(T21)討論Cursor  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T21)討論Cursor  
=======================================================================  
1. Create Sample Data

2. Cursor basic

3. Cursor Scroll,FIRST, NEXT

4. Cursor Scroll,LAST, PRIOR

5. Cursor Scroll,ABSOLUTE 9, RELATIVE 10

6. Cursor Scroll,ABSOLUTE 9, RELATIVE 10

-----------

7. Cursor options and scope

8. Cursor basic, store procedure

-----------

9. Update TableA.ColumnA1 with TableACursor

9.1. Update with Cursor

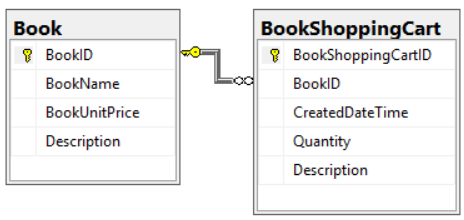
9.2. Replace Cursor by Normal Update.

-----------

10. Update TableB.ColumnB4 with TablAACursor

11. Clean up  
=======================================================================

1. Create Sample Data



--=========================================================

--T021\_01\_Create Sample Data

--=========================================================

--=========================================================

--T021\_01\_01

----------------------------------------------------------

--T021\_01\_01\_01

--Create Table

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookShoppingCart' ) )

    BEGIN

        DROP TABLE BookShoppingCart;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Book' ) )

    BEGIN

        DROP TABLE Book;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE [dbo].[Book]

(

  [BookID] [INT] PRIMARY KEY

                 IDENTITY(1, 1)

                 NOT NULL ,

  [BookName] [NVARCHAR](100) NULL ,

  [BookUnitPrice] [MONEY] NULL ,

  [Description] [NVARCHAR](1000) NULL

);

GO -- Run the previous command and begins new batch

CREATE TABLE [dbo].[BookShoppingCart]

(

  [BookShoppingCartID] [INT] PRIMARY KEY

                             IDENTITY(1, 1)

                             NOT NULL ,

  [BookID] [INT] FOREIGN KEY REFERENCES [dbo].[Book] ( [BookID] )

                 NOT NULL ,

  [CreatedDateTime] [DATETIME] NULL ,

  [Quantity] [INT] NULL ,

  [Description] [NVARCHAR](1000) NULL

);

GO -- Run the previous command and begins new batch

----------------------------------------------------------

--T021\_01\_01\_02

--Insert sample data to [Book] table

--Whole T021\_01\_01 has to execute together.

--Book Counter

--\*\*Changeable: Amount of data Rows

DECLARE @TotalBookRows INT = 70;

DECLARE @BookCount INT= 1;

-- random UnitPrice between 1 and 100

DECLARE @RandomUnitPrice MONEY;

DECLARE @BookUnitPrice\_Max INT;

DECLARE @BookUnitPrice\_Min INT;

SET @BookUnitPrice\_Min = 1;

SET @BookUnitPrice\_Max = 100;

WHILE ( @BookCount <= @TotalBookRows )

    BEGIN

        SELECT  @RandomUnitPrice = FLOOR(RAND() \* ( @BookUnitPrice\_Max

                                                    - @BookUnitPrice\_Min )

                                         + @BookUnitPrice\_Min);

        INSERT  INTO [dbo].[Book]

        VALUES  ( 'Book ' + CAST(@BookCount AS NVARCHAR(20)), @RandomUnitPrice,

                  'Book Description ' + CAST(@BookCount AS NVARCHAR(20)) );

        PRINT @BookCount;

        SET @BookCount += 1;

    END;

/\*

1.

Random Number

1.1.

RAND([seed])

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/rand-transact-sql>

<https://www.w3schools.com/sql/func_mysql_rand.asp>

Returns a pseudo-random float value from 0 through 1, exclusive.

0 <= ReturnNumber < 1

Same seed always returns the same RAND([seed]) value.

1.2.

FLOOR(RAND()\*(b-a)+a);

Where a is the smallest number and b is the largest number that you want to generate a random number for.

Reference:

<https://www.techonthenet.com/sql_server/functions/rand.php>

PRINT FLOOR(RAND()\*(25-10)+10);

10 <= IntNumber < 25

2.

Random DateTime

--Ch25\_08

--Get Random DateTime

--Reference: <http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/>

DECLARE @RandomDateTime DATETIME;

DECLARE @DateFrom DATETime = '2012-01-01'

DECLARE @DateTo DATeTime = '2017-06-30'

DECLARE @DaysRandom Int= 0

DECLARE @MillisRandom Int=0

--get random number of days

select @DaysRandom= DATEDIFF(day,@DateFrom,@DateTo)

SELECT @DaysRandom = ROUND(((@DaysRandom -1) \* RAND()), 0)

--get random millis

SELECT @MillisRandom = ROUND(((99999999) \* RAND()), 0)

SELECT @RandomDateTime = DATEADD(day, @DaysRandom, @DateFrom)

SELECT @RandomDateTime = DATEADD(MILLISECOND, @MillisRandom, @RandomDateTime)

SELECT @RandomDateTime

\*/

----------------------------------------------------------

--T021\_01\_01\_03

--Insert sample data to [BookShoppingCart] table

--Whole T021\_01\_01 has to execute together.

--BookShoppingCart Counter

DECLARE @TotalBookShoppingCartRows INT;

DECLARE @BookShoppingCartCount INT;

SET @BookShoppingCartCount = 1;

SET @TotalBookShoppingCartRows = 100;

-- @RandomBookID

DECLARE @RandomBookID INT;

DECLARE @RandomBookID\_Max INT;

DECLARE @RandomBookID\_Min INT;

SET @RandomBookID\_Min = 1;

SET @RandomBookID\_Max = @TotalBookRows - ( @TotalBookRows \* 0.1 );

--Should be @RandomBookID\_Max = @TotalBookRows,

--but I purposely set  @RandomBookID\_Max =  @TotalBookRows - ( @TotalBookRows \* 0.1 )

--I want some book data that was never sold.

