass parameters to a procedures

mysql> call hra_calc('KING', 5000, 0.4); ← calling through CLC

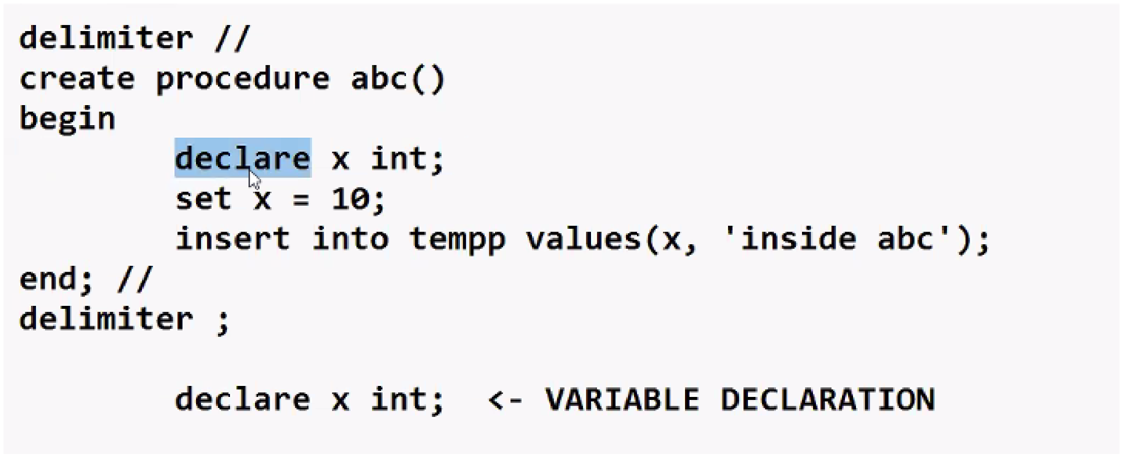
- OVERLOADING OF STORED PROCEDURES IS NOT SUPPORTED; you cannot create 2 or more procedures with the same name even if the NUMBER of parameters passed is different or the DATATYPE of parameters passed is different; because its a stored object



```
mysql>

create procedure abc()
begin
        insert into tempp values(1, 'Hello');
end;

-->> Read, Compile, Plan, and Store it in the DB in the COMPILED FORMAT.

Procedure created.
```

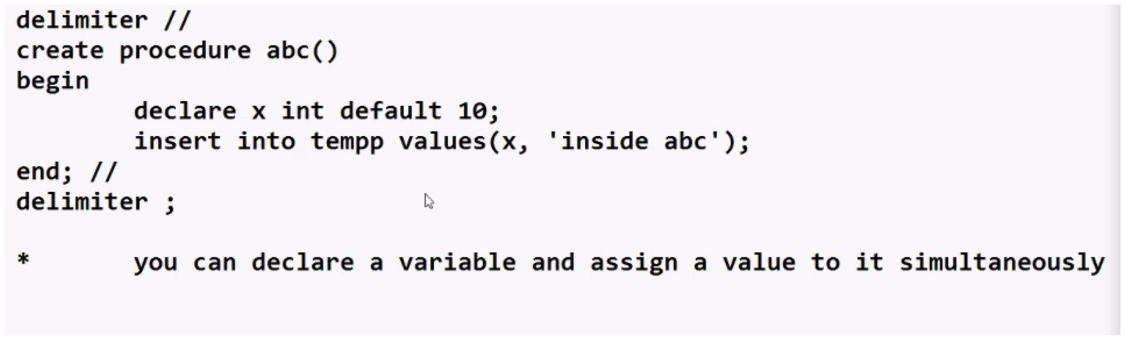


```
mysql> call abc();

mysql> select * from tempp;

          fir      sec
          ---      ---
          1        Hello

mysql> commit;
```

