CO-3

CO3 Topics

- PL/SQL
- Normalization
- Database File Organization
- Database File indexing
- Algorithms for PROJECT and Set Operations
- Implementing Aggregate Operations and OUTER JOINs


Oracle
PL/SQL

PL/SQL

- Introduction
- Features of PL/SQL
- PL/SQL Structure
- Programming constructs

Introduction to PL/SQL

- What is PL/SQL?
  - Developed by Oracle corporation in the 1980s.
  - It's a procedure language extension for SQL and Oracle relational database.

Features of PL/SQL

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types as SQL.
- It offers a variety of programming structures
- It supports object-oriented programming.
- It supports the development of web applications and server pages.
- It is not case sensitive.

PL/SQL structure

```
DECLARE
    <declarations section>
BEGIN
    <executable command(s)>
EXCEPTION
    <exception handling>
END;
```

The 'Hello World' Example

```
DECLARE
        message varchar2(20) := ‘Hello World!’;
BEGIN
        dbms_output.put_line(message);
END;
```

Output:

```
Hello World!

PL/SQL procedure successfully completed.
```

Programming structures

- Conditional statements
- Loops
- Procedures
- Functions
- Cursors
- Triggers

MySQL
Stored Procedures

MySQL Stored Procedures

Sid	Sname	Age	Dept	Year	Mobile	Address	CGPA

- **SELECT** Sid, Dept, Year, CGPA **From** students;
- MySQL processes the query and returns the result set.
- If you want to save this query on the database server for execution later, one way is to use a stored procedure.
- Once you save the stored procedure, you can invoke it by using the CALL statement:

MySQL Stored Procedures…

- A Stored procedure is a subroutine which contains a set of statements, stored in database.
- A procedure has a name, a parameter list, and SQL statement(s).
- It does not return a value.

Syntax to create procedure

CREATE PROCEDURE procedure_name [(parameter_type parameter datatype
[, parameter_type parameter datatype])]
BEGIN

declaration_section

executable_section

END;

Syntax for calling Procedure:

CALL procedure_name([parameters]);

Example

```
CREATE PROCEDURE GetStudents( )

BEGIN

  SELECT Sid, Dept, Year, CGPA FROM students;

END
```

```
CALL GetStudents( );
```

DROP Procedure

- Once you have created your procedure in MySQL, you might find that you need to remove it from the database.

- The syntax to a drop a procedure in MySQL is:

  DROP procedure [IF EXISTS] ***procedure_name*****;**

Steps to follow before creating a procedure

1. Select a database
2. Pick a Delimiter

Examples

- Simple procedure

```
delimiter @
create procedure add1()
begin
 declare a, b, c int;

 set a=10;
 set b=20;
 set c=a+b;
 select c;
end @
delimiter ;

call add();

drop procedure add1;
```

Examples

- Procedure with parameters

```
delimiter @
create procedure add2(in a int, in b int)
begin
 declare c int;

 set c=a+b;
 select c;
end @
delimiter ;

call add2(10,20);

drop procedure add2;
```

IF-THEN-ELSE Statement

IF-THEN-ELSE Statement

```
IF condition1 THEN
  {...statements to execute when condition1 is TRUE...}

[ ELSEIF condition2 THEN
  {...statements to execute when condition1 is FALSE and condition2 is TRUE...}]

[ ELSE
  {...statements to execute when both condition1 and condition2 are FALSE...}]

END IF;
```

Examples

- Simple IF-ELSE

```
drop procedure if exists set_b;
delimiter @
create procedure set_b(in a int, out b int)
begin

   declare c int;
   set c = 100;
   if a < 20 then
      set b = c + a;
   else
      set b = c - a;
   end if;

end @
delimiter ;

call set_b(30, @b);
select @b;
```