--@RandomCreatedDateTime

--Reference: <http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/>

DECLARE @RandomCreatedDateTime DATETIME;

DECLARE @DateFrom DATETIME = '2012-01-01';

DECLARE @DateTo DATETIME = '2017-06-30';

DECLARE @DaysRandom INT= 0;

DECLARE @MillisRandom INT= 0;

-- @RandomQuantity is between 1 to 10

DECLARE @RandomQuantity INT;

DECLARE @RandomQuantity\_Max INT;

DECLARE @RandomQuantity\_Min INT;

SET @RandomQuantity\_Min = 1;

SET @RandomQuantity\_Max = 10;

WHILE ( @BookShoppingCartCount <= @TotalBookShoppingCartRows )

    BEGIN

             --1. @RandomBookID

        SELECT  @RandomBookID = FLOOR(RAND() \* ( @RandomBookID\_Max

                                                 - @RandomBookID\_Min )

                                      + @RandomBookID\_Min);

             --2. @RandomQuantity

        SELECT  @RandomQuantity = FLOOR(RAND() \* ( @RandomQuantity\_Max

                                                   - @RandomQuantity\_Min )

                                        + @RandomQuantity\_Min);

             --3. @RandomCreatedDateTime

             --get random number of days

        SELECT  @DaysRandom = DATEDIFF(DAY, @DateFrom, @DateTo);

        SELECT  @DaysRandom = ROUND(( ( @DaysRandom - 1 ) \* RAND() ), 0);

             --get random millis

        SELECT  @MillisRandom = ROUND(( ( 99999999 ) \* RAND() ), 0);

        SELECT  @RandomCreatedDateTime = DATEADD(DAY, @DaysRandom, @DateFrom);

        SELECT  @RandomCreatedDateTime = DATEADD(MILLISECOND, @MillisRandom,

                                                 @RandomCreatedDateTime);

        INSERT  INTO [dbo].[BookShoppingCart]

        VALUES  ( @RandomBookID, @RandomCreatedDateTime, @RandomQuantity, ('Description ' + CONVERT(NVARCHAR,@BookShoppingCartCount)) );

        PRINT @BookShoppingCartCount;

        SET @BookShoppingCartCount += 1;

    END;

GO -- Run the previous command and begins new batch

--=========================================================

--T021\_01\_02

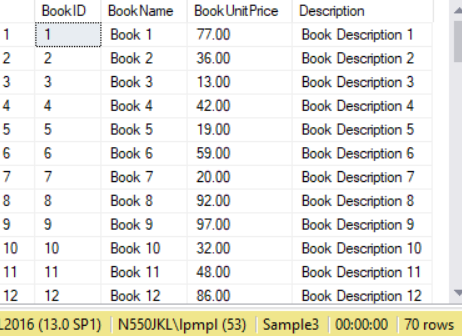
SELECT  \*

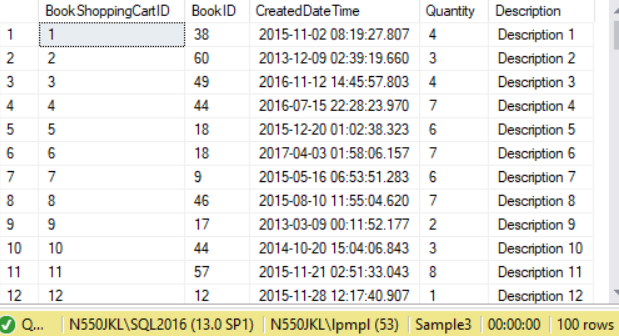
FROM    [dbo].[Book];

SELECT  \*

FROM    [dbo].[BookShoppingCart];

GO -- Run the previous command and begins new batch





==================================================

2. Cursor basic

--=========================================================

--T021\_02\_Create Cursor basic

--=========================================================

DECLARE @BookID INT;

DECLARE @BookName NVARCHAR(MAX);

DECLARE BookCursor CURSOR

FOR

    SELECT  b.BookID ,

            b.BookName

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

WHILE @@FETCH\_STATUS = 0

    BEGIN

        SELECT  @BookName + ' Sold DateTime' ,

                bsc.CreatedDateTime AS BookSoldDateTime

        FROM    dbo.BookShoppingCart bsc

        WHERE   bsc.BookID = @BookID;

             --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

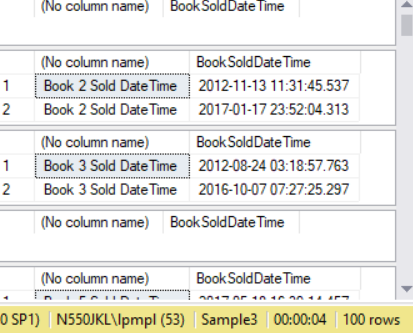
        FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

    END;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



/\*

1.

Cursor Syntax 1:

--DECLARE @ColumnA1 dataType;

--DECLARE @ColumnA2 dataType;

--DECLARE CursorName CURSOR (options...)

--FOR

--    SELECT  a.ColumnA1 ,

--            a.ColumnA2

--    FROM    TableA a;

--OPEN CursorName;

---- FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2

---- must map to SELECT  a.ColumnA1 ,a.ColumnA2

--FETCH NEXT FROM CursorName INTO @ColumnA1, @ColumnA2;

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           SELECT  @ColumnA2,

--                         b.ColumnB3 AS AliasName

--        FROM    TableB b

--        WHERE   b.ColumnA1 = @ColumnA1;

--           --FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

--        FETCH NEXT FROM CursorName INTO @ColumnA1, @ColumnA2;

--    END;

--CLOSE CursorName;

--DEALLOCATE CursorName;

1.1.

CURSOR is a way to step through a set of records one row at a time.

It is like a pointer in each record and moving through one step at a time.

--OPEN CursorName;

---FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2

Open the Cursor and point to the (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) data set

must map to SELECT  a.ColumnA1 ,a.ColumnA2.

If you have selected extra Columns,

then you have to declare extra variable to map them.

In this case,

declare @ColumnA1 and @ColumnA2 and map to  --SELECT  a.ColumnA1 ,a.ColumnA2

Read the value for each record set and put into variables @ColumnA1 and @ColumnA2

--WHILE @@FETCH\_STATUS = 0

means successfully read a next record into variables @ColumnA1 and @ColumnA2.

--SELECT  @ColumnA2,

--           b.ColumnB3 AS AliasName

--FROM    TableB b

--WHERE   b.ColumnA1 = @ColumnA1;

during while loop, you can use  @ColumnA1 and @ColumnA2 in other sql statement.

----FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

Use the Cursor and point to the (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) data set,

--WHILE @@FETCH\_STATUS = 0

check if successfully read a (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) record into variables @ColumnA1 and @ColumnA2.

This while loop will run until NOT(@@FETCH\_STATUS = 0)

That means the pointer reach to the end of loop and get out the loop.

1.2.

--DECLARE CursorName CURSOR (options...)

--FOR

--    SELECT  ...

--OPEN CursorName;

--FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2

--...

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           ...

--    END;

--CLOSE CursorName;

--DEALLOCATE CursorName;

1.2.1.

This is the life of CURSOR

--DECLARE CursorName CURSOR (options...)

--FOR

Then

--OPEN CursorName;

--FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2

Then

--WHILE @@FETCH\_STATUS = 0

Then

--CLOSE CursorName;

Then

--DEALLOCATE CursorName;

means get rid of the CursorName.

-------------------------------------------------

2.

