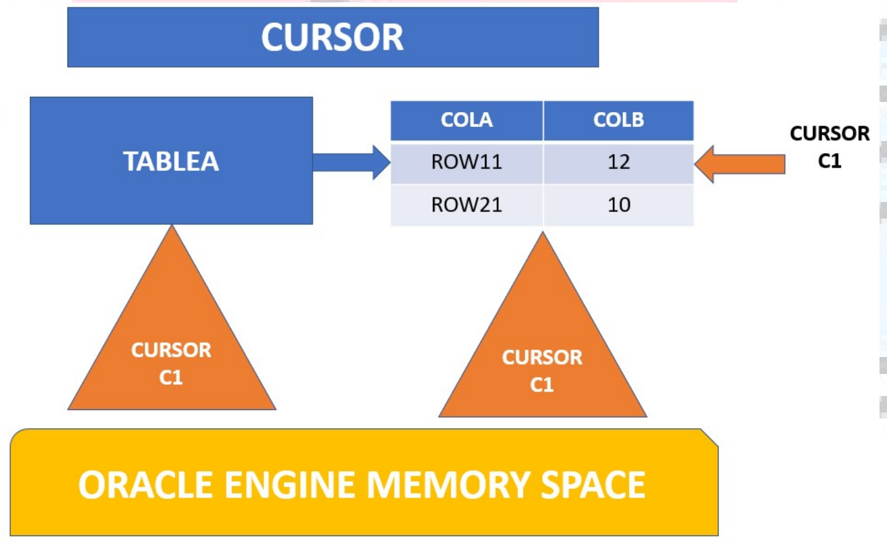
# CHAPTER – 5

# CURSORS

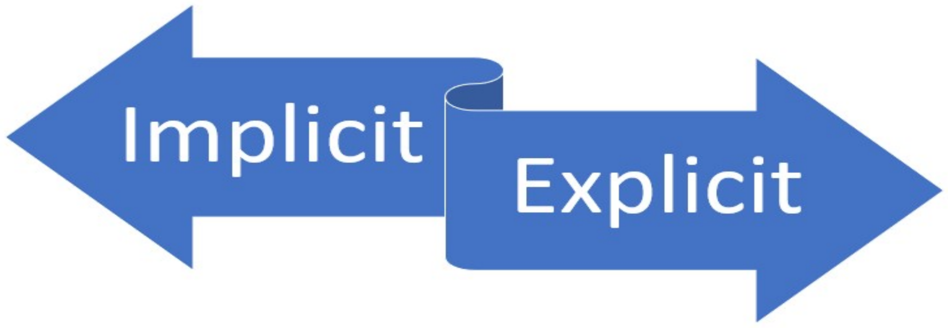
*Let's make coding fun!*

## CURSORS

**Oracle** engine uses work area for its internal processing in order to execute an SQL statements. This work area is private to SQL operations called **CURSOR.** Data stored in Cursor called **Active Data set.** 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.



## TYPES OF CURSORS



- Automatically created by Oracle Engine and managed by Oracle engine internally
- Attributes of this cursor can be used to access information about the status of last SQL statement execution

- Defined by developer to get more control over context area.
- Attributes helps to access active data set and program accordingly

### 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. The SQL cursor has additional attributes and **%BULK_EXCEPTIONS**, designed for use with the **FORALL** statement.

The following table provides the description of the most used attributes –

| | |
|---|---|
| **%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. |
| **%ISOPEN** | Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement |
| **%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. |
| **%ROWCOUNT** | Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement. |