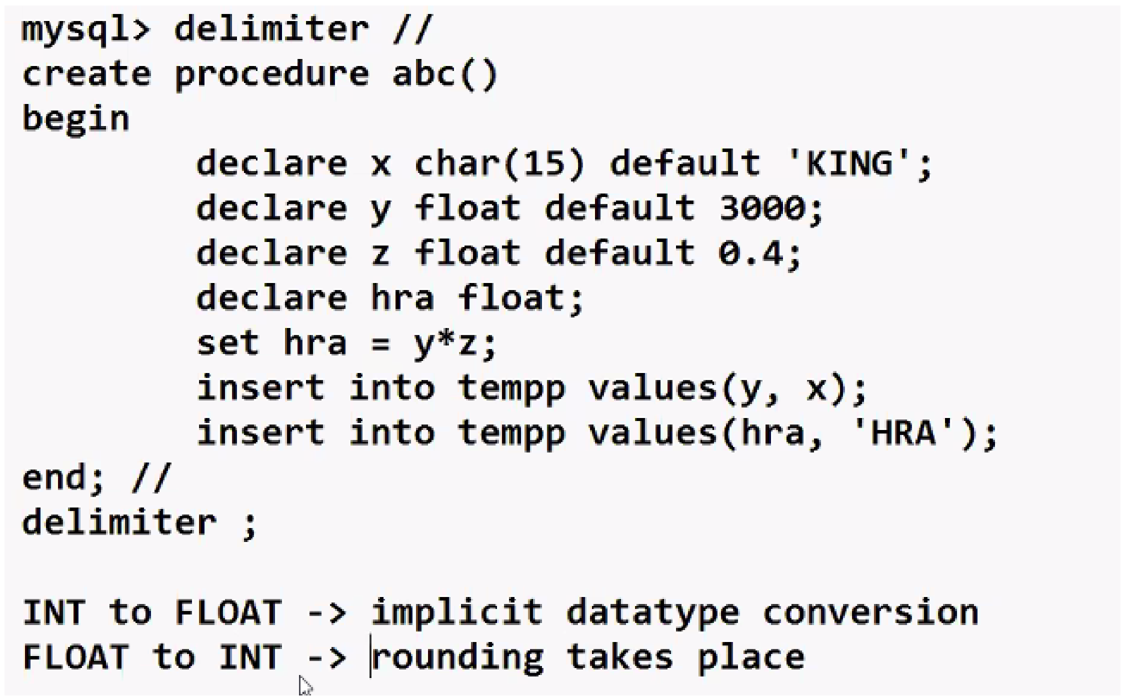


```
delimiter //
create procedure abc()
begin
        declare x int;
        set x = 10;
        insert into tempp values(x, 'inside abc');
end; //
delimiter ;

        declare x int;  <- VARIABLE DECLARATION
```

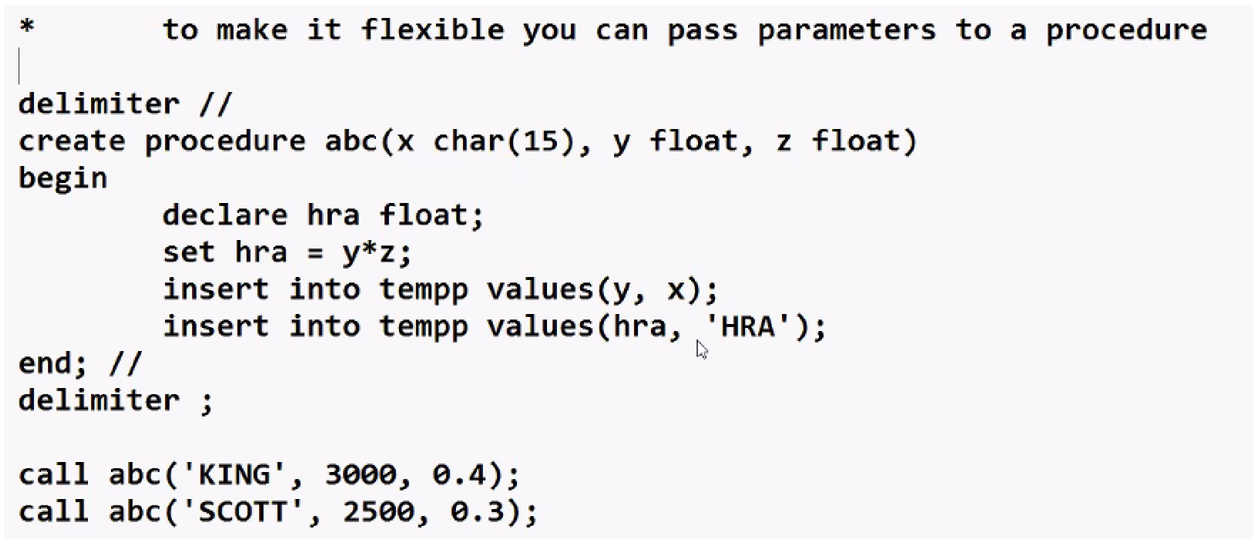
- Declare the variables at the top
- in MySQL; when you declare a variable, if you don’t initialize it; then it will store a null value.
- set x =10; ← ASSIGNMENT OPERATOR