--DECLARE @BookID INT;

--DECLARE @BookName NVARCHAR(MAX);

--DECLARE BookCursor CURSOR

--FOR

--    SELECT  b.BookID ,

--            b.BookName

--    FROM    dbo.Book b;

--OPEN BookCursor;

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           SELECT  @BookName + ' Sold DateTime',

--                         bsc.CreatedDateTime AS BookSoldDateTime

--        FROM    dbo.BookShoppingCart bsc

--        WHERE   bsc.BookID = @BookID;

--        FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

--    END;

--CLOSE BookCursor;

--DEALLOCATE BookCursor;

2.1.

--DECLARE BookCursor CURSOR

--FOR

--    SELECT  b.BookID ,

--            b.BookName

--    FROM    dbo.Book b;

--OPEN BookCursor;

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

2.1.1.

CURSOR is a way to step through a set of records one row at a time.

It is like a pointer in each record and moving through one step at a time.

2.1.2.

--DECLARE BookCursor CURSOR

--FOR

--    SELECT  b.BookID ,

--            b.BookName

--    FROM    dbo.Book b;

declare @BookID and @BookName variables to read the values for each record set.

2.1.3.

--OPEN BookCursor;

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

Open the Cursor and point to the first data set,

Read the value and put into variables @BookID and @BookName

2.2.

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           SELECT  @BookName + ' Sold DateTime',

--                         bsc.CreatedDateTime AS BookSoldDateTime

--        FROM    dbo.BookShoppingCart bsc

--        WHERE   bsc.BookID = @BookID;

--        FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

--    END;

2.2.1.

--WHILE @@FETCH\_STATUS = 0

means successfully read a next record into variables @BookID and @BookName.

--SELECT  @BookName + ' Sold DateTime',

--           bsc.CreatedDateTime AS BookSoldDateTime

--FROM    dbo.BookShoppingCart bsc

--WHERE   bsc.BookID = @BookID;

during while loop, you can use  @ColumnA1 and @ColumnA2 in other sql statement.

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

Use the Cursor and point to the next data set,

Read the value and put into variables @BookID and @BookName

--WHILE @@FETCH\_STATUS = 0

check if successfully read a next record into variables @BookID and @BookName.

This while loop will run until NOT(@@FETCH\_STATUS = 0)

That means the pointer reach to the end of loop and get out the loop.

2.3.

-- DECLARE BookCursor CURSOR

--FOR

--    SELECT  ...

--OPEN BookCursor;

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

--...

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           ...

--    END;

--CLOSE BookCursor;

--DEALLOCATE BookCursor;

2.3.1.

This is the life of CURSOR

-- DECLARE BookCursor CURSOR

--FOR

Then

--OPEN BookCursor;

Then

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

Then

--WHILE @@FETCH\_STATUS = 0

Then

--CLOSE BookCursor;

Then

--DEALLOCATE BookCursor;

means get rid of the CURSOR.

\*/

==================================================

3. Cursor Scroll,FIRST, NEXT

--=========================================================

--T021\_03\_Cursor Scroll,FIRST, NEXT

--=========================================================

DECLARE BookCursor CURSOR SCROLL

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice ,

            b.[Description]

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH FIRST FROM BookCursor;

--FETCH NEXT FROM BookCursor;

WHILE @@FETCH\_STATUS = 0

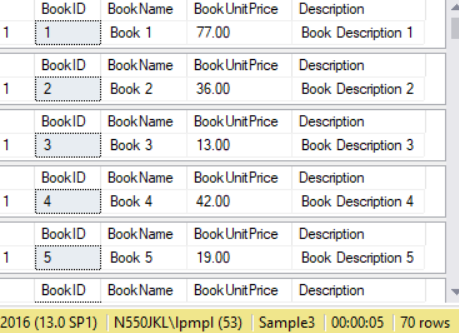
    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

    FETCH NEXT FROM BookCursor;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



/\*

1.

--DECLARE BookCursor CURSOR SCROLL

When you declare CURSOR with SCROLL option

Then it will allow you to

---- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

--FETCH FIRST FROM BookCursor;

----FETCH NEXT FROM BookCursor;

--WHILE @@FETCH\_STATUS = 0

--     --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

--     FETCH NEXT FROM BookCursor;

It will start to SELECT the first data row

and then SELECT forward One by One until LAST records.

--FETCH FIRST FROM BookCursor;

----FETCH NEXT FROM BookCursor;

means get the first one.

--     FETCH NEXT FROM BookCursor;

means read the next one until end of while loop.

\*/

==================================================

4. Cursor Scroll,LAST, PRIOR

--=========================================================

--T021\_04\_Cursor Scroll,LAST, PRIOR

--=========================================================

DECLARE BookCursor CURSOR SCROLL

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice ,

            b.[Description]

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH LAST FROM BookCursor;

WHILE @@FETCH\_STATUS = 0

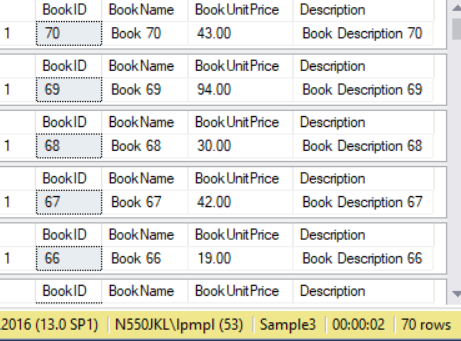
    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

    FETCH PRIOR FROM BookCursor;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



/\*

1.

--DECLARE BookCursor CURSOR SCROLL

When you declare CURSOR with SCROLL option

Then it will allow you to

---- FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

--FETCH LAST FROM BookCursor;

--WHILE @@FETCH\_STATUS = 0

--     --FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

--     FETCH PRIOR FROM BookCursor;

It will start to SELECT the last data row

and then SELECT backforward One by One until first records.

--FETCH LAST FROM BookCursor;

means get the last one.

--     FETCH PRIOR FROM BookCursor;

means read the previous one until end of while loop.

\*/

==================================================

5. Cursor Scroll,ABSOLUTE 9, RELATIVE 10

--=========================================================

--T021\_05\_Cursor Scroll,ABSOLUTE 9, RELATIVE 10

--=========================================================

DECLARE BookCursor CURSOR SCROLL

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice ,

            b.[Description]

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH ABSOLUTE 9 FROM BookCursor;

WHILE @@FETCH\_STATUS = 0

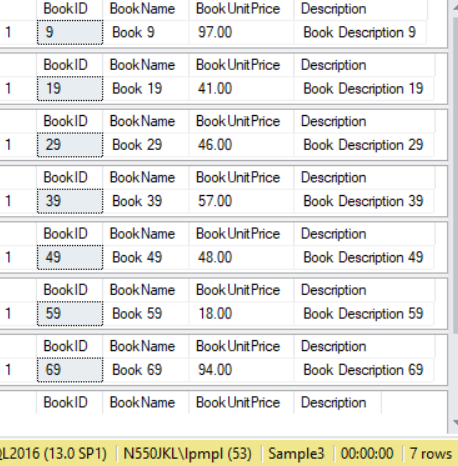
    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

    FETCH RELATIVE 10 FROM BookCursor;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



/\*

1.

