Explicit Cursors

By Rahul Barve

Explicit Cursors

- Must be created when you are executing a `SELECT` statement that returns more than one row.
- Even though the cursor stores multiple records, only one record can be processed at a time, which is called as a current row.
- When you fetch a row, the current row position moves to the next row.

Explicit Cursors

- An explicit cursor is defined in the declaration section of the PL/SQL Block.
- It is created on a `SELECT` Statement which returns more than one row.

Explicit Cursors

- Syntax:

  CURSOR <cursor-name> IS

  <SELECT statement>

Steps for Explicit Cursors

By Rahul Barve

Steps for Explicit Cursors

- Declare the cursor in the declaration section.
- Open the cursor in the execution section.
- Fetch the data from cursor into PL/SQL variables or records in the execution section.
- Close the cursor in the execution section before the end of PL/SQL Block.

Using %ROWTYPE

By Rahul Barve

Using %ROWTYPE

- A special attribute used in case of cursors to declare a variable that will hold an entire row fetched from the cursor.

Stored Procedures

By Rahul Barve

Stored Procedures

- A procedure is a subroutine that is used to perform a specific operation on the data available in the table.
- A procedure has a header and a body.

Stored Procedures

- The header consists of the name of the procedure and the parameters or variables if any, passed to the procedure.
- The body consists of a declaration section, execution section and exception section similar to a general PL/SQL Block.

Procedure Parameters

By Rahul Barve

Procedure Parameters

- A procedure can accept 3 types of parameters
  - `IN (Default)`
  - `OUT`
  - `IN OUT`

Stored Procedures

- Syntax:

```
CREATE [OR REPLACE] PROCEDURE proc_name
[([list of parameters])]
AS
[Declaration section ]
BEGIN
    Execution section
[ EXCEPTION
    Exception section]
END;
```

Bind Variables

By Rahul Barve

Bind Variables

- When a procedure accepts an OUT parameter, then while executing the procedure from the SQL prompt, there has to be some variable available to catch the value processed by that procedure.
- Such a variable is called as BIND variable.

Bind Variables

- Creating bind variables:
  - `variable <var-name> <datatype>;`

Bind Variables

- Assigning values to bind variables:
  - `exec :<var-name> := <value>`