**Chapter 9: PROCEDURAL LANGUAGE SQL**

**Basics**

* One of the **three** **Oracle** embedded languages.
* PL/SQL is **portable**, **high-performing** and **OS independent**.
* Can be called from **cmd**.

**Features**

* Integrated with SQL.
* Error debugging feature.
* Many data type.
* Supports functions.
* Supports OOP.
* Can make web applications and servers.

**Advantages**

* Supports both static & dynamic SQL.
* Static supports DML & dynamic supports DDL.
* Error debugging feature.
* High performance.

**Syntax Format**

* DECLARE
* BEGIN
* EXCEPTION **(\*optional)**
* END

**Basic Syntax**

***DECLARE***

***message varchar2(20):= ‘Hello, World!’;***

***BEGIN***

***dbms\_output.put\_line(message);***

***END;***

***--single line comment***

***/\* multiline comment \*/***

**About Syntax**

* Identifier must **not** exceed **30** characters.
* Reserved words are also **identifiers**.
* Storage location is allotted as per variable name.
* Some data types include **integer, real, number** etc.
* If a **DECLARE** block is declared before **BEGIN** block, then it contains global variables.
* And **DECLARE** block after **BEGIN** block contains local variable.
* You **can write SQL commands** in PL/SQL.

**Value Output**

***dbms\_output.put\_line(‘Value of x is: ’ || x);***

**Cursor**

* A cursor is used for selecting rows.
* Consists of 2 types:
  + **Implicit:** Automatic and predefined.
  + **Explicit:** User defined and called manually.

**Cursor Keywords**

* %FOUND
* %NOTFOUND
* %ISOPEN
* %ROWCOUNT

**Cursor Example**

***IF sql%FOUND THEN***

***--codes***

***ELSEIF sql%NOTFOUND THEN***

***--more codes***

***ENDIF;***

**Creating Explicit Cursor**

***CURSOR cursor\_name IS select\_statement;***

***OPEN cursor\_name; --opening it***

***CLOSE cursor\_name;***

***FETCH name,id; --to fetch***

***EXIT WHEN c%NOTFOUND --to exit while searching***

**Loops**

***LOOP***

***--codes***

***END LOOP;***

**Triggers**

* Programs that execute automatically.
* **Advantages:-**
  + \*write literally anything\*

**Trigger Syntax**

***CREATE TRIGGER trigger\_name;***

***REPLACE TRIGGER trigger\_name;***

***--BEFORE/ AFTER/ WHEN/ INSTEAD OF for setting trigger condition***

**Stored Procedure**

* Function
* **Contains two parts:** Header & Body

**Syntax for Stored Procedure**

***CREATE PROCEDURE proc\_name***

***AS***

***--codes (add BEGIN, END etc)***

***GO***

***DROP PROCEDURE proc\_name***

**About Stored Procedured**

* **RETURN** statement **does exist** in PL/SQL stored procedures.