--DECLARE BookCursor CURSOR SCROLL

When you declare CURSOR with SCROLL option

Then it will allow you to

---- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

--FETCH ABSOLUTE 9 FROM BookCursor;

--WHILE @@FETCH\_STATUS = 0

--    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

--     FETCH RELATIVE 10 FROM BookCursor;

In the beginning, your CURSOR point to id 1

--FETCH ABSOLUTE 9 FROM BookCursor

will make your CURSOR point to id 9

--     FETCH RELATIVE 10 FROM BookCursor

will make your CURSOR point to next item with id 9+10=19

and while loop to keep going to move next item with 9+10+10=29

unitl end of while loop.

\*/

==================================================

6. Cursor Scroll,ABSOLUTE 9, RELATIVE 10

DECLARE BookCursor CURSOR SCROLL

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice ,

            b.[Description]

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH ABSOLUTE -1 FROM BookCursor;

WHILE @@FETCH\_STATUS = 0

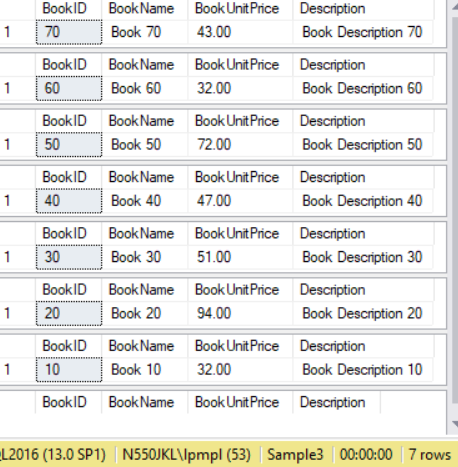
    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

    FETCH RELATIVE -10 FROM BookCursor;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



/\*

1.

--DECLARE BookCursor CURSOR SCROLL

When you declare CURSOR with SCROLL option

Then it will allow you to

---- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

--FETCH ABSOLUTE -1 FROM BookCursor;

--WHILE @@FETCH\_STATUS = 0

In the beginning, your CURSOR point to the position -1

that means the last data set with the id 300

--    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

--     FETCH RELATIVE -10 FROM BookCursor;

will make your CURSOR point to next item with id 300-10=290

until the end of loop.

\*/

==================================================

7. Cursor options and scope

--=========================================================

--T021\_07\_Cursor options and scope

--=========================================================

--DECLARE BookCursor CURSOR (options...)

--DECLARE BookCursor CURSOR (LOCAL/GLOBAL/SCROLL/GLOBAL SCROLL/...

--.../FORWARD\_ONLY/FAST\_FORWARD/STATIC/KEYSET/DYNAMIC/...

--.../Read\_Only/SCROLL\_LOCKS/OPTIMISTIC/...

--.../GLOBAL FORWARD\_ONLY STATIC READ\_ONLY/SCROLL FORWARD\_ONLY/...

--.../SCROLL FAST\_FORWARD {invalid because scroll is not read only}  ...)

DECLARE BookCursor CURSOR SCROLL

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice ,

            b.[Description]

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH ABSOLUTE -1 FROM BookCursor;

WHILE @@FETCH\_STATUS = 0

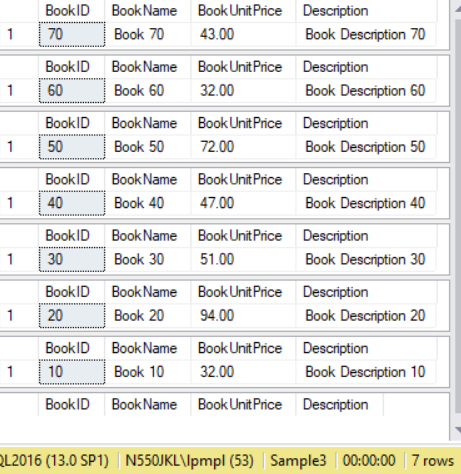
    --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

    FETCH RELATIVE -10 FROM BookCursor;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



/\*

1.

Scope of CURSOR

LOCAL, GLOBAL

1.1.

-- DECLARE BookCursor CURSOR LOCAL

LOCAL means the CURSOR is valid in current batch

That means it is valid before GO

1.2.

-- DECLARE BookCursor CURSOR GLOBAL

GLOBAL means the CURSOR is valid in any batch

That means it is still valid after GO

By default, CURSOR is GLOBAL

However, you may change the default.

Database Name --> Right Click --> Properties -->

options --> Default CURSOR --> GLOBAL / LOCAL

2.

Scroll seting for CURSOR

SCROLL, FORWARD\_ONLY

2.1.

--DECLARE BookCursor CURSOR SCROLL

When you declare SCROLL option for CURSOR

you may

--FETCH First FROM BookCursor;   or

--FETCH LAST FROM BookCursor;    or

--FETCH ABSOLUTE -1 FROM BookCursor

....

-- FETCH PRIOR FROM BookCursor;

-- FETCH RELATIVE -10 FROM BookCursor;

That means you may choose forward of backward.

2.2.

--DECLARE BookCursor CURSOR FORWARD\_ONLY

You may only fetch forward.

--FETCH NEXT FROM BookCursor;

You can not use

--FETCH First FROM BookCursor;

3.

Record Set Types for CURSOR,

STATIC, DYNAMIC, KEYSET, FAST\_FORWARD

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/declare-cursor-transact-sql>

3.1.

-- DECLARE BookCursor CURSOR FAST\_FORWARD

FAST\_FORWARD is read-only and forward only,

but it enable performance optimizations

if you don't need to make any changes

3.2.

-- DECLARE BookCursor CURSOR STATIC

STATIC option means create a copy of your select statement results

into a temp db which you can not make any changes.

If any changes from other user, you will not see.

Because you are seeing a copy of temp db.

3.3.

-- DECLARE BookCursor CURSOR KEYSET

KEYSET option means create a copy of key value

from your select statements results into a temp db.

Because the temp db only store key, so you may see other user changes.

However, you can not see any changes if other user delete or insert new records.

3.4.

-- DECLARE BookCursor CURSOR DYNAMIC

That means you can update, delete, insert.

You may also see other users update, delete, insert.

4.

Record Locking Options for CURSORs

Read\_Only, SCROLL\_LOCKS, OPTIMISTIC

4.1.

-- DECLARE BookCursor CURSOR Read\_Only

Read\_Only means you may not make any changes.

