Bài thực hành số 3

I. Mục đích

Nội dung: Phân tích và tối ưu viết truy vấn SQL

- Phân tích cấu trúc câu lệnh SQL và quan sát việc thực thi câu lênh
- Một số tiện tích của SQL Server hỗ trợ tinh chỉnh hiệu năng thực thi truy vấn

II. Giới thiệu chung

1. Tìm hiểu Execution plan

   Excution plan dùng để

   - Hiện thị kế hoạch thực thi câu truy vấn bằng đồ họa
   - Đọc sơ đồ từ trái qua phải, từ trên xuống dưới
   - Mũi tên không chỉ biểu thị hướng luồng dữ liệu mà còn chỉ rõ tổng số dòng dữ liệu được chuyển từng bước trong tiến trình thực thi
   - Chi phí được thể hiện ở mỗi bước, từ đó đưa ra tổng chi phí ước lược cho thực thi câu truy vấn
   - The cost for the statement can be compared to other statements run in the same batch
   - Chi phí cho mỗi câu lệnh được so sánh với các câu lệnh khác chạy trong cùng 1 lúc.

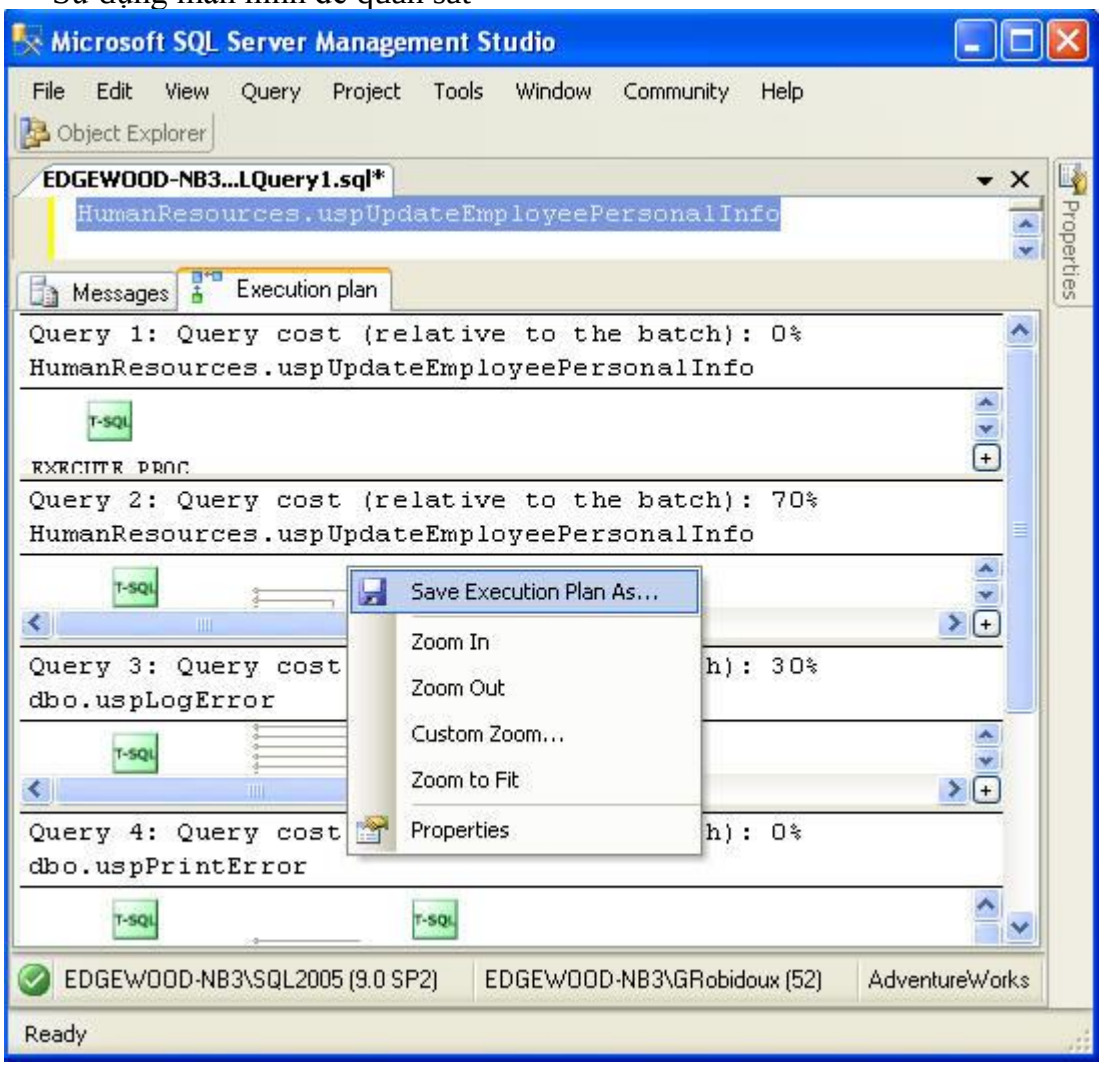
Một số lệnh

	Lệnh	**Giải thích**
1	SHOWPLAN_TEXT	- Theo dõi kết quả tối ưu đơn giản - Cú pháp: SET SHOWPLAN_TEXT **ON** GO <SQL statement>
2	SHOWPLAN_ALL	- Theo dõi kết quả tối ưu chi tiết - Cú pháp: SET SHOWPLAN_ALL **ON** GO <SQL statement>
3	Kế hoạch truy	

