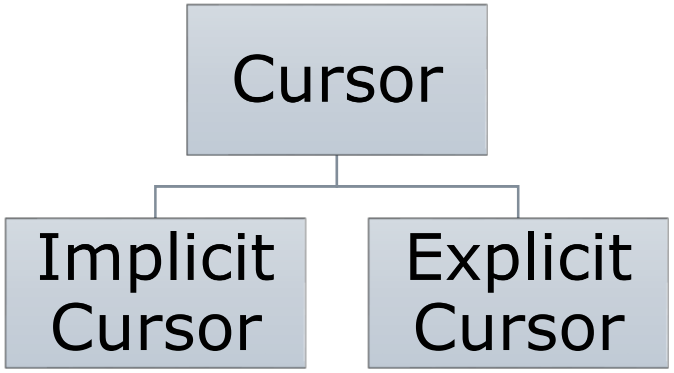
Datenbank und Informationssysteme

DI. Hilbe Klaus, MBA

Cursor



- Cursor is a pointer to a memory (context area)
- Holds information about Select or DML Statements
  - Rows returned by the query
  - Number of processed rows
  - Points to the parsed query in the shared pool

Implicit Cursor

- Implicit Cursor
  - Automatically created for every SQL statement (Select, Update,…) as long as no explicit Cursor is created for that query
  - Can‘t be created/controlled by the developer
  - Holds information of the most recent executed SQL statement (where no explicit Cursor was defined)
  - Also called the SQL-Cursor as the name of the implicit Cursor is “**SQL**”
- Explicit Cursor
  - Defined by the developer
  - Developer has full control of the Cursor
  - Must be used for Select – statements which return more than one row (remember with the Select… INTO… statement you needed to assure that only one row was returned)

Attributes of the Cursor

Attribute	Description
%FOUND	Returns TRUE if the SQL statement returned/effected one or more rows
%NOTFOUND	Opposite of %FOUND, returns TRUE if the SQL statement returned/effected zero rows
%ISOPEN	Returns TRUE if the cursor is open. Returns always FALSE for implicit cursors as Oracle closes the cursor automatically after the SQL Statement is executed
%ROWCOUNT	Returns the number of returned/effected rows of the SQL statement

Cursor
Example – Implicit Cursor

```
Set serveroutput ON;
DECLARE
    v_effected_rows    INTEGER := 0;
BEGIN
    UPDATE employees
    SET salary = salary + 10
    WHERE last_name = 'Kochhar';
    IF SQL%NOTFOUND THEN
      dbms_output.Put_line('No employees updated');
    ELSIF SQL%FOUND THEN
      v_effected_rows := SQL%ROWCOUNT;
      dbms_output.Put_line(v_effected_rows
        || ' employees updated');
    END IF;
END;
```

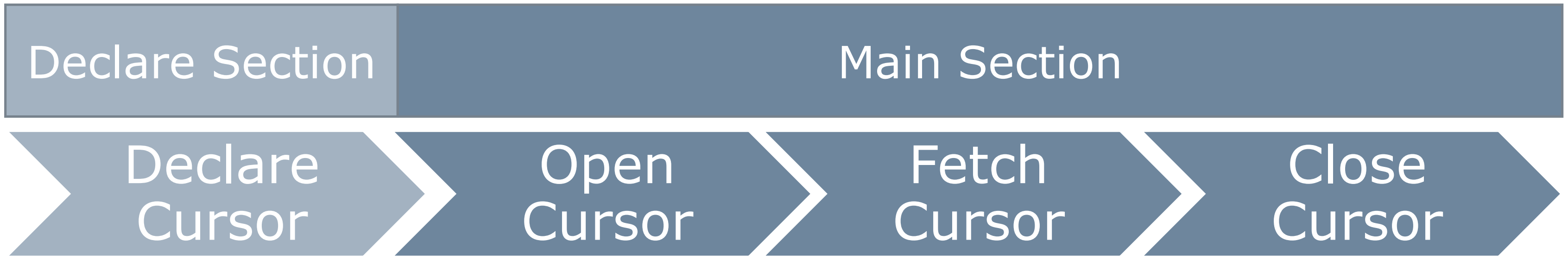
"SQL" is the name of the implicit Cursor

Cursor
Why do we need a explicit Cursor

```
Set serveroutput ON;
DECLARE
    v_last_name employees.last_name%TYPE;
BEGIN
    SELECT last_name
    INTO   v_last_name
    FROM   employees WHERE  salary > 15000;
    dbms_output.Put_line(v_last_name
       || ' earns a lot');
END;
```

