(T20)大量Data的Performance。比較Sub-Query和Join  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T20)大量Data的Performance。比較Sub-Query和Join  
=======================================================================  
0. Summary

1. Create Sample Data

-----------

2. Get the book that has never been sold

2.1. GET the book that has never been sold - SubQuery

2.2. Get the book that has never been sold - JOIN

-----------

3. CorrelatedSubquery V.S. NonCorrelatedSubquery

3.1. non-corelated sub-query

3.2. corelated sub-query

-----------

4. PerformanceTesting

4.1. Create large amount of data

4.2. Select ...

-----------

5. SubQuery V.S. JoinsPerformance

5.1. Compare Join V.S. SubQuery

5.2. Compare Join V.S. SubQuery

-----------

6. Clean up  
=======================================================================

0. Summary

1.

CHECKPOINT;

GO

-- Clears query cache

DBCC DROPCLEANBUFFERS;

GO

-- Clears execution plan cache

DBCC FREEPROCCACHE;

GO

2.

Random Number

2.1.

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.

2.2.

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

3.

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

4.

Theoretically, joins is faster than sub-queries.

In reality, SQL Server always transforms query on an execution plan.

If sql server generates the same execution plan from both queries,

then it will return the same result.

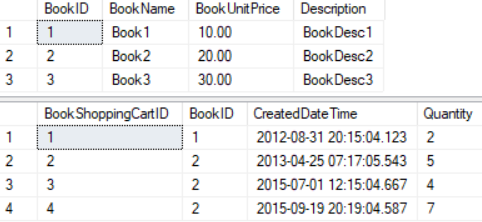
It is always better to do real testing and make a decision.

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

1. Create Sample Data

Graphical user interface

Description automatically generated with medium confidence



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

--T020\_01\_Create Sample Data

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

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'BookShoppingCart' ) )

    BEGIN

        TRUNCATE TABLE BookShoppingCart;

        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

        TRUNCATE TABLE Book;

        DROP TABLE Book;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE 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

INSERT  INTO Book

VALUES  ( 'Book1', 10, 'BookDesc1' );

INSERT  INTO Book

VALUES  ( 'Book2', 20, 'BookDesc2' );

INSERT  INTO Book

VALUES  ( 'Book3', 30, 'BookDesc3' );

GO -- Run the previous command and begins new batch

CREATE TABLE BookShoppingCart

    (

      BookShoppingCartID INT PRIMARY KEY

                                 IDENTITY(1, 1)

                                 NOT NULL ,

      BookID INT FOREIGN KEY REFERENCES Book ( [BookID] )

                     NOT NULL ,

      CreatedDateTime DATETIME NULL ,

      Quantity INT NULL,

    )

GO -- Run the previous command and begins new batch

INSERT  INTO BookShoppingCart

VALUES  ( 1, '2012-08-31 20:15:04.123', 2 );

INSERT  INTO BookShoppingCart

VALUES  ( 2, '2013-04-25 07:17:05.543', 5 );

INSERT  INTO BookShoppingCart

VALUES  ( 2, '2015-07-01 12:15:04.667', 4 );

INSERT  INTO BookShoppingCart

VALUES  ( 2, '2015-09-19 20:19:04.588', 7 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    Book;

SELECT  \*

FROM    BookShoppingCart;

GO -- Run the previous command and begins new batch

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

2. Get the book that has never been sold

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

--T020\_02\_Get the book that has never been sold

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

2.1. GET the book that has never been sold - SubQuery

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

--T020\_02\_01

--GET the book that has never been sold - SubQuery

SELECT  b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

WHERE   b.BookID NOT IN ( SELECT DISTINCT

                                    bsc.BookID

                          FROM      BookShoppingCart bsc );

GO -- Run the previous command and begins new batch

2.2. Get the book that has never been sold - JOIN

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

--T020\_02\_02

--Get the book that has never been sold - JOIN

SELECT  b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

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

WHERE   bsc.BookID IS NULL;

GO -- Run the previous command and begins new batch

/\*

Reference:

[https://technet.microsoft.com/en-us/library/ms189575(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms189575%28v=sql.105%29.aspx)

subqueries can be nested upto 32 levels.