	vấn đồ họa	- Quan sát kế hoạch truy vấn đồ họa từ bộ tối ưu SQL Server - Đặc tính này được kích hoạt bằng cách ấn CTRL + K trong bộ phân tích truy vấn

http://msdn.microsoft.com/en-us/library/ms187735.aspx

- Sử dụng màn hình để quan sát



- http://msdn.microsoft.com/en-us/library/ms172984(v=sql.100).aspx
- http://www.sql-server-performance.com/2006/query-execution-plan-analysis/

2. Một số gợi ý cho việc tinh chỉnh câu truy vấn

. *+ Ghi rõ tên cột trong câu lệnh SELECT*

```
SELECT * FROM MyTable;

SELECT ID, Description, DateModified FROM MyTable;
```

Khai báo rõ ràng tên các cột trong câu lệnh SELECT máy chủ SQL sẽ chỉ trả về dữ liệu cần thiết cho ứng dụng chứ không phải là nhiều dữ liệu mà trong đó có nhiều dữ liệu không cần đến. Ngoài ra, nhờ không sử dụng dấu sao (*) nên bạn đã giảm thiểu lưu lượng truyền tải qua mạng (số byte) cần thiết để gửi các dữ liệu liên quan đến câu lệnh SELECT tới ứng dụng.

+ *Ghi rõ tên cột trong câu lệnh INSERT*

Nên chỉ rõ tên từng cột bạn muốn chèn dữ liệu vào trong câu lệnh INSERT. Không nên viết câu lệnh INSERT như sau:

```
INSERT INTO MyTable VALUES (‘A’,'B’,'C’);
```

Với câu lệnh trên, máy chủ SQL đòi hỏi chỉ đúng ba cột được định nghĩa trong bảng MyTable, và giá trị “A” sẽ được chèn vào cột đầu tiên, “B” vào cột thứ hai, “C” vào cột cuối. Nếu ai đó thêm mới một cột vào bảng MyTable, ứng dụng của bạn sẽ bị lỗi:

```
Msg 213, Level 16, State 1, Line 1
Column name or number of supplied values does not match table definition.
(Tên cột hoặc số lượng giá trị không khớp với bảng)
```

Vì thế, thay vì viết câu lệnh INSERT như trên, bạn nên viết như sau:

```
INSERT INTO MyTable(So1, So2, So3) VALUES (‘A’,'B’,'C’);
```

+ *Thêm tiền tố cho wildcard để tăng tốc tìm kiếm*

Sử dụng các ký tự thay thế (wildcard) thích hợp có thể cải thiện hiệu suất câu truy vấn. Ví dụ muốn tìm kiếm trong bảng AdventureWorks.Person.Person tất cả LastNames kết thúc bằng “sen”. Giả dụ rằng đã xây dựng một chỉ mục trên cột LastName. Nếu bạn viết câu lệnh tìm kiếm như sau:

```
SELECT Distinct LastName
FROM Person.Contact
WHERE LastName LIKE ’%sen’
```

Câu lệnh sử dụng ký tự phần trăm (%) để thay thế cho không hoặc nhiều ký tự được theo sau bởi chuỗi “sen” trong trường LastName. Điều này khiến máy chủ SQL thực hiện thao tác quét chỉ mục nhằm tìm kiếm tất cả các tên kết thúc bằng “sen” để giải quyết câu truy vấn. Việc này rất có ý nghĩa bởi cho đến khi toàn bộ bảng được quét, máy chủ SQL không thể đảm bảo rằng đã tìm ra toàn bộ các bản ghi có LastName kết thúc bằng “sen”.

Nếu tìm kiếm các bản ghi có LastName dài đúng sáu ký tự và kết thúc bằng “sen”, bạn có thể viết câu lệnh tìm kiếm như sau:

```
SELECT Distinct LastName
FROM Person.Contact
WHERE LastName LIKE ’___sen’
```

Ở đây, câu lệnh sử dụng ký tự gạch dưới (_) để thay thế cho một ký tự đơn. Máy chủ SQL có thể trả về kết quả nhanh hơn nếu nó không phải đọc toàn bộ chỉ mục bằng cách sử dụng cơ chế quét. Máy chủ SQL đủ thông minh để nhận biết khi bạn đặt thêm tiền tố

trước ký tự thay thế (%, _, v.v..), nó có thể dùng một thao tác tìm kiếm chỉ mục để tiến hành giải quyết tiêu chí tìm kiếm.

