VALENTINE
Page No.:
Date: / /

Question Bank

* Descriptive and scenario based question

1. Differentiate between SQL and Procedural SQL.

Ans:

SQL	Procedural SQL
1. SQL is a single query that is used to perform DML and DDL operations.	1. Procedural SQL is a block of codes that used to write the entire program blocks/ procedure/function, etc.
2. It is declarative, that defines what needs to be done; rather than how things need to be done.	2. Procedural SQL is procedural that defines how the things needs to be done.
3. Execute as a single statement.	3. Execute as a whole block.
4. Mainly used to manipulate data.	4. Mainly used to create an application
5. Cannot contain procedural SQL code in it.	5. It is an extension of SQL, so it can contain SQL inside it.

VALENTINE
Page No.:
Date: / /

2. Explain structure of procedural block giving an example. (Pg-78)

Ans:

A block consists of various types of declarations (eg. variables, cursors, handlers) and program code (eg. assignments, conditional statements, loops). The order in which these can occur is a follow:

1. Variable and condition declarations.
~~Variables~~
2. Cursor declarations
3. Handler declarations
4. Program code.

⇒ If we violate this order - for instance, by issuing a declare statement after a set statement - MySQL will generate an error message when you try to create your stored program code.

⇒ The error messages do not always clearly indicate that you have used statements in the wrong order, so it's important to develop the habit of declaring things in the correct order.

⇒ you can also name a block with a label. The label can occur both

before the begin statement and after the end statement.

Example: [label:] Begin

variable and condition declarations
cursor declarations
handler declarations

program code

end [label1];

⇒ create procedure P_cursor()
Begin
declare v_sname varchar(30);
declare row_finished boolean default 0;

Declare cursor_sname cursor for select sname from salesman;
Declare continue handler for not found set row_finished=1;

Open cursor_sname;

label1: Loop

```
Fetch cursor_sname into v_sname;
select v_sname;

        If row_finished = 1 then
                leave label1;
        End if;
        End loop label1;
close cursor_sname;
end //
call p_cursor();
```

3. Advantages of stored procedural. [Pg 4]

Ans:

1. The used of stored programs can lead to a more secure database.
2. Stored programs offer a mechanism to abstract data access ma routines, which can improve the maintainability of your code as underlying data structures evolve.
3. Stored programs can reduce network traffic, because the program can work on the data from within the server, rather than having to transfer the data across the network.
4. Stored programs can be used to implement common routines accessible

from multiple applications - possibly using otherwise incompatible frameworks - executed either within or from outside the database server.

5. Database-centric logic can be isolated in stored program and implemented by programmers with more specialized, database experience.

6. The use of stored programs can, under some circumstances, improve the portability of your application.

4. What are the disadvantages of stored procedure? [Pg-264]

Ans:

1. Stored programs might be slower - especially for computationally expensive operations - than equivalent middle-tier code.

2. The use of stored program can lead to fragmentation of your application logic - logic may be split between the database and the application server tier, making it difficult to track down design flaws or implementation bugs.

3. Stored programs are usually written in a different language

from your application server tier, requiring a wider range of skills in your development team.

4. Stored program can be more difficult to debug.

5. Most object-relational mapping systems cannot seamlessly exploit stored programs.

6. While stored program calls may sometimes be more portable than native SQL, in practice this is not true for all implementations. Of the "big four", only DB2 and MySQL implement the ANSI standard for stored programs. As a result, MySQL stored programs calls often look and act substantially different from calls made in oracle or SQL Server.

5. In which situation else if ladder is used? Demonstrate with an example? [Eg. 8c]

Ans: ⇒ The full syntax of the if statement allows for multiple conditions to be defined.

⇒ The first condition that evaluates to true will execute. If none of the statements evaluates to true, the the else clause (if present) will execute.

⇒ You can have as many elseif conditions as you like.

⇒ The conditions do not need to be mutually exclusive. That is, more than one of the condition can evaluate to true. The first condition that evaluates to true is the one that executes.

⇒ ~~creating overlaps~~ Example:

```
If (sale_value > 200 and customer_status = 'Preferred') then
    call free_shipping(sale_id);
    call apply_discount(sale_id, 20);
elseif (sale_value > 200) then
    call free_shipping(sale_id);
end if;
```

Explain case and search case statement with their syntax and example? [Pg-89]

Ans: Case statement:

⇒ case statement - compares the output of an expression with multiple conditions.

⇒ Syntax:

```
case expression
    when value then
        statements
    [when value then statements...]
    [else statements]
end case;
```

⇒ This syntax is useful when we are checking the output of some expression against a set of distinct values.

Example:

```
set CH = 2;
case CH
    when 1 then
        set result = N1 + N2;
    when 2 then
        set result = N1 - N2;
end case;
```

⇒ The simple case statement is useful when comparing the value of an expression to a series of specific values.

⇒ However, the simple case statement cannot easily or naturally match ranges, or handle more complex conditions involving multiple expressions.

Searched case statement.

⇒ The searched case statement is functionally equivalent to an if-elseif-else-end if block.

⇒ Syntax:

```
case
    when condition then
        statements
    [when condition then statements...]
    [else statements]
end case;
```

⇒ Using the search case structure we can solve following example:

Example:

```
case
    when (CH=1) then
        set result = N1+N2;
```

VALENTINE
Page No.:
Date: / /

```
when (CH=2) then
    set result = N1-N2;
end case;
```

6. List four types of if statement. Explain any one statement with syntax and example. [82 Pg]

Ans: ① If statement
② if - then
③ if - then - else
④ if - then - elseif - else

⇒ If - then - else statements

- Adding an else condition to your if statements allow you to specify statements that will execute if the If condition is not true. We will ~~so~~ emphasize again - because it is important - that not true does not always mean false.