\*/

Graphical user interface, text, application

Description automatically generated

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

3. CorrelatedSubquery V.S. NonCorrelatedSubquery

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

--T020\_03\_CorrelatedSubquery V.S. NonCorrelatedSubquery

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

3.1. non-corelated sub-query

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

--T020\_03\_01

--non-corelated sub-query

SELECT  b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

WHERE   b.BookID NOT IN ( SELECT DISTINCT

                                    bsc.BookID

                          FROM      BookShoppingCart bsc );

GO -- Run the previous command and begins new batch

/\*

A non-corelated sub-query can be executed independently.

E.g.

--SELECT DISTINCT bsc.BookID

--FROM   BookShoppingCart bsc

\*/



3.2. corelated sub-query

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

--T020\_03\_02

--corelated sub-query

SELECT  b.BookID ,

        b.BookName ,

        ( SELECT    SUM(bsc.Quantity)

          FROM      BookShoppingCart bsc

          WHERE     b.BookID = bsc.BookID

        ) AS TotalOrderQuantity

FROM    Book b

ORDER BY b.BookName;

GO -- Run the previous command and begins new batch

/\*

A corelated sub-query can NOT be executed independently,

because sub-query depends on the value of outer query.

E.g.

--SELECT    SUM(bsc.Quantity)

--FROM      BookShoppingCart bsc

--WHERE     b.BookID = bsc.BookID

\*/

Graphical user interface, table

Description automatically generated

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

4. PerformanceTesting

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

--T020\_04\_PerformanceTesting

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

4.1. Create large amount of data

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

--T020\_04\_01

--Create large amount of data

--T020\_04\_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 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 BookShoppingCart

(

  BookShoppingCartID INT PRIMARY KEY

                         IDENTITY(1, 1)

                         NOT NULL ,

  BookID INT FOREIGN KEY REFERENCES Book ( [BookID] )

             NOT NULL ,

  CreatedDateTime DATETIME NULL ,

  Quantity INT NULL,

 );

GO -- Run the previous command and begins new batch

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

--T020\_04\_01\_02

--Insert to Book

--Whole T020\_04\_01 part need to execute together.

--Book Counter

DECLARE @TotalBookRows INT = 300000;

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 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

\*/

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

--T020\_04\_01\_02

--Insert sample data to [BookShoppingCart] table

--Whole T020\_04\_01 part need to execute together.

--BookShoppingCart Counter

DECLARE @TotalBookShoppingCartRows INT;

DECLARE @BookShoppingCartCount INT;

SET @BookShoppingCartCount = 1;

SET @TotalBookShoppingCartRows = 400000;

-- @RandomBookID

DECLARE @RandomBookID INT;

DECLARE @RandomBookID\_Max INT;

DECLARE @RandomBookID\_Min INT;

SET @RandomBookID\_Min = 1;

SET @RandomBookID\_Max = @TotalBookRows - 100;

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

--but I purposely set  @RandomBookID\_Max = @TotalBookRows-100

--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 BookShoppingCart

        VALUES  ( @RandomBookID, @RandomCreatedDateTime, @RandomQuantity );

        PRINT @BookShoppingCartCount;

        SET @BookShoppingCartCount += 1;

    END;

GO -- Run the previous command and begins new batch

4.2. Select ...

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

--T020\_04\_02

SELECT  \*

FROM    Book;

SELECT  \*

FROM    BookShoppingCart;

GO -- Run the previous command and begins new batch

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

5. SubQuery V.S. JoinsPerformance

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

--T020\_05\_SubQuery V.S. JoinsPerformance

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

5.1. Compare Join V.S. SubQuery

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

--T020\_05\_01

--Compare Join V.S. SubQuery

SELECT  b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

WHERE   b.BookID IN ( SELECT    bsc.BookID

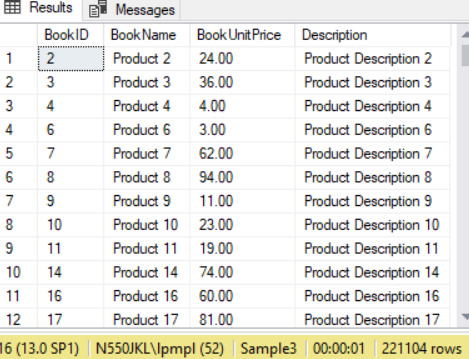
                      FROM      BookShoppingCart bsc );

GO -- Run the previous command and begins new batch

/\*

Run 221073 rows in 1 second.