+ Chỉ dùng DISTINCT khi cần

Đặt từ khóa DISTINCT trong câu lệnh SELECT sẽ loại bỏ các kết quả trùng lặp trong số những kết quả trả về của câu truy vấn. Do phải thực hiện thêm thao tác SORT để sắp xếp dữ liệu nhằm nhận biết và loại bỏ các bản trùng lặp. Vì thế, nếu biết trước các kết quả trả về sẽ không trùng lặp thì không nên dùng từ khóa DISTINCT trong câu lệnh T-SQL.

+ Chỉ dùng UNION khi cần

Cũng giống như trường hợp từ khóa DISTINCT, toán tử UNION đòi hỏi thêm thao tác SORT để máy chủ SQL có thể loại bỏ những kết quả trùng lặp. Nếu cần dùng toán tử UNION để nối hai tập hợp bản ghi với nhau, trong đó các bản ghi là độc nhất không trùng lặp, tốt hơn bạn nên dùng toán tử UNION ALL.

+ Thủ tục lưu trữ đa năng

Thủ tục lưu trữ đa năng là thủ tục chấp nhận nhiều tham số khác nhau có liên quan đến thủ tục. Dựa trên các tham số được truyền vào, thủ tục lưu trữ đa năng xác định bản ghi nào sẽ được trả về. Sau đây là một ví dụ về thủ tục lưu trữ đa năng:

```
CREATE PROCEDURE JackOfAllTrades

(

@SalesOrderID int = NULL

,@SalesOrderDetailID int = NULL

,@CarrierTrackingNumber nvarchar(25) = NULL

)

AS

SELECT * FROM AdventureWorks.Sales.SalesOrderDetail

WHERE

(SalesOrderID = @SalesOrderID or @SalesOrderID IS NULL)

AND (SalesOrderDetailID = @SalesOrderDetailID or @SalesOrderDetailID IS
NULL)

AND (CarrierTrackingNumber = @CarrierTrackingNumber or
@CarrierTrackingNumber IS NULL)

GO
```

Ở đây SP JackOfAllTrades chấp nhận ba tham số khác nhau. Tất cả các tham số này có giá trị mặc định là NULL. Khi một giá trị được truyền vào, nó sẽ được sử dụng như một tham số trong mệnh đề WHERE để ràng buộc các bản ghi trả về. Mỗi tham số trong SP

được dùng để xây dựng một mệnh đề WHERE phức tạp chứa logic sau đây trong mệnh đề WHERE đối với mỗi tham số truyền vào:

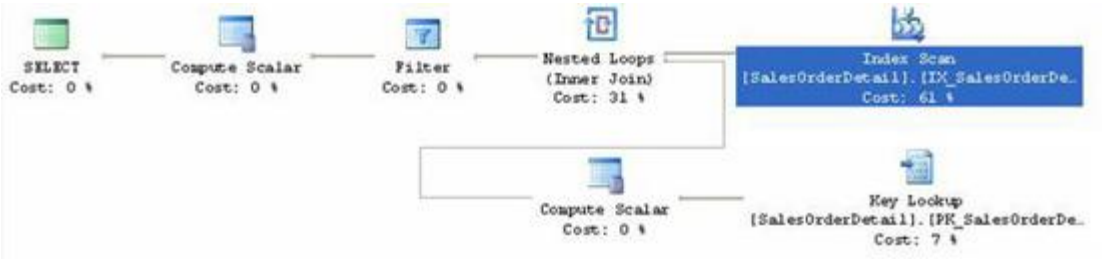
(<TableColumn> = @PARM or @PARM IS NULL)

Logic trên cho biết nếu @PARM được truyền giá trị non-null thì sẽ ràng buộc bản ghi trả về để chắc chắn rằng <TableColumn> bằng giá trị của @PARM. Phần thứ hai của điều kiện đó là "@PARM IS NULL". Phần này có nghĩa nếu @PARM không có giá trị truyền vào (bằng NULL) thì không ràng buộc dữ liệu dựa trên tham số ấy.