⇒ If the If statement condition evaluates to null, then the else statements will still be executed; this can lead to subtle bugs if you don't protect against null variables in your if conditions.

Syntax:

```
If <expression> then
    <statements that execute if the
    expression is true>
else
    <statements that execute if the
    expression is false or null>
end if;
```

Example:

```
If sales_value < 200 then
    call apply_shipping;
else
    call apply_discount;
end if;
```

7. Explain nested if with an appropriate example.

Ans:

⇒ A nested if is an if statement that is the target of another if statement. Nested if-then statements mean an if statement inside another if statement.

⇒ Syntax:

```
if (condition1) then
    executes when condition1
        is true
```

```
    if (condition2) then
        executes when condition2
            is true
    end if;
end if;
```

Example:

```
if (n1 > n2) then
    if (n1 > n3) then
        select n1 is greater;
    else
        n3 is greater;
else
    if (n2 > n3) then
    select n2 is greater;
    else
    select n3 is greater;
end if;
end if;
```

8. Explain while-Do loop and repeat-until loop giving examples.

Ans: ① While-Do loop:

⇒ A while-Do loop execute as long as condition is true. If the condition is not true to begin with, then the loop will never

execute.

⇒ Syntax:

```
while (expression) do
    statement 1;
    [statement-2]...
end while;
```

Example:

```
while i<4 do
  set s= concat(s'i=', i, 'l');
  set i= i+1;
end while;

select s as message;
```

② Repeat... until loop

⇒ The repeat and until statements can be used to create a loop that continues until some logical condition is met.

⇒ Syntax:

```
[label:] repeat
    statements
  until expression
end repeat [label]
```

Example:
```
repeat
    select 2*n;
    set n = n+1;
    until n=11;
end repeat;
```

9. What is the use of case statement? List its type.

Ans: The case expression can be used anywhere scalar expression are allowed, including in where and having clauses of the select statement.

⇒ In this article, I would like to show the most commonly used case expression in:

- stored procedure
- Formula of a particular column
- View.

⇒ Types

① Simple case
② Searched case.

Long questions

12. Differentiate between leave and iterate statements. Explain giving examples.

Ans: 12

Leave	Iterate
• The Leave statement allows us to terminate loop.	• The iterate statement is used to restart execution at the beginning of a loop.
• The general syntax for the leave statement is Leave label;	• The general syntax for iterate statement is iterate label;
• Eg: set i=1; myloop: loop set i=i+1; if i=10 then leave myloop; end if;	• Eg: set i=0; Loop1: Loop set i=i+1; if i≥10 then leave loop1; elseif mod(i,2)=0 then

```
end loop myloop;
select 'I ran
count to 10';
```

```
iterate loop1;
end if;
select concat
(i,"is an
odd number");
end loop loop1;
```

16. Write the syntax for user defined function and explain each clause.

Ans: Syntax for user defined

```
create function function_name
(parameter[,...])
return datatype
function_statements
```

- The returns clause is mandatory and defines the data type that the function will return.
- You cannot specify the in, out or inout modifiers to parameters. All parameters are implicity in parameters

signature

DATE: / /

- The function body must contain one or more return statements, which terminate function execution and return the specified result to the calling program, as described.

18. How procedure is called? Explain parameterize procedure with example.

Ans: Calling one stored program from another is perfectly simple.

⇒ You do this with the call statement, just as you would from the MySQL command-line client.

⇒ Parameterize in SQL:

① IN

This mode is the default. It indicates that the parameter can be passed into the stored program but that any modification are not returned to the calling program.

② OUT

⇒ This mode means that the stored program can assign a value to the parameter, and that value will passed back to the calling program.

③ INOUT

⇒ This mode means that the stored program can read the parameter. and that the calling program can see any modifications that the stored program may make to that parameter.

Eg:
```
Deliniter ;
drop procedure if exists
my_sqrt
create procedure my_sqrt
(input_number int, out out_
number float)
Begin
    set out_number =
        sqrt(input_number);
End //
Deliniter ;
```

signature

DATE : / /

23. What are the Codd's 12 Rule? Explain them in details.

Ans: Dr. Edgar Frank - "Ted" Codd - A computer scientist working for IBM proposed the relational model for database management which forms the theoretical basis for rational database.

⇒ According, if a database has to be called as true relational database management system, then it has to follow all these rules.

0. Base rule

⇒ "The system has to qualify as a relational, a database and a management system".

• A system to qualify as a relational DBMS, must utilize the facilities to manage database.

1. Information rule.

⇒ "All the information including metadata (data about data)

has to be represented as stored data in cells of tables."

- Also says that the rows and columns have to be strictly unordered.
- All the data must be in a table format.

2. Guaranteed access rule.

⇒ "Each unique piece of data should be accessible by the combination of table name, primary key and attribute."

- It simply means that all the data has to be accessible.
- Accessed by: table name + primary key + attribute and not by other means like pointers.

3. Systematic treatment of null values

⇒ "RDBMS should be capable of allowing each attribute

signature

DATE: / /

to remain as null, should also support the representation of missing information and inapplicable information."

- Has to be handled consistently, cannot be zero or blank.
- Primary key cannot be null.
- Null can be missing data, data that is not applicable or also no data.
- Can used when data does not exist or when data should not be used for the case