\*/



CHECKPOINT;

GO -- Run the previous command and begins new batch

-- Clears query cache

DBCC DROPCLEANBUFFERS;

GO -- Run the previous command and begins new batch

-- Clears execution plan cache

DBCC FREEPROCCACHE;

GO -- Run the previous command and begins new batch

SELECT DISTINCT

        b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

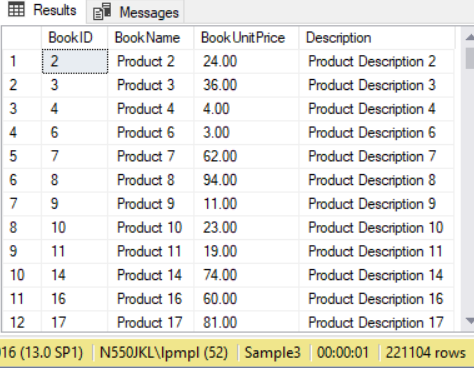
        INNER JOIN BookShoppingCart bsc ON b.BookID = bsc.BookID;

GO -- Run the previous command and begins new batch

/\*

Run 221073 rows in 1 second.

\*/



CHECKPOINT;

GO -- Run the previous command and begins new batch

-- Clears query cache

DBCC DROPCLEANBUFFERS;

GO -- Run the previous command and begins new batch

-- Clears execution plan cache

DBCC FREEPROCCACHE;

GO -- Run the previous command and begins new batch

5.2. Compare Join V.S. SubQuery

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

--T020\_05\_02

--Compare Join V.S. SubQuery

/\*

Theoretically, joins is faster than sub-queries.

In reality, SQL Server always transforms query on an execution plan.

If sql server generates the same execution plan from both queries,

then it will return the same result.

It is alwys better to do real testing and make a decision.

\*/

SELECT  b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

WHERE   b.BookID NOT IN ( SELECT    bsc.BookID

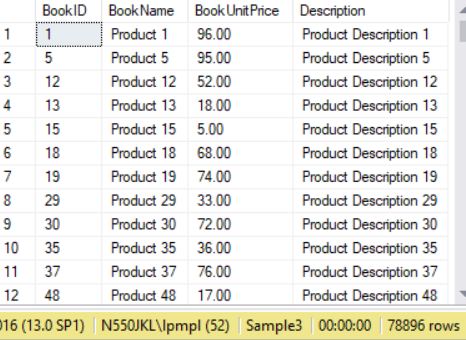
                      FROM      BookShoppingCart bsc );

GO -- Run the previous command and begins new batch

/\*

Run 78927 rows less than 1 second.

\*/



CHECKPOINT;

GO -- Run the previous command and begins new batch

-- Clears query cache

DBCC DROPCLEANBUFFERS;

GO -- Run the previous command and begins new batch

-- Clears execution plan cache

DBCC FREEPROCCACHE;

GO -- Run the previous command and begins new batch

SELECT DISTINCT

        b.BookID ,

        b.BookName ,

        b.BookUnitPrice ,

        b.[Description]

FROM    Book b

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

WHERE   bsc.BookID IS NULL;

GO -- Run the previous command and begins new batch

/\*

Run 78927 rows less than 1 second.

\*/

Table, Excel

Description automatically generated

CHECKPOINT;

GO -- Run the previous command and begins new batch

-- Clears query cache

DBCC DROPCLEANBUFFERS;

GO -- Run the previous command and begins new batch

-- Clears execution plan cache

DBCC FREEPROCCACHE;

GO -- Run the previous command and begins new batch

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

6. Clean up

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

--T020\_06\_Clean up

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

--Clean up

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