Giả sử ta thực thi SP với lệnh sau:

```
EXEC JackOfAllTrades @SalesOrderID = 43659
```

Khi chạy câu lệnh, sơ đồ thực thi trông như sau:



Ở đây bạn có thể thấy đối với mỗi tham số đơn được truyền vào, máy chủ quyết định sử dụng thao tác "quét chỉ mục". Câu lệnh SELECT của SP ràng buộc cột duy nhất @SalesOrderID – một phần của khóa chỉ mục cụm. Với điều kiện "@PARM IS NULL", nó như một hằng số đối với máy chủ SQL. Vì thế máy chủ coi như không có chỉ mục nào hữu ích giúp xử lý điều kiện "(<TableColumn> = @PARM1 or @PARM1 IS NULL)" bởi lẽ hằng số đang ở trong mệnh đề WHERE. Chính vì vậy mà máy chủ SQL quyết định sử dụng thao tác "quét chỉ mục" để giải quyết vấn đề. Thủ tục lưu trữ đa năng càng có nhiều tham số, hiệu suất càng giảm do tác động của số lượng thao tác quét cần thiết cho mỗi tham số truyền vào.

TÓM LẠI:

- Bảng (table) phải có khóa chính (Primary Key).
- Bảng (table) phải có ít nhất 01 clustered index.
- Bảng (table) phải có số lượng non-clustered index phù hợp.
- Non-clustered index phải được tạo trên các cột (column) của bảng (table) dựa vào nhu cầu truy vấn.
- Dựa theo sự sắp xếp thứ tự như sau khi có bất kỳ index được tạo:

a. WHERE clause, b. JOIN clause, c. ORDER BY clause, d. SELECT clause

- Không nên dùng Views thay cho bảng (table) gốc.
- Triggers không nên sử dụng nếu không cần thiết, nên nhập những xử lý từ trigger vào trong thủ tục (stored procedure).
- Gỡ bỏ những câu lệnh query trực tiếp và thay bằng thủ tục (stored procedure).
- Phải có ít nhất 30% không gian đĩa cứng trên phân vùng chứa database.
- Nếu có thể hãy chuyển UDF (user defined function) sang SP (stored procedure) .
- Chỉ SELECT những cột cần thiết, không nên SELECT *.
- Gỡ bỏ các joins từ các bảng (table) không cần thiết.

- Hạn chế sử dụng con trỏ (cursor)
- Đảm bảo phần cứng đáp ứng nhu cầu của hệ thống.

III. Bài tập thực hành

Sử dụng ngôn ngữ SQL để viết các câu truy vấn sau:

1. Lấy ra các đơn hàng có giá trị lớn hơn 10000 và nhỏ hơn 20000.
2. Tính tổng doanh thu của sản phẩm có ProductID là 712 trong tất cả các đơn hàng của mỗi tháng.
3. Thống kê doanh thu bán hàng của từng tháng
4. Thống kê thành phố và tổng giá trị các đơn hàng mà khách hàng ở thành phố đó tiêu thụ.
5. Thống kê số sản phẩm dành cho chỉ nam, chỉ nữ hoặc cho cả nam và nữ.
6. Thực hiện các yêu cầu sau
    a. Giải thích câu truy vấn sau:
        i. Truy vấn 1

```
SELECT CustomerID, SalesOrderID, Sale, RecordNo
FROM   (SELECT CustomerID,
               SalesOrderID,
               TotalDue AS Sale,
               SUM(TotalDue) OVER(PARTITION BY CustomerID) AS SalesTotal,
               ROW_NUMBER() OVER

  (PARTITION BY CustomerID ORDER BY TotalDue DESC) AS RecordNo
        FROM   Sales.SalesOrderHeader) AS soh
WHERE  SalesTotal > 100000
and soh.RecordNo < 6
ORDER BY CustomerID, RecordNo
```

        ii. Truy vấn 2

```
WITH cteTopSales
     AS (SELECT   ROW_NUMBER()
                    OVER(PARTITION BY sod.ProductID
                         ORDER BY SUM(sod.LineTotal) DESC) AS SeqNo,
                  FirstName + ' ' + LastName   AS [Name],
                  ProductName = p.Name, -- SQL Server currency formatting
                  '$' + convert(VARCHAR,convert(MONEY,SUM(sod.LineTotal)),
                                1) AS TotalBySalesPerson,
                  p.ProductNumber,
                  sod.ProductID
         FROM     Sales.SalesOrderDetail AS sod
                  INNER JOIN Production.Product AS p
                    ON sod.ProductID = p.ProductID
                  INNER JOIN Sales.SalesOrderHeader soh
                    ON sod.SalesOrderID = soh.SalesOrderID
                  INNER JOIN Person.Contact c
                    ON soh.SalesPersonID = c.ContactID
         WHERE    soh.SalesPersonID IS NOT NULL
         GROUP BY FirstName + ' ' + LastName,
                  sod.ProductID,
```

```
                  p.ProductNumber,
                  p.Name)
-- Easy-going main query
SELECT *
FROM     cteTopSales cte
WHERE    SeqNo <= 5
ORDER BY ProductID,
         SeqNo
```

b. Thống kê danh sách 5 khách hàng thân thiết của từng năm (khách hàng thân thiết là khách hàng có giá trị đơn hàng lớn nhất).