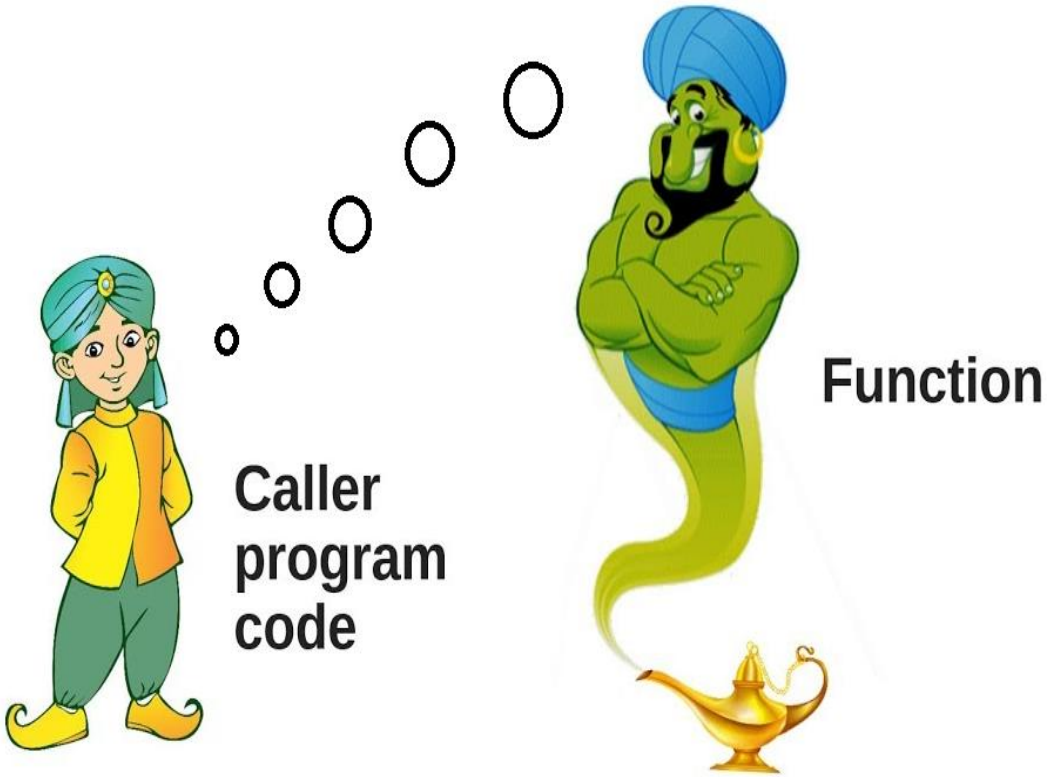
Examples

- IF-ELSEIF-ELSE

```
drop procedure if exists grade_result;
delimiter @
create procedure grade_result(in p int)
begin
   if p >= 90 then
      select 'Grade : X';
   elseif p >= 75 and p < 90 then
      select 'Grade : A';
   elseif p >= 60 and p < 75 then
      select 'Grade : B';
   elseif p >= 40 and p < 60 then
      select 'Grade : C';
   else
      select 'Grade : Fail';
   end if;
end @
delimiter ;

call grade_result(80);
```

MySQL Functions



MySQL Functions

- In MySQL, Functions can also be created.
- Similar to a procedure.
- A function always **returns a value** using the return statement.
- Function call is done using **SELECT.**

Function Syntax

Creating a function

```
CREATE FUNCTION function_name [ (parameter datatype
                                 [, parameter datatype]) ]
RETURNS return_datatype
BEGIN
        Declaration_section
        Executable_section
END;
```

Drop a function

```
DROP FUNCTION [ IF EXISTS ] function_name;
```

Example

- Simple function

```
drop function if exists grade_result;
delimiter @
create function grade_result(p int) returns varchar(10)
begin
   declare grade varchar(10);
   if p >= 90 then
      set grade = 'X';
   elseif p >= 75 and p < 90 then
      set grade = 'A';
   elseif p >= 60 and p < 75 then
      set grade = 'B';
   elseif p >= 40 and p < 60 then
      set grade = 'C';
   else
      set grade = 'Fail';
   end if;
   return grade;
end @
delimiter ;

select grade_result(80);
```

CASE Statements

CASE Statement

```
CASE [ expression ]
        WHEN condition_1 THEN result_1
        WHEN condition_2 THEN result_2   ...
        WHEN condition_n THEN result_n
        ELSE result
END
```

Note

- If no condition is found to be true, then the CASE function will return the value in the ELSE clause.
- If the ELSE clause is omitted and no condition is found to be true, then the CASE statement will return NULL.

