CURSOR: (30-11-2024)
========

- it is a temporary memory / work space / sql private area.
- there two types of cursors in sqlserver.
    - i) Implicit cursors
    - ii) Explicit cursors

i) Implicit cursors:
===============

- these cursors are created by the system by default when user perform DML operations on a table in the database.
- implicit cursor memory is used to store the information about DML command that means to check the status of DML command is executed or not.

EX:

```
INSERT INTO DEPT VALUES(50,'DBA','HYD');
(1 row affected)

UPDATE DEPT SET LOC='PUNE' WHERE DEPTNO=50;
(1 row affected)

DELETE FROM DEPT WHERE DEPTNO=50;
(1 row affected)
```

ii) Explicit cursors:
=================

- these cursors are created by user for fetching multiple rows from a database table.
- explicit cursor is holding multiple rows but it can access only one row at a time.so that users are fetching these multiple rows from a table in "one-by-one / row-by-row" manner.
- these explicit cursors are created with the following five steps are,

step1: Declare a cursor variable:
==========================

syntax:
======

```
declare <cursorname> cursor for <select query>;
```

step2: Open cursor connection:
=========================

syntax:
======

```
open <cursor name>;
```

step3: Fetching rows from a table row-by-row manner:
==

syntax:
======

fetch <next / first / last / prior / absolute n / relative n> from <cursor name> [into <variables>];

step4: Close cursor connection:
==========================

syntax:
=======

close <cursor name>;

step5: Deallocating cursor memory:
==============================

syntax:
=======

deallocate <cursor name>;

Fetching methods of Explicit cursor:
==============================

i) Next :
======

- it is a default method of cursor.which is used to fetch rows from a table in next-by-next manner i.e forward direction only.

ii) First :
=======

- Fetching the first row from a table.

iii) Last :
=======

- Fetching the last row from a table

iv) Prior :
========

- Fetching rows from a table in back ward direction.

v) Absolute n:
===========

- Fetching an exact position of row from a table.Here "n" is number.

vi) Relative n:
===========

- Fetching rows from a table either incremental or decremental manner.Here "n" is number.

Ex:
write a cursor program to fetch a single row from emp table?
WITH VARIABLES:
================

```
DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH NEXT FROM C1 INTO @ENAME,@SAL
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
=========

```
THE EMPLOYEE SMITH SALARY IS:- 800
```

WITHOUT VARIABLES:
====================

```
DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP
OPEN C1
FETCH NEXT FROM C1
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
========

```
ENAME     SAL
======    ====
SMITH 800.00
```

02-12-2024:
==========
Ex:
write a cursor program to fetch multiple rows from emp table?(i.e forward direction)
i) with variables:
==============

```
DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH NEXT FROM C1 INTO @ENAME,@SAL------------LOOP STARTS FROM 1ST ROW
```

```
WHILE @@FETCH_STATUS=0
BEGIN
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
FETCH NEXT FROM C1 INTO @ENAME,@SAL------------LOOP CONTINUE UPTO LAST
END
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
=========

```
THE EMPLOYEE SMITH SALARY IS:- 800
THE EMPLOYEE ALLEN SALARY IS:- 1600
.................................................................
.................................................................
```

ii) without variables:
=================

```
DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP
OPEN C1
FETCH NEXT FROM C1 ------------LOOP STARTS FROM 1ST ROW
WHILE @@FETCH_STATUS=0
BEGIN
FETCH NEXT FROM C1 ------------LOOP CONTINUE UPTO LAST
END
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
========

```
ENAME        SAL
======       ====
SMITH 800.00
ALLEN 1600.00
..........  .............
..........  .............
```

What is @@fetch_status:
=====================

- it is a pre-defined variable which is used to check the status of cursor is having data or not.
  > if cursor is having data then it return 0
  > if cursor is not having data then it return -1.

EX:
write a cursor program to fetch multiple rows from a table in backward direction?

```
DECLARE C1 CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH LAST FROM C1 INTO @ENAME,@SAL------------LOOP STARTS FROM LAST ROW
WHILE @@FETCH_STATUS=0
BEGIN
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
FETCH PRIOR FROM C1 INTO @ENAME,@SAL------------LOOP CONTINUE UPTO FIRST
END
CLOSE C1
DEALLOCATE C1
```

ERROR:

```
fetch: The fetch typ last cannot be used with forward only cursors.
```

NOTE:
======
- By default cursor are fetching rows from a table in forward direction only and it supports only one
fetching method i.e "NEXT".
- If we want fetch the rows from a table in any directional then we must use "SCROLL" cursors and
it supports all fetching methods are "NEXT,PRIOR,FIRST,LAST,ABSOLUTE n and RELATIVE n".