Error report:
ORA-01422: exact fetch returns more than requested number of rows

Cusor
Steps when using a Cursor



- Declaring the cursor defines the name for the cursor and the associated SQL/DML statement
- Opening the cursor and allocates the memory for it
- Fetching the cursor involves retrieving the data row by row
- Closing the cursor to release the allocated memory

Explicit Cursor – Syntax

```
DECLARE
        …
        CURSOR Cursorname IS Select Statement;*
        …
BEGIN
        …
        OPEN Cursorname;
         …
        FETCH Cursorname INTO variablename1,
            variablename2,…;
        …
        CLOSE Cursorname;
        …
END;
```

* SQL Statement must exclude the INTO variable name part

Explicit Cursor Example

```
DECLARE
    v_last_name employees.last_name%TYPE;
    CURSOR c_rich_emp IS SELECT last_name
       FROM employees WHERE salary > 15000;
BEGIN
    OPEN c_rich_emp;
    LOOP
        FETCH c_rich_emp INTO v_last_name;
        EXIT WHEN c_rich_emp%NOTFOUND;
        dbms_output.Put_line(v_last_name
          || ' earns a lot');
    END LOOP;
    CLOSE c_rich_emp;
END;
```

Rowtype/Cursortype – Syntax

```
DECLARE
    …
    CURSOR Cursorname IS Select Statement;
    Variablename Cursorname%ROWTYPE;
     …
BEGIN
     …
     FETCH Cursorname INTO Variablename
     SELECT Columnname1, Columnname2,  …
         ColumnnameN
     FROM…; *
     …
     DBMS_OUTPUT.PUT_LINE (Variablename.Columnname)
     …
END;
```

* Instead of specifying the Select Columns explicit it is as well possible to retrieve all Columns with '*'

Rowtype/Cursortype Example

```
DECLARE
   CURSOR c_rich_emp IS SELECT * FROM
         employees WHERE  salary > 15000;

 v_rich_emp_rec c_rich_emp%ROWTYPE;

BEGIN
   ...
   FETCH c_rich_emp INTO

              v_rich_emp_rec

   dbms_output.Put_line( v_rich_emp_rec.last_name
        || ' earns a lot');
    ...
END;
```

Cursor with FOR Loop – Syntax

DECLARE

        …

        CURSOR *Cursorname* **IS** *Select Statement;*

        …

BEGIN

        …

        FOR *Variablename*[*] **IN** *Cursorname*

        LOOP

                …

        END LOOP;

        …

END;

Most of the time a FOR Loop CURSOR is used as the number of iterations is known upfront!

[*] Type of Variable is ROWTYPE

Cursor with FOR Loop Example

```
DECLARE
   CURSOR c_rich_emp IS
      SELECT * FROM   employees
      WHERE  salary > 15000;
BEGIN
   FOR v_rich_emp_rec IN c_rich_emp
   LOOP
      dbms_output.Put_line(
         v_rich_emp_rec.last_name
         || ' earns a lot');
   END LOOP;
END;
```

FOR Loop with Query in Statement Example

```
--DECLARE Section not necessary
BEGIN
   FOR v_rich_emp_rec IN
      (SELECT *
       FROM    employees
       WHERE   salary > 15000)
   LOOP
      dbms_output.Put_line(
         v_rich_emp_rec.last_name
         || ' earns a lot');
   END LOOP;
END;
```

Most of the time a FOR Loop Cursor is used as the number of iteration is known upfront!

Tasks

- Geben Sie alle Mitarbeiter aus, welche im ersten Firmenjahr eingestellt wurden (1987). Ausgabe Vorname, Nachname und Eintrittsdatum
- Geben Sie den Mitarbeitern eine Gehaltserhöhung (erste Bedingung ist zu wählen) :
  - wenn die Abteilung_ID 40 ist -> 9 %
  - wenn die Abteilung_ID 70 ist -> 17%
  - wenn der Provisionsprozentsatz (commission_pct) > 0,35% ist -> 4%
  - in jedem anderen Fall ->11%
- Erstellen Sie eine Funktion, welcher die ManagerID (Spalte "ManagerID" in Tabelle "Employees" übergeben wird. Diese Funktion soll als Returnwert eine Liste aller Nachnamen seiner Mitarbeiter liefern