If you use FAST\_FORWARD which will automaticly apply Read\_Only

4.2.

-- DECLARE BookCursor CURSOR SCROLL\_LOCKS

SCROLL\_LOCKS option means

when your cursor moves to a record and that record is locked.

It prevents other users from making changes to the record you locked.

Thus, it guarantee you are always able to successfully update the record.

4.3.

-- DECLARE BookCursor CURSOR OPTIMISTIC

OPTIMISTIC option only locks a record at the instant you try to make change.

If another user had made a change to the record in between your cursor scrolling

to it, then attempting to make the change would fail.

5.

Combining Cursor Options

-- DECLARE BookCursor CURSOR GLOBAL FORWARD\_ONLY STATIC READ\_ONLY

It is fine when you combining cursor option.

-- DECLARE BookCursor CURSOR SCROLL FORWARD\_ONLY

or

-- DECLARE BookCursor CURSOR SCROLL FAST\_FORWARD

it is not valid, because scroll is not read only

\*/

==================================================

8. Cursor basic, store procedure

--=========================================================

--T021\_08\_Cursor basic, store procedure.

--=========================================================

--If store procedure is EXISTS, then drop it.

IF EXISTS(SELECT \*

          FROM   INFORMATION\_SCHEMA.ROUTINES

          WHERE  ROUTINE\_NAME = 'spListBookSoldDateTime'

                 AND SPECIFIC\_SCHEMA = 'dbo')

  BEGIN

      DROP PROCEDURE spListBookSoldDateTime

  END

GO -- Run the previous command and begins new batch

--Create Store Procedure

CREATE PROC spListBookSoldDateTime

    (

      @BookID INT ,

      @BookName NVARCHAR(100)

    )

AS

    BEGIN

        SELECT  @BookName + ' Sold DateTime' ,

                bsc.CreatedDateTime AS BookSoldDateTime

        FROM    dbo.BookShoppingCart bsc

        WHERE   bsc.BookID = @BookID;

    END;

GO -- Run the previous command and begins new batch

--Cursor basic with store procedure.

--See the comment in

DECLARE @BookID INT;

DECLARE @BookName NVARCHAR(MAX);

DECLARE BookCursor CURSOR

FOR

    SELECT  b.BookID ,

            b.BookName

    FROM    dbo.Book b;

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

WHILE @@FETCH\_STATUS = 0

    BEGIN

             --SELECT  @BookName + ' Sold DateTime',

             --           bsc.CreatedDateTime AS BookSoldDateTime

        --FROM    dbo.BookShoppingCart bsc

        --WHERE   bsc.BookID = @BookID;

        EXEC spListBookSoldDateTime @BookID, @BookName;

             --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2;

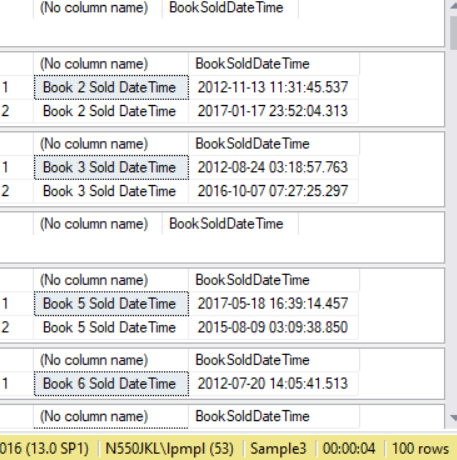
        FETCH NEXT FROM BookCursor INTO @BookID, @BookName;

    END;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch



==================================================

9. Update TableA.ColumnA1 with  TableACursor

--=========================================================

--T021\_09\_Update TableA.ColumnA1 with  TableACursor

--Replace TableACursor by normal Update TableA.

--=========================================================

/\*

Goal:

1.

Update TableA.ColumnA1 with  TableACursor

2.

dbo.Book b has b.BookID, b.BookName, b.BookUnitPrice.

Depending on b.BookUnitPrice, we need to update b.[Description]

3.

Curson is very bad in Performance.

Thus, Replace TableACursor by normal Update TableA.

\*/

9.1. Update with Cursor

--=========================================================

--T021\_09\_01

--Update with Cursor

DECLARE @BookID INT;

DECLARE @BookName NVARCHAR(100);

DECLARE @BookUnitPrtice MONEY;

DECLARE @DescriptionExtraInfo NVARCHAR(100);

DECLARE BookCursor CURSOR

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice

    FROM    dbo.Book b

       -- \*\*\* declare a CURSOR "BookCursor" and help to update the field [Book].[Description]

       FOR UPDATE OF b.[Description];

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

--INTO @BookID, @BookName, @BookUnitPrtice;    must map to

--SELECT  b.BookID , b.BookName, b.BookUnitPrice

FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

WHILE @@FETCH\_STATUS = 0

    BEGIN

             --set @DescriptionExtraInfo

        SELECT  @DescriptionExtraInfo =

                    CASE

                           WHEN ( @BookUnitPrtice > 25 AND @BookUnitPrtice <= 50)

                THEN '2nd level book.'

                WHEN ( @BookUnitPrtice > 50 AND @BookUnitPrtice <= 75)

                THEN '3rd level book.'

                WHEN ( @BookUnitPrtice > 75 )

                THEN '4th level book.'

                ELSE ''

            END;

             -- Update [Description]

        UPDATE  dbo.Book

        SET     dbo.Book.[Description] += (' -- ' + @DescriptionExtraInfo)

             --\*\*\* WHERE   Book.BookID = @BookID;

        WHERE CURRENT OF BookCursor;

             --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

             FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

    END;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch

SELECT  b.BookID,

             b.BookName,

             b.BookUnitPrice,

             b.[Description] AS BookDescription

FROM    dbo.Book b

WHERE   (

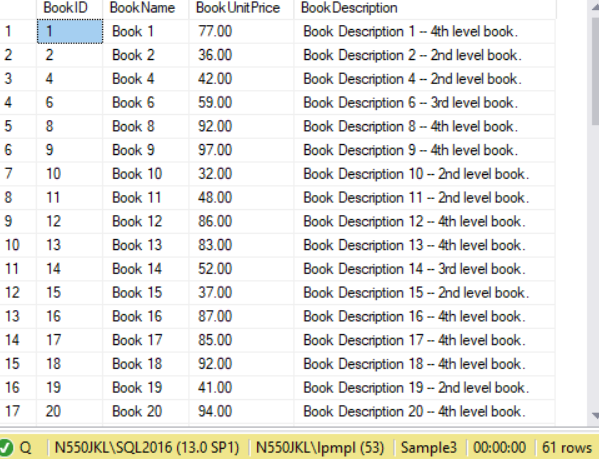
                    ( b.BookUnitPrice > 25 AND b.BookUnitPrice <= 50) OR

                    ( b.BookUnitPrice > 50 AND b.BookUnitPrice <= 75) OR

                    ( b.BookUnitPrice > 75)

        );

GO -- Run the previous command and begins new batch



9.2. Replace Cursor by Normal Update.

--=========================================================

--T021\_09\_02

--Replace Cursor by Normal Update.

