create table department(dno varchar2(10), total\_sal integer default 100)

create table employee(eid varchar2(10), dno varchar2(10), sal integer )

CREATE OR REPLACE TRIGGER display\_salary\_changes

BEFORE DELETE OR INSERT OR UPDATE ON employee

FOR EACH ROW

WHEN (NEW.sal> 0)

DECLARE

sal\_diff number;

BEGIN

sal\_diff := :NEW.sal - :OLD.sal;

dbms\_output.put\_line('Old salary: ' || :OLD.sal);

dbms\_output.put\_line('New salary: ' || :NEW.sal);

dbms\_output.put\_line('Salary difference: ' || sal\_diff);

END;

create or replace trigger dept\_sal

after insert ON employee

for each row

when(new.sal>0)

begin

insert into department values(:new.dno,:new.sal);

END;

create or replace PROCEDURE raise\_salary (

emp\_id varchar2, amount integer

) IS

BEGIN

UPDATE employee

SET sal = sal + amount

WHERE eid = emp\_id;

END raise\_salary;

BEGIN

raise\_salary('1', 200);

END;

CREATE OR REPLACE FUNCTION totalemp

RETURN number IS

total number(2) := 0;

BEGIN

SELECT count(\*) into total

FROM employee;

RETURN total;

END;

DECLARE

c number(2);

BEGIN

c := totalemp();

dbms\_output.put\_line('Total no. of Employees: ' || c);

END;

**CURSORS**

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

* Implicit cursors
* Explicit cursors

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK\_ROWCOUNT** and **%BULK\_EXCEPTIONS**, designed for use with the **FORALL** statement. The following table provides the description of the most used attributes −

|  |  |
| --- | --- |
| **S.No** | **Attribute & Description** |
| 1 | **%FOUND**  Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE. |
| 2 | **%NOTFOUND**  The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE. |
| 3 | **%ISOPEN**  Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement. |
| 4 | **%ROWCOUNT**  Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement. |

Any SQL cursor attribute will be accessed as **sql%attribute\_name**

DECLARE

total\_rows number(2);

BEGIN

UPDATE customers

SET salary = salary + 500;

IF sql%notfound THEN

dbms\_output.put\_line('no customers selected');

ELSIF sql%found THEN

total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' customers selected ');

END IF;

END;

/

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is –

*CURSOR cursor\_name IS select\_statement;*

Working with an explicit cursor includes the following steps −

* Declaring the cursor for initializing the memory
* Opening the cursor for allocating the memory
* Fetching the cursor for retrieving the data
* Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example –

*CURSOR c\_customers IS*

*SELECT id, name, address FROM customers;*

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it.

*OPEN c\_customers;*

Fetching the Cursor

Fetching the cursor involves accessing one row at a time.

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

Closing the Cursor

Closing the cursor means releasing the allocated memory.

*CLOSE c\_customers;*

DECLARE

e\_id employee.eid%type;

e\_dept employee.dno%type;

e\_sal employee.sal%type;

CURSOR e\_employees is

SELECT eid,dno,sal FROM employee;

BEGIN

OPEN e\_employees;

LOOP

FETCH e\_employees into e\_id, e\_dept,e\_sal;

EXIT WHEN e\_employees%notfound;

dbms\_output.put\_line(e\_id || '' || e\_dept || '' || e\_sal);

END LOOP;

CLOSE e\_employees;

END;