Bài thực hành số 2

I. Mục đích

Nội dung: Tìm hiểu dữ liệu mẫu Adventure Works, sử dụng index, xem xét hiệu năng thực thi câu truy vấn

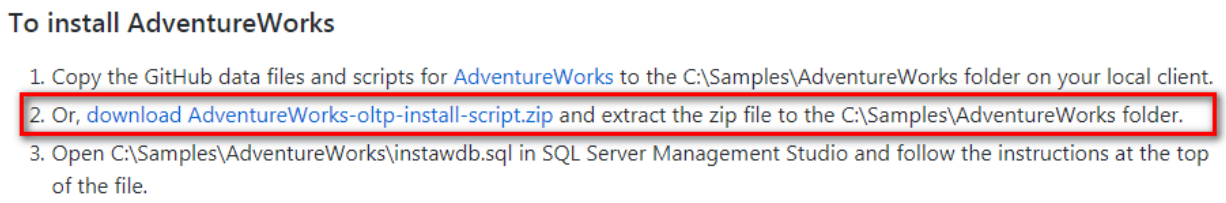
II. Giới thiệu chung

1. Dữ liệu mẫu Adventure Work

a. Download và hướng dẫn cài đặt dữ liệu mẫu AdventureWor

Link: https://github.com/Microsoft/sql-server-samples/blob/master/samples/databases/adventure-works/README.md

Thực hiện:



Trong file nén tải về có file **instawdb.sql** là file script chứa các lệnh để tạo CSDL AdventureWorks. Trong file này có biến môi trường SqlSamplesDatabasePath để trỏ tới thư mục hiện thời. Giá trị biến này sẽ dùng để đưa vào câu lệnh nạp dữ liệu hàng loạt (**bulk load**) được code trong file script instawdb.sql (dữ liệu nguồn để nạp vào hệ thống lấy từ các file .csv kèm trong file nén ban đầu).
Một điểm lưu ý quan trọng nữa là file script instawdb.sql cần phải được thực thi ở chế độ SQLCMD MODE bên trong môi trường SQL Server Management Studio.

Các bước cài đặt:

- Tải về file zip.
- Giải nén toàn bộ vào một thư mục, ví dụ C:\AdventureWorks2017
- Mở file script, ví dụ C:\AdventureWorks2017\instawdb.sql trong SQL Server Management Studio
- Cập nhật giá trị của biến môi trường cho phù hợp, ví dụ thành C:\AdventureWorks2017
- Trên thanh công cụ, chọn Queyr -> SQLCMD Mode
- Thực thi file script

b. Tìm hiểu qua dữ liệu mẫu

Address Table
AddressType Table
AWBuildVersion Table
BillOfMaterials Table
Contact Table
ContactCreditCard Table
ContactType Table
CountryRegion Table
CountryRegionCurrency Table
CreditCard Table
Culture Table
Currency Table
CurrencyRate Table
Customer Table
CustomerAddress Table
DatabaseLog Table
Department Table
Document Table
Employee Table
EmployeeAddress Table
EmployeeDepartmentHistory Table
EmployeePayHistory Table
ErrorLog Table
Illustration Table
Individual Table
JobCandidate Table
Location Table
Product Table
ProductCategory Table
ProductCostHistory Table
ProductDescription Table
ProductDocument Table
ProductModelIllustration Table
ProductModelProductDescriptionCulture Table
ProductPhoto Table
ProductProductPhoto Table
ProductReview Table
ProductSubcategory Table
ProductVendor Table
PurchaseOrderDetail Table
PurchaseOrderHeader Table
SalesOrderDetail Table
SalesOrderHeader Table
SalesOrderHeaderSalesReason Table
SalesPerson Table
SalesPersonQuotaHistory Table
SalesReason Table
SalesTaxRate Table
SalesTerritory Table
SalesTerritoryHistory Table
ScrapReason Table
Shift Table
ShipMethod Table
ShoppingCartItem Table
SpecialOffer Table
SpecialOfferProduct Table
StateProvince Table
Store Table
StoreContact Table
TransactionHistory Table
TransactionHistoryArchive Table
UnitMeasure Table
Vendor Table
VendorAddress Table

ProductInventory Table
ProductListPriceHistory Table
ProductModel Table
VendorContact Table
WorkOrder Table
WorkOrderRouting Table

2. Sử dụng Index

Index là phương tiện rất mạnh để tăng hiệu năng thực hiện của câu lệnh. Ví dụ sử dụng bảng Sale.Customer để tạo ra hai bảng mới là Sale.Customer_noIndex và Sale.Customer_Index, đồng thời tạo 1 index trên trường CustomerID cho bảng Sale.Customer_Index:

Code:

```
SELECT *
INTO Sales.Customer_NoIndex
FROM Sales.Customer

SELECT *
INTO Sales.Customer_Index
FROM Sales.Customer