UPDATE  dbo.Book

SET     dbo.Book.[Description] +=

             CASE

                    WHEN ( Book.BookUnitPrice > 25 AND Book.BookUnitPrice <= 50)

                    THEN ' -- 2nd level book.'

                    WHEN ( Book.BookUnitPrice > 50 AND Book.BookUnitPrice <= 75)

                    THEN ' -- 3rd level book.'

                    WHEN ( Book.BookUnitPrice > 75)

                    THEN ' -- 4th level book.'

                    ELSE ''

        END

GO -- Run the previous command and begins new batch

SELECT  b.BookID,

             b.BookName,

             b.BookUnitPrice,

             b.[Description] AS BookDescription

FROM    dbo.Book b

WHERE   (

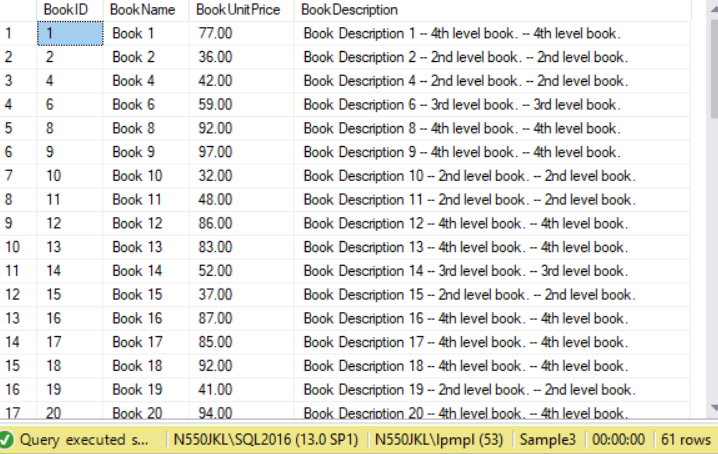
                    ( b.BookUnitPrice > 25 AND b.BookUnitPrice <= 50) OR

                    ( b.BookUnitPrice > 50 AND b.BookUnitPrice <= 75) OR

                    ( b.BookUnitPrice > 75)

        );

GO -- Run the previous command and begins new batch



/\*

1.

Cursor Syntax 2:

--DECLARE @ColumnA1 dataType;

--DECLARE @ColumnA2 dataType;

--DECLARE @ColumnA3 dataType;

--DECLARE @ColumnA4ExtraInfo dataType;

--DECLARE CursorName CURSOR (options...)

--FOR

--    SELECT  b.ColumnA1 ,

--            b.ColumnA2 ,

--            b.ColumnA3 ,

--    FROM    dbo.TableName b

--       -- \*\*\* declare a CursorName and help to update the field b.ColumnA4

--       FOR UPDATE OF b.ColumnA4;

--OPEN CursorName;

---- FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3

---- must map to SELECT  a.ColumnA1 ,a.ColumnA2, @ColumnA3

--FETCH NEXT FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           --Set @ColumnA4ExtraInfo = ...

--           ...

--           -- Update [ColumnA4]

--        UPDATE  TableName

--        SET     TableName.[ColumnA4] += (' -- ' + @ColumnA4ExtraInfo)

--             --\*\*\* WHERE   TableName.[ColumnA1] = @ColumnA1;

--        WHERE CURRENT OF CursorName;

--           --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

--           FETCH NEXT FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

--    END;

--CLOSE CursorName;

--DEALLOCATE CursorName;

1.1.

CURSOR is a way to step through a set of records one row at a time.

It is like a pointer in each record and moving through one step at a time.

--OPEN CursorName;

-- FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3

Open the Cursor and point to the (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) data set

must map to SELECT  a.ColumnA1 ,a.ColumnA2, @ColumnA3

If you have selected extra Columns,

then you have to declare extra variable to map them.

In this case,

declare @ColumnA1, @ColumnA2, ColumnA3   must map to  --SELECT  b.ColumnA1 ,b.ColumnA2, b.ColumnA3

Read the value for each record set and put into variables @ColumnA1, @ColumnA2, @ColumnA3

--WHILE @@FETCH\_STATUS = 0

means successfully read a next record into variables @ColumnA1, @ColumnA2, @ColumnA3

during while loop, you can use  @ColumnA1, @ColumnA2, @ColumnA3

to update  TableName.ColumnA4 or any other sql statement.

----FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3

Use the Cursor and point to the (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) data set,

--WHILE @@FETCH\_STATUS = 0

check if successfully read a (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) record

into variables @ColumnA1, @ColumnA2, @ColumnA3

This while loop will run until NOT(@@FETCH\_STATUS = 0)

That means the pointer reach to the end of loop and get out the loop.

1.2.

--DECLARE CursorName CURSOR (options...)

--FOR

--    SELECT  ...

--OPEN CursorName;

--FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3

--...

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           ...

--    END;

--CLOSE CursorName;

--DEALLOCATE CursorName;

1.2.1.

This is the life of CURSOR

--DECLARE CursorName CURSOR (options...)

--FOR

Then

--OPEN CursorName;

--FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3

Then

--WHILE @@FETCH\_STATUS = 0

Then

--CLOSE CursorName;

Then

--DEALLOCATE CursorName;

means get rid of the CursorName.

-------------------------------------------------

2.

--DECLARE @BookID INT;

--DECLARE @BookName NVARCHAR(100);

--DECLARE @BookUnitPrtice MONEY;

--DECLARE @DescriptionExtraInfo NVARCHAR(100);

--DECLARE BookCursor CURSOR

--FOR

--    SELECT  b.BookID ,

--            b.BookName ,

--            b.BookUnitPrice

--    FROM    dbo.Book b

--       -- \*\*\* declare a CURSOR "BookCursor" and help to update the field [Book].[Description]

--       FOR UPDATE OF b.[Description];

--OPEN BookCursor;

---- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

----INTO @BookID, @BookName, @BookUnitPrtice;    must map to

----SELECT  b.BookID , b.BookName, b.BookUnitPrice

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--             --set @DescriptionExtraInfo

--        SELECT  @DescriptionExtraInfo =

--                  CASE

--                         WHEN ( @BookUnitPrtice > 25 AND @BookUnitPrtice <= 50)

--                THEN '2nd level book.'

--                WHEN ( @BookUnitPrtice > 50 AND @BookUnitPrtice <= 75)

--                THEN '3rd level book.'

--                WHEN ( @BookUnitPrtice > 75 )

--                THEN '4th level book.'