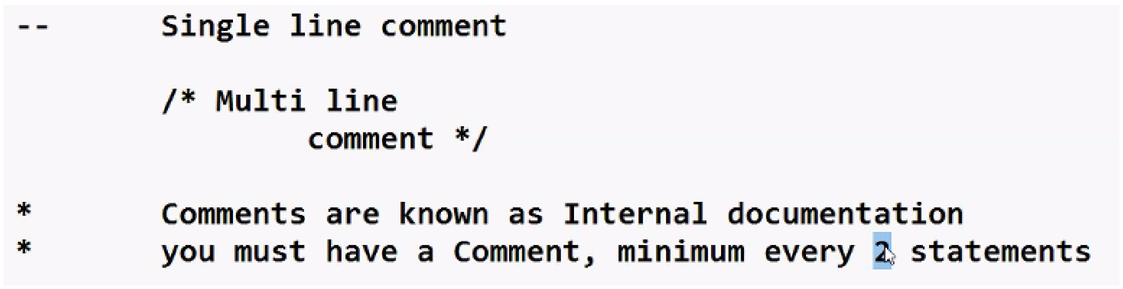
```
delimiter //
create procedure abc()
begin
        declare x int default 10;
        insert into tempp values(x, 'inside abc');
end; //
delimiter ;

*       you can declare a variable and assign a value to it simultaneously
```



```
mysql> delimiter //
create procedure abc()
begin
        declare x char(15) default 'KING';
        declare y float default 3000;
        declare z float default 0.4;
        declare hra float;
        set hra = y*z;
        insert into tempp values(y, x);
        insert into tempp values(hra, 'HRA');
end; //
delimiter ;

INT to FLOAT -> implicit datatype conversion
FLOAT to INT -> rounding takes place
```



```
*       to make it flexible you can pass parameters to a procedure

delimiter //
create procedure abc(x char(15), y float, z float)
begin
        declare hra float;
        set hra = y*z;
        insert into tempp values(y, x);
        insert into tempp values(hra, 'HRA');
end; //
delimiter ;

call abc('KING', 3000, 0.4);
call abc('SCOTT', 2500, 0.3);
```



```
--      Single line comment

        /* Multi line
                comment */

*       Comments are known as Internal documentation
*       you must have a Comment, minimum every 2 statements
```

To see which all procedures are created:

show procedures status; ← shows all procedures in all databases

show procedure status where db ='cdacmumbai';

show procedure status where name like 'a%';

To view the source code of stored procedure:

show create procedure abc;

To share the procedure with other users:

root_mysql> grant execute on procedure cdacmumbai.abc to rohit@localhost

for calling:

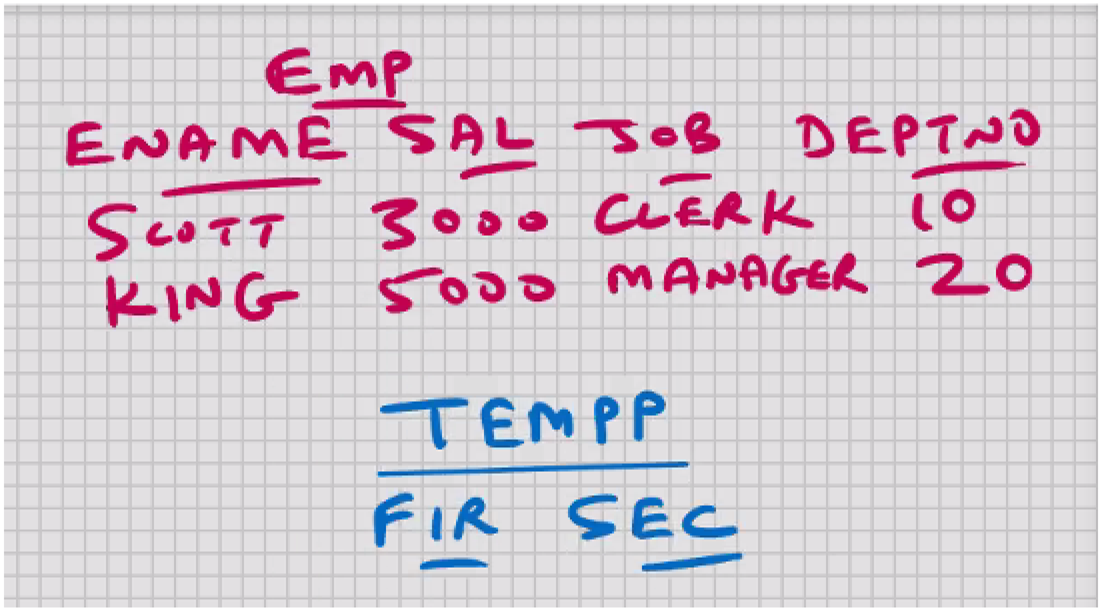
rohit_mysql> call cdacmumbai.abc();

root_mysql> revoke execute on procedure cdacmumbai.abc from rohit@localhost

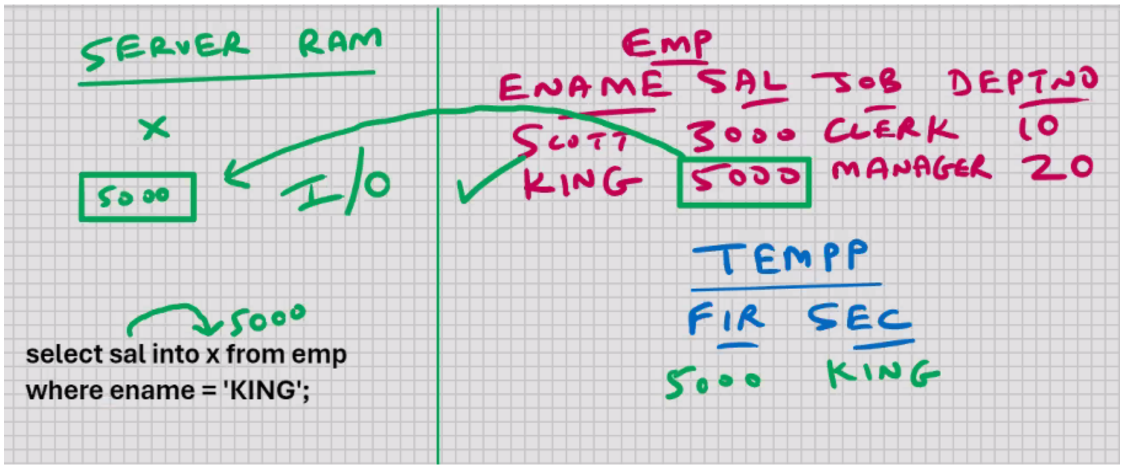
To change the source code:

drop procedure abc;

create procedure abc()



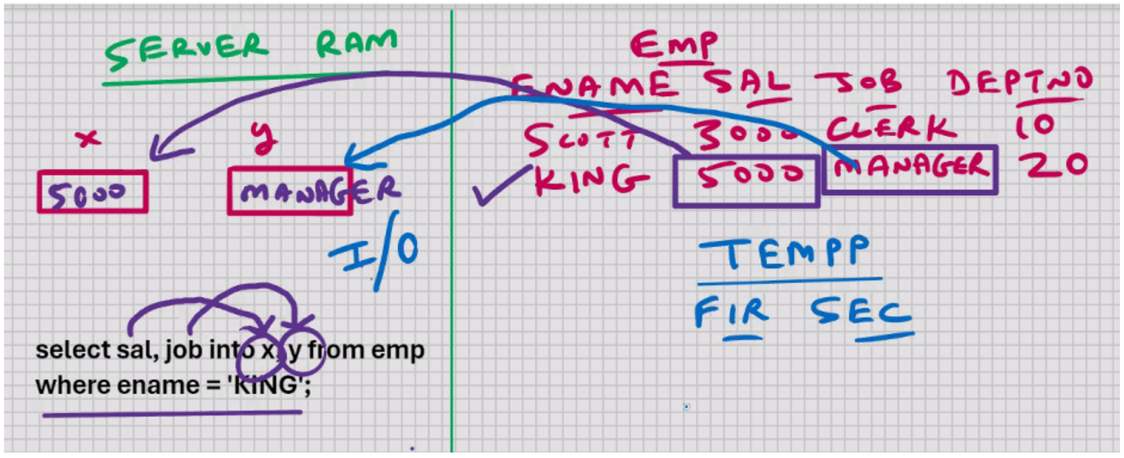
```
delimiter //
create procedure abc()
begin
        declare x int;
        select sal into x from emp
        where ename = 'KING';
        /* processing, e.g. set hra = x*0.4, etc. */
        insert into tempp values(x, 'KING');
end; //
delimiter ;

select <columnname> into <varname> from <table> where ...............;
```



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

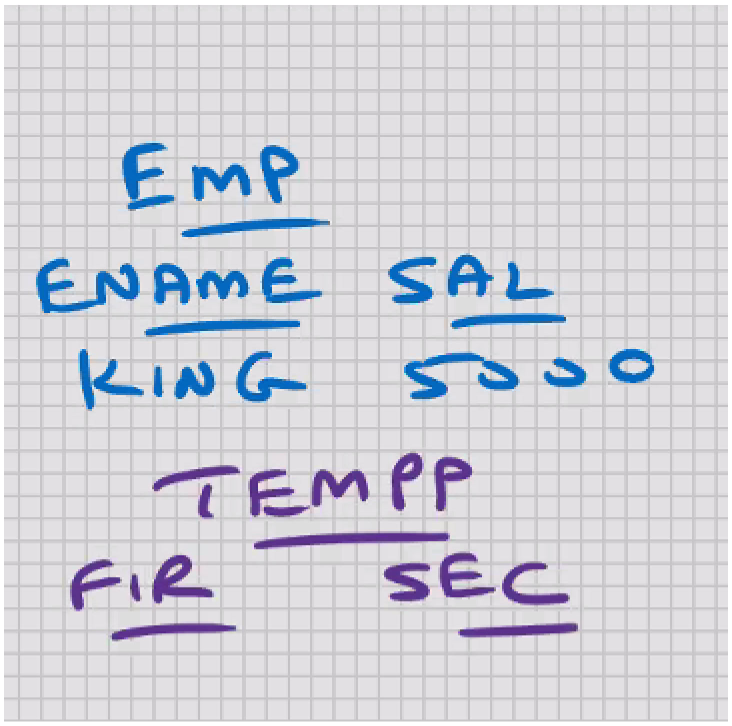
```
delimiter //
create procedure abc(y char(15))
begin
        declare x int;
        select sal into x from emp
        where ename = y;
        /* processing, e.g. set hra = x*0.4, etc. */
        insert into tempp values(x, y);
end; //
delimiter ;

call abc('KING');
call abc('SCOTT');
```

```
delimiter //
create procedure abc()
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, set y = lower(y), etc. */
        insert into tempp values(x, y);
end; //
delimiter ;

select <col1>, <col2>, ...., <coln> into <var1>, <var2>, ...., <varn>
from <table> where ............................;
```



```
select sal, job into x, y from emp
where ename = 'KING';
```

Decision making using IF statement:



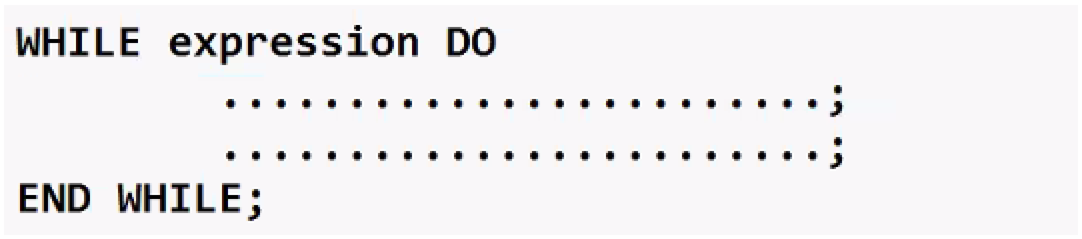
```
delimiter //
creatr procedure abc()
begin
        declare x int;
        select sal into x from emp where ename = 'KING';
        if x > 4000 and x < 5000 then
                insert into tempp values(x, 'High sal');
        end if;
end; //
delimiter ;

if x > 4000 then
        ...............................;
        ...............................;
        ...............................;
end if;
```

Loops

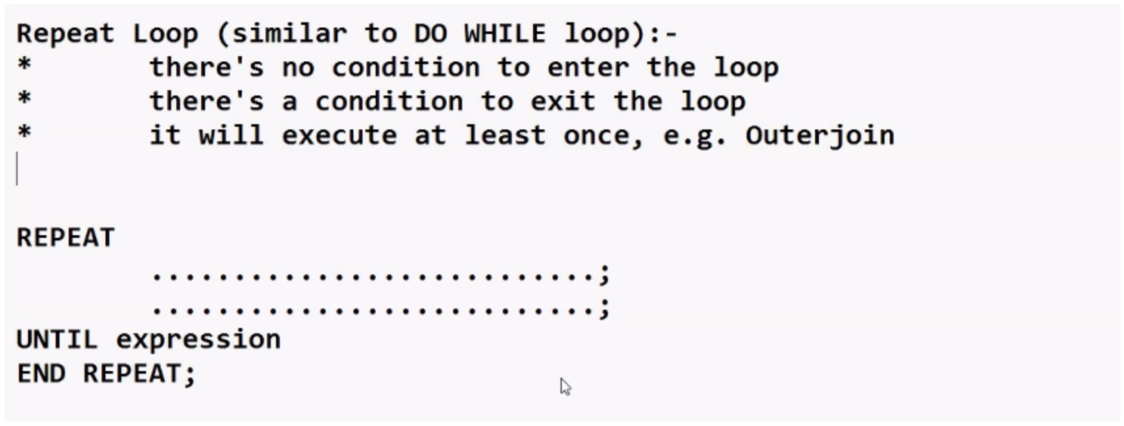
- for repetitive/iterative processing

While loop

- Check for some condition before entering the loop
- set x = x + 1; ← is very important



```
WHILE expression DO
        ........................;
        ........................;
END WHILE;
```



```
Repeat Loop (similar to DO WHILE loop):-
*       there's no condition to enter the loop
*       there's a condition to exit the loop
*       it will execute at least once, e.g. Outerjoin


REPEAT
        ........................;
        ........................;
UNTIL expression
END REPEAT;
```

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’ of ‘C’ programming)
- Loop statement executes a block of code repeatedly with an additional flexibility of using a loop label.

delimiter //

create procedure abc()

begin

    declare x int default 1;

```
    pqr_loop:loop
        if x>10 then
            leave pqr_loop
        end if;
        set x =x+1;
        if mod(x, 2) ≠ 0 then
            iterate pqr_loop;
        else
            insert into tempp values(x, ‘inside loop’);
        end if;
    end loop;
    end; //
delimiter ;
```

- in deeply nested loops, going from innermost loop to a point outside the outermost loop; leave would be the fastest way of doing it

---

MySQL - Global variables

mysql> set @x = 10; ←remains in the server RAM till you exit(end of session)

mysql> select @x from dual;←(output: 10)

- Global variables can be used in SELECT, INSERT, UPDATE, DELETE, statements and MySQL-PL programs also; can be used in front-end s/w also
- No datatype feature in global variable

mysql> set @x=@x+1; ← possible(output: 11)

---

char variable:

set @y=’CDAC’;

---

date variable:

set @z = ‘2024-10-18’;