By using "SCROLL" cursor:
======================

```
DECLARE C1 SCROLL CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH LAST FROM C1 INTO @ENAME,@SAL------------LOOP STARTS FROM LAST ROW
WHILE @@FETCH_STATUS=0
BEGIN
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
FETCH PRIOR FROM C1 INTO @ENAME,@SAL------------LOOP CONTINUE UPTO FIRST
END
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
========

```
THE EMPLOYEE MILLER SALARY IS:-1300.00
THE EMPLOYEE FORD SALARY IS:-3000.00
......................................................................
......................................................................
```

EX:
write a cursor program to fetch the even position rows from a table?

```
DECLARE C1 SCROLL CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH ABSOLUTE 2 FROM C1 INTO @ENAME,@SAL
WHILE @@FETCH_STATUS=0
BEGIN
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
FETCH RELATIVE 2 FROM C1 INTO @ENAME,@SAL
END
CLOSE C1
DEALLOCATE C1
```

EX:
write a cursor program to fetch the first row and the last row from a table?

```
DECLARE C1 SCROLL CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH FIRST FROM C1 INTO @ENAME,@SAL
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
FETCH LAST FROM C1 INTO @ENAME,@SAL
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
CLOSE C1
DEALLOCATE C1
```

EX:
write a cursor program to fetch the 9th position row from a table?

```
DECLARE C1 SCROLL CURSOR FOR SELECT ENAME,SAL FROM EMP
DECLARE @ENAME VARCHAR(10),@SAL MONEY
OPEN C1
FETCH ABSOLUTE 9 FROM C1 INTO @ENAME,@SAL
PRINT 'THE EMPLOYEE'+' '+@ENAME+' '+'SALARY IS:-'+CAST(@SAL AS VARCHAR)
CLOSE C1
DEALLOCATE C1
```

03-12-2024:
===========

Using all fetching method in a cursor program:
===

```
DECLARE C1 SCROLL CURSOR FOR SELECT EMPNO FROM EMP
DECLARE @EMPNO INT
OPEN C1
FETCH LAST FROM C1 INTO @EMPNO
PRINT @EMPNO
FETCH PRIOR FROM C1 INTO @EMPNO
PRINT @EMPNO
FETCH ABSOLUTE 9 FROM C1 INTO @EMPNO
PRINT @EMPNO
FETCH RELATIVE -2 FROM C1 INTO @EMPNO
PRINT @EMPNO
FETCH FIRST FROM C1 INTO @EMPNO
PRINT @EMPNO
FETCH NEXT FROM C1 INTO @EMPNO
PRINT @EMPNO
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
========

```
7934
7902
7839
7782
7369
7499
```

STATIC & DYNAMIC CURSORS:
==========================

- IF A CURSOR IS DECLARE AS STATIC AFTER OPENING THE CURSOR ANY MODIFICATIONS
THAT ARE PERFORMED TO THE DATA IN THE TABLE WILL NOT BE REFLECTED INTO CURSOR SO THE
CURSOR CONTAINS OLD VALUES ONLY IN IT.

EX ON STATIC CURSOR:
=====================

```
DECLARE C1 CURSOR STATIC FOR SELECT SAL FROM EMP WHERE EMPNO=7839
DECLARE @SAL MONEY
OPEN C1
UPDATE EMP SET SAL=8000 WHERE EMPNO=7839
```

```
FETCH NEXT FROM C1 INTO @SAL
PRINT @SAL
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
========

```
salary in emp table is     : 8000 (new salary)
salary in cursor table is : 5000 (old salary)
```

EX ON DYNAMIC CURSOR:
=====================

```
DECLARE C1 CURSOR DYNAMIC FOR SELECT SAL FROM EMP WHERE EMPNO=7839
DECLARE @SAL MONEY
OPEN C1
UPDATE EMP SET SAL=9000 WHERE EMPNO=7839
FETCH NEXT FROM C1 INTO @SAL
PRINT @SAL
CLOSE C1
DEALLOCATE C1
```

OUTPUT:
========

```
salary in emp table is     : 9000 (new salary)
salary in cursor table is  :  9000 (new salary)
```

NOTE:
======

- CURSORS ARE DEGRADING THE PERFORMANCE OF AN APPLICATION BECAUSE THESE
ARE FETCHING ROWS FROM A TABLE IN ONE – BY – ONE MANNER / ROW – BY –ROW MANNER.

==