--                ELSE ''

--            END;

--             -- Update [Description]

--        UPDATE  dbo.Book

--        SET     dbo.Book.[Description] += (' -- ' + @DescriptionExtraInfo)

--             --\*\*\* WHERE   Book.BookID = @BookID;

--        WHERE CURRENT OF BookCursor;

--             --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

--             FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

--    END;

--CLOSE BookCursor;

--DEALLOCATE BookCursor;

2.1.

--DECLARE BookCursor CURSOR

--FOR

--    SELECT  b.BookID ,

--            b.BookName ,

--            b.BookUnitPrice

--    FROM    dbo.Book b

--     FOR UPDATE OF b.[Description];

--OPEN BookCursor;

---- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

----INTO @BookID, @BookName, @BookUnitPrtice;    must map to

----SELECT  b.BookID , b.BookName, b.BookUnitPrice

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

2.1.1.

CURSOR is a way to step through a set of records one row at a time.

It is like a pointer in each record and moving through one step at a time.

2.1.2.

--DECLARE BookCursor CURSOR

--FOR

--    SELECT  b.BookID ,

--            b.BookName ,

--            b.BookUnitPrice

--    FROM    dbo.Book b

--     FOR UPDATE OF b.[Description];

declare a CURSOR "BookCursor" to read the values for each record set.

This select statement will help to

update the field [Book].[Description]

2.1.3.

--OPEN BookCursor;

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

Open the Cursor and point to the first data set,

Read the value and put into variables @BookID and @BookName, @BookUnitPrtice

which need to map to   SELECT  b.BookID , b.BookName, b.BookUnitPrice

2.2.

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--             --set @DescriptionExtraInfo = ...

--        ...

--             -- Update [Description]

--        UPDATE  dbo.Book

--        SET     dbo.Book.[Description] += (' -- ' + @DescriptionExtraInfo)

--        WHERE CURRENT OF BookCursor;

--             --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

--             FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

--    END;

2.2.1.

--WHILE @@FETCH\_STATUS = 0

means successfully read a next record into variables @BookID, @BookName, @BookUnitPrtice.

during while loop, you can use  @BookID, @BookName, @BookUnitPrtice

to update Book.[Description] or any other sql statement.

FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

Use the Cursor and point to the next data set,

Read the value and put into variables @BookID, @BookName, @BookUnitPrtice.

--WHILE @@FETCH\_STATUS = 0

check if successfully read a next record into variables @BookID, @BookName, @BookUnitPrtice.

This while loop will run until NOT(@@FETCH\_STATUS = 0)

That means the pointer reach to the end of loop and get out the loop.

2.3.

-- DECLARE BookCursor CURSOR

--FOR

--    SELECT  ...

--       FOR UPDATE OF b.[Description];

--OPEN BookCursor;

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

--...

--WHILE @@FETCH\_STATUS = 0

--    BEGIN

--           ...

--    END;

--CLOSE BookCursor;

--DEALLOCATE BookCursor;

2.3.1.

This is the life of CURSOR

-- DECLARE BookCursor CURSOR

--FOR

--    SELECT  ...

--       FOR UPDATE OF b.[Description];

Then

--OPEN BookCursor;

Then

--FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

Then

--WHILE @@FETCH\_STATUS = 0

Then

--CLOSE BookCursor;

Then

--DEALLOCATE BookCursor;

means get rid of the CURSOR.

\*/

==================================================

10. Update TableB.ColumnB4 with  TablAACursor

--=========================================================

--T021\_10\_Update TableB.ColumnB4 with  TablAACursor

--Replace TablAACursor by using  Update TableB...From TableB Join Join TableA .... WHERE...

--=========================================================

/\*

Goal:

1.

Update TableB.ColumnB4 with  TablAACursor

2.

dbo.Book b has b.BookID, b.BookName, b.BookUnitPrice.

Depending on b.BookUnitPrice, we need to update dbo.BookShoppingCart.[Description]

3.

Curson is very bad in Performance.

Thus, Replace TablAACursor by using  Update TableB...From TableB Join Join TableA .... WHERE...

4.

See the comment in Ch63toCh64\_08

\*/

--Update TableB.ColumnB4 with  TablAACursor

DECLARE @BookID INT;

DECLARE @BookName NVARCHAR(100);

DECLARE @BookUnitPrtice MONEY;

DECLARE @DescriptionExtraInfo NVARCHAR(100);

DECLARE BookCursor CURSOR

FOR

    SELECT  b.BookID ,

            b.BookName ,

            b.BookUnitPrice

    FROM    dbo.Book b

       --FOR UPDATE OF BookShoppingCart.[Description];

       ----Error: The multi-part identifier "BookShoppingCart.Description" could not be bound.

       --When Update TableB.ColumnB4 with  TablAACursor,

       --we can not use   TablAACursor "FOR UPDATE OF "

OPEN BookCursor;

-- \*\*\* FETCH (NEXT/First/LAST/ABSOLUTE 9/ABSOLUTE -1) FROM CursorName INTO @ColumnA1,

--INTO @BookID, @BookName, @BookUnitPrtice;    must map to

--SELECT  b.BookID , b.BookName, b.BookUnitPrice

FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

WHILE @@FETCH\_STATUS = 0

    BEGIN

             --set @DescriptionExtraInfo

        SELECT  @DescriptionExtraInfo =

                    CASE

                           WHEN ( @BookUnitPrtice > 25 AND @BookUnitPrtice <= 50)

                           THEN ' -- 2nd level book Sale.'

                           WHEN ( @BookUnitPrtice > 50 AND @BookUnitPrtice <= 75)

                           THEN ' -- 3rd level book Sale.'

                           WHEN ( @BookUnitPrtice > 75)

                           THEN ' -- 4th level book Sale.'

                           ELSE ''

            END;

             PRINT @DescriptionExtraInfo

             -- Update [Description]

        UPDATE  dbo.BookShoppingCart

        SET     dbo.BookShoppingCart.[Description] += (' -- ' + @DescriptionExtraInfo)

             --WHERE CURRENT OF BookCursor;

             ----Error: The cursor does not include the table being modified or the table is not updatable through the cursor.