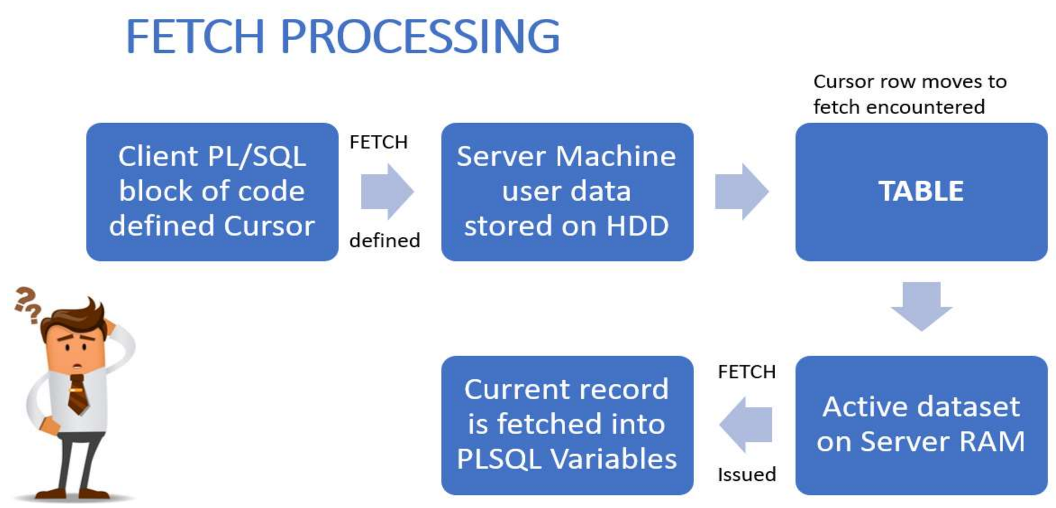
## Explicit Cursors

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


FETCH PROCESSING
Client PL/SQL block of code defined Cursor
FETCH
defined
Server Machine user data stored on HDD
Cursor row moves to fetch encountered
TABLE
Active dataset on Server RAM
FETCH
Issued
Current record is fetched into PLSQL Variables

## 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. For example, we will open the above defined cursor as follows –

OPEN c_customers;

## Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows

FETCH c_customers INTO c_id, c_name, c_addr;

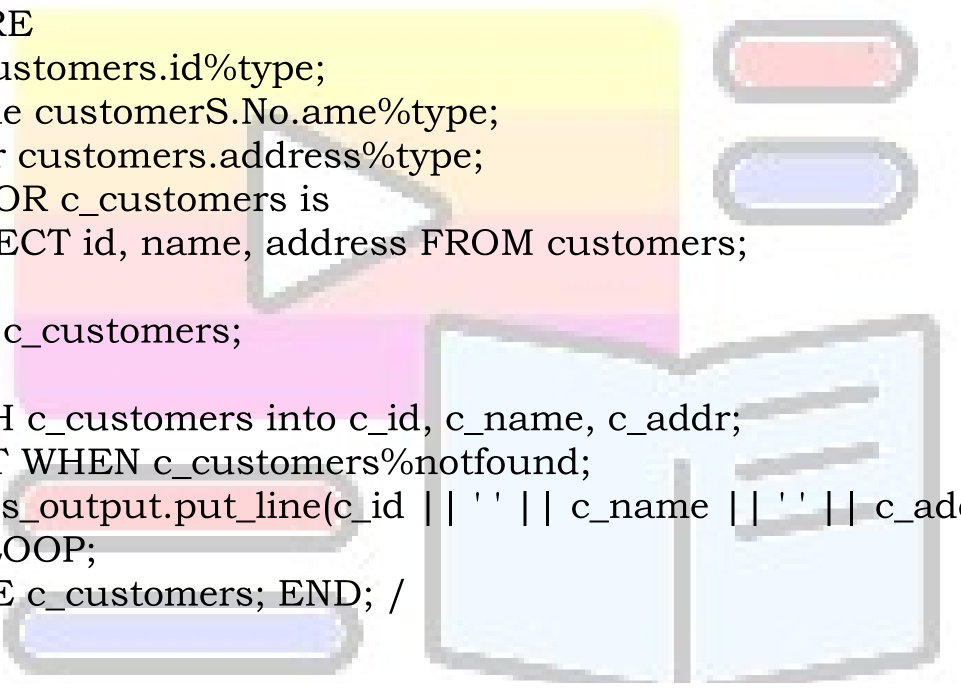
## Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows –

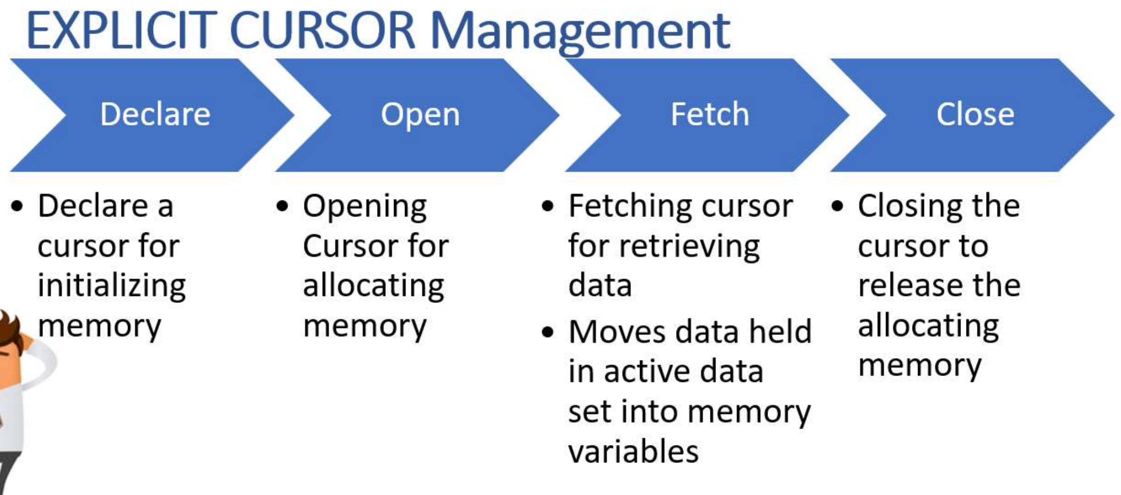
CLOSE c_customers;

## Example

Following is a complete example to illustrate the concepts of explicit cursors:



```
DECLARE
   c_id customers.id%type;
   c_name customerS.No.ame%type;
   c_addr customers.address%type;
   CURSOR c_customers is
      SELECT id, name, address FROM customers;
BEGIN
   OPEN c_customers;
   LOOP
   FETCH c_customers into c_id, c_name, c_addr;
      EXIT WHEN c_customers%notfound;
      dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
   END LOOP;
   CLOSE c_customers; END; /
```

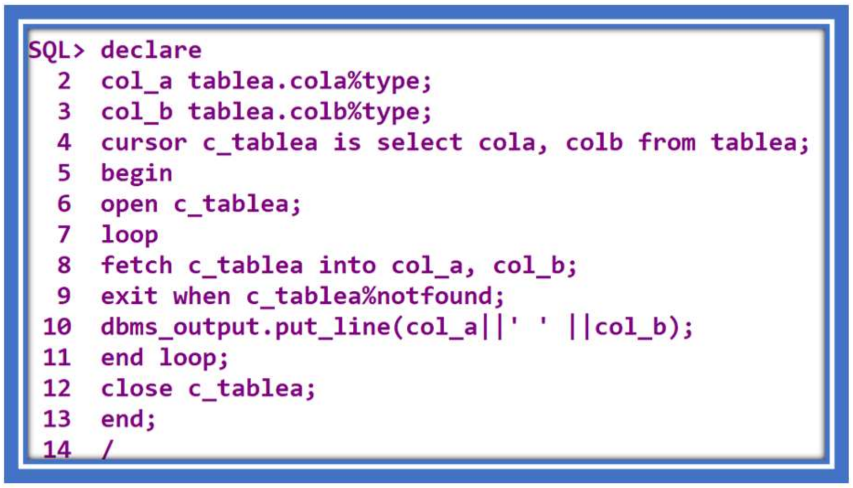

EXPLICIT CURSOR Management
Declare
Open
Fetch
Close
• Declare a cursor for initializing memory
• Opening Cursor for allocating memory
• Fetching cursor for retrieving data
• Moves data held in active data set into memory variables
• Closing the cursor to release the allocating memory

## CURSOR SYNTAX

```
DECLARE
CURSOR <cursor_name>
BEGIN
OPEN cursor_name;
FETCH cursor_name into
variables;
CLOSE cursor_name;
End;
```

| ATTRIBUTES | DETAILS |
|---|---|
| %FOUND | Last fetch success |
| %NOTFOUND | Last fetch failed |
| %ISOPEN | Explicit cursor is open |
| %ROWCOUNT | Returns number of rows fetched from active set |

## Cursors Demonstration



```
SQL> declare
col_a tablea.cola%type;
col_b tablea.colb%type;
cursor c_tablea is select cola, colb from tablea;
begin
open c_tablea;
loop
fetch c_tablea into col_a, col_b;
exit when c_tablea%notfound;
dbms_output.put_line(col_a||' ' ||col_b);
end loop;
close c_tablea;
end;
/
```

Table A

| COLA | COLB |
|---|---|
| ROW11 | 20 |
| ROW21 | 12 |

## CONCLUSION

In this chapter, we explained about Cursors, types of cursors, example of types of cursor and different uses of cursors in application.