Summary – Day 14

Exception Handling:

- When an error occurs inside a stored procedure, it is important to handle it appropriately, such as continuing or exiting the current code block's execution, and issuing a meaningful error message.

- MySQL provides an easy way to define handlers that handle from general conditions such as warnings or exceptions to specific conditions e.g., specific error codes.

- To declare a handler, you use the DECLARE HANDLER statement as follows.

```
DECLARE action HANDLER FOR condition_value statement;
```

- If a condition whose value matches the condition_value , MySQL will execute the statement and continue or exit the current code block based on the action.

- The action accepts one of the following values:
  - CONTINUE : the execution of the enclosing code block (BEGIN … END) continues.
  - EXIT : the execution of the enclosing code block, where the handler is declared, terminates.

- The condition_value specifies a particular condition or a class of conditions that activate the handler. The condition_value accepts one of the following values:
  - A MySQL error code.

    - A standard SQLSTATE value. Or it can be an SQLWARNING , NOTFOUND or SQLEXCEPTION condition, which is shorthand for the class of SQLSTATE values. The NOTFOUND condition is used for a cursor or SELECT INTO variable list statement.
    - A named condition associated with either a MySQL error code or SQLSTATE value.

- The statement could be a simple statement or a compound statement enclosing by the BEGIN and END keywords.

Loops:

- The MySQL LOOP statement could be used to run a block of code or set of statements, again and again, depending on the condition. Stored procedures are a subset of SQL statements that are kept in the SQL catalog as subroutines. These procedures may have both IN and OUT parameters. If you use SELECT statements, they might return result sets or multiple result sets. In MYSQL, functions can also be made.

- IF, CASE, ITERATE, LEAVE LOOP, WHILE, and REPEAT are examples of flow control statements supported by MySQL, much like in other programming languages. These statements can be used in stored programs (procedures), and stored functions can use RETURN. One Flow Control Statement may be used inside another.

LOOP:

- The LOOP can have optional labels at the beginning and end of the block.

- The LOOP executes the statement_list repeatedly. The statement_list may have one or more statements, each terminated by a semicolon (;) statement delimiter.

- Typically, you terminate the loop when a condition is satisfied by using the LEAVE statement.
- Syntax of the LOOP statement with LEAVE Statement.

```
[label]: LOOP
    ...
    -- terminate the loop
    IF condition THEN
        LEAVE [label];
    END IF;
    ...
END LOOP;
```

REPEAT:

- The REPEAT statement executes one or more statements until a search condition is true.
- Syntax:

```
[begin_label:] REPEAT
    statement
UNTIL search_condition
END REPEAT [end_label]
```

- The REPEAT executes the statement until the search_condition evaluates to true.
- The REPEAT checks the search_condition after the execution of statement, therefore, the statement always executes at least once. This is why the REPEAT is also known as a post-test loop.
- The REPEAT statement can have labels at the beginning and at the end. These labels are optional.

WHILE:

- The WHILE loop is a loop statement that executes a block of code repeatedly as long as a condition is true.
- Syntax:

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

- In this syntax:
  - First, specify a search condition after the WHILE keyword.
  - The WHILE checks the search_condition at the beginning of each iteration.
  - If the search_condition evaluates to TRUE, the WHILE executes the statement_list as long as the search_condition is TRUE.
  - The WHILE loop is called a pretest loop because it checks the search_condition before the statement_list executes.
  - Second, specify one or more statements that will execute between the DO and END WHILE keywords.
  - Third, specify optional labels for the WHILE statement at the beginning and end of the loop construct.

Cursors:

- Cursor in My SQL is used to iterate through a result set returned by a select statement.
- To handle a result set inside a stored procedure, we use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row accordingly.

Steps:

1. DECLARE cursor_name CURSOR FOR SELECT_statement;
2. DECLARE CONTINUE HANDLER FOR NOT FOUND // termination statement
3. OPEN cursor_name;
4. FETCH cursor_name INTO variables_list;
5. CLOSE cursor_name;

Features of a MySQL Cursor:

- A cursor is read-only and cannot update or remove data in the result set from the procedure.

- A cursor needs to be declared before it can be used. The cursor definition is only a step to tell MySQL that such a cursor exists and does not retrieve and data.
- You can only retrieve data in the order specified by the select statement and not in any reverse order, commonly known as non-scrollable.
- You use a cursor by opening it and then perform fetch operations on the data stored.
- You must close a cursor after the fetch operations complete.