             --When Update TableB.ColumnB4 with  TablAACursor,

             --we can not update use "WHERE CURRENT OF TablAACursor;"

             WHERE   BookShoppingCart.BookID = @BookID;

             --\*\*\* FETCH (NEXT/PRIOR/RELATIVE 10/RELATIVE -10) FROM CursorName INTO @ColumnA1, @ColumnA2, @ColumnA3;

             FETCH NEXT FROM BookCursor INTO @BookID, @BookName, @BookUnitPrtice;

    END;

CLOSE BookCursor;

DEALLOCATE BookCursor;

GO -- Run the previous command and begins new batch

SELECT  b.BookID,

             b.BookName,

             b.BookUnitPrice,

             b.[Description] AS BookDescription,

             bsc.BookShoppingCartID,

             bsc.CreatedDateTime,

             bsc.Quantity,

             bsc.[Description] AS BookShoppingCartDescription

FROM    dbo.Book b

        JOIN dbo.BookShoppingCart bsc ON b.BookID = bsc.BookID

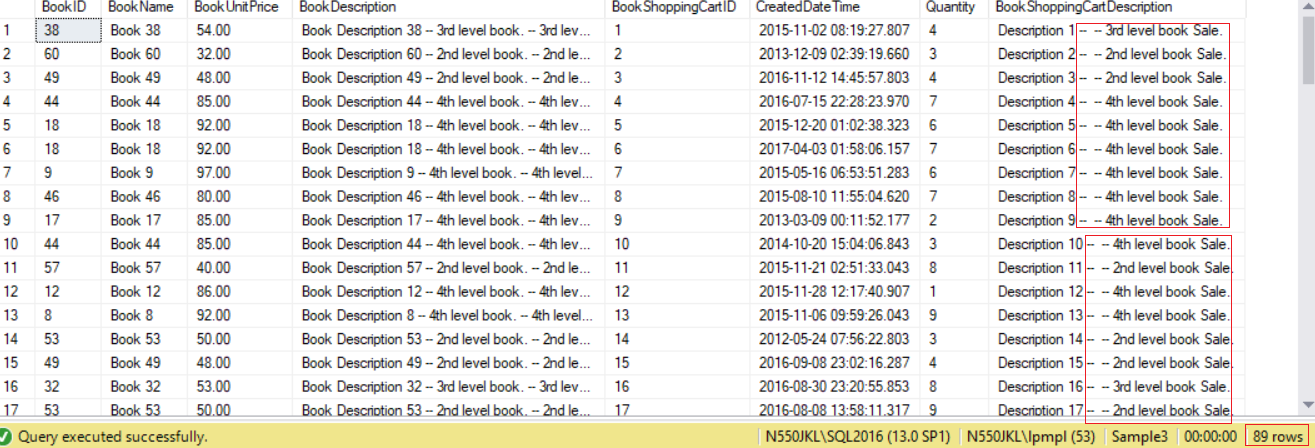
WHERE   (

                    ( b.BookUnitPrice > 25 AND b.BookUnitPrice <= 50) OR

                    ( b.BookUnitPrice > 50 AND b.BookUnitPrice <= 75) OR

                    ( b.BookUnitPrice > 75)

        );



------------------------------------------------

--Replace TablAACursor by using  Update TableB...From TableB Join Join TableA .... WHERE...

UPDATE  dbo.BookShoppingCart

SET     dbo.BookShoppingCart.[Description] +=

             CASE

                    WHEN ( b.BookUnitPrice > 25 AND b.BookUnitPrice <= 50)

                    THEN ' -- 2nd level book Sale.'

                    WHEN ( b.BookUnitPrice > 50 AND b.BookUnitPrice <= 75)

                    THEN ' -- 3rd level book Sale.'

                    WHEN ( b.BookUnitPrice > 75)

                    THEN ' -- 4th level book Sale.'

                    ELSE ''

        END

FROM    dbo.BookShoppingCart bsc

        JOIN dbo.Book b ON bsc.BookID = b.BookID

WHERE   (

                    ( b.BookUnitPrice > 25 AND b.BookUnitPrice <= 50) OR

                    ( b.BookUnitPrice > 50 AND b.BookUnitPrice <= 75) OR

                    ( b.BookUnitPrice > 75)

        );

GO -- Run the previous command and begins new batch

SELECT  b.BookID,

             b.BookName,

             b.BookUnitPrice,

             b.[Description] AS BookDescription,

             bsc.BookShoppingCartID,

             bsc.CreatedDateTime,

             bsc.Quantity,

             bsc.[Description] AS BookShoppingCartDescription

FROM    dbo.Book b

        JOIN dbo.BookShoppingCart bsc ON b.BookID = bsc.BookID

WHERE   (

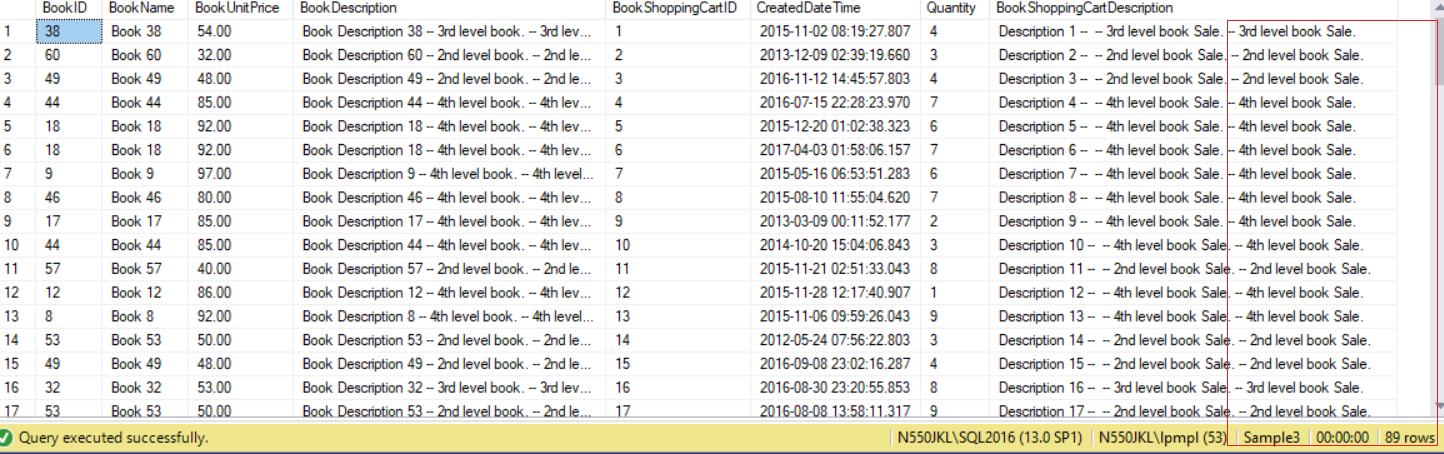
                    ( b.BookUnitPrice > 25 AND b.BookUnitPrice <= 50) OR

                    ( b.BookUnitPrice > 50 AND b.BookUnitPrice <= 75) OR

                    ( b.BookUnitPrice > 75)

        );

GO -- Run the previous command and begins new batch



==================================================

11. Clean up

--=========================================================

--T021\_11\_Clean up

--=========================================================

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookShoppingCart' ) )

    BEGIN

        DROP TABLE BookShoppingCart;

    END;

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Book' ) )

    BEGIN

        DROP TABLE Book;

    END;