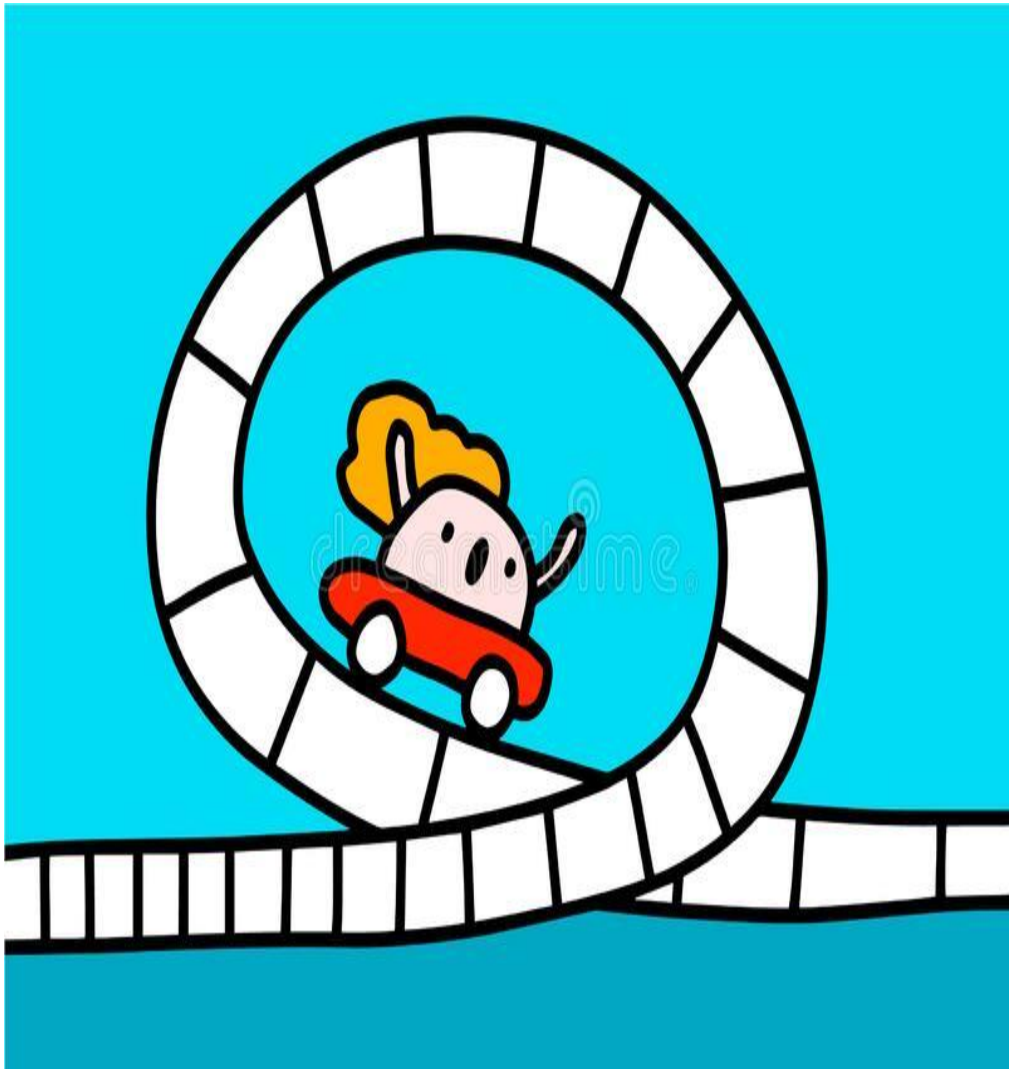
Example

- Simple CASE

```
drop function if exists grade;
delimiter @
create function grade(p int) returns varchar(10)
begin
   declare grade varchar(10);
   case
   when p >= 90 then
      set grade = 'X';
   when p >= 75 and p < 90 then
      set grade = 'A';
   when p >= 60 and p < 75 then
      set grade = 'B';
   when p >= 40 and p < 60 then
      set grade = 'C';
   else
      set grade = 'Fail';
   end case;
   return grade;
end @
delimiter ;

select grade(80);
```

LOOP Statements



MySQL Loops

- LOOP
  - allows you to execute one or more statements repeatedly.
- LEAVE
  - exits the flow control that has a given label.
- REPEAT
  - executes one or more statements until a search condition is true.
- WHILE
  - executes a block of code repeatedly as long as a condition is true.

Syntax of MySQL Loops

LOOP	LEAVE
[begin_label:] **LOOP** statement_list **END LOOP** [end_label]	**LEAVE** label;
REPEAT	**WHILE**
[begin_label:] **REPEAT** statement_list **UNTIL** search_condition **END REPEAT** [end_label]	[begin_label:] **WHILE** search_condition **DO** statement_list **END WHILE** [end_label]

Examples

- LOOP and LEAVE

```
drop procedure if exists mulofn;
delimiter @
create procedure mulofn(a int)
begin

   declare c,i int;
   set i=1;
   L1: loop
        set c = i * a;
        select a,i,c;
        set i = i + 1;
        if i<=10 then
           iterate L1;
        end if;
        leave L1;
        end loop L1;

   end @
delimiter ;

call mulofn(10);
```

Examples

- REPEAT

```
drop function if exists sumofn;
delimiter @
create function sumofn(n int) returns int
begin

   declare sum,i int;
   set sum = 0, i = 1;
   L1: repeat
        set sum = sum + i;
        set i = i + 1;
        until i>n
        end repeat L1;
   return sum;
   end @
delimiter ;

call mulofn(5);
```

Examples

- WHILE

```
drop procedure if exists mulofn;
delimiter @
create procedure mulofn(a int)
begin

   declare c,i int;
   set i=1;
   L1: while i<=10 do
        set c = i * a;
        select a,i,c;
        set i = i + 1;
   end while L1;

end @
delimiter ;

call mulofn(10);
```

Example

- To extract data from tables

```
drop function if exists grade;
delimiter @
create function grade(id int) returns varchar(10)
begin
   declare grade varchar(10);
   declare s1,s2,s3,sum int;
   declare p float;
   select m1 into s1 from students where sno = id;
   select m2 into s2 from students where sno = id;
   select m3 into s3 from students where sno = id;
   set sum = s1 + s2 + s3;
   set p = (sum / 90)*100;
   case
   when p >= 90 then
      set grade = 'X';
   when p >= 75 and p < 90 then
      set grade = 'A';
   when p >= 60 and p < 75 then
      set grade = 'B';
   when p >= 40 and p < 60 then
      set grade = 'C';
   else
      set grade = 'Fail';
   end case;
   return grade;
end @
delimiter ;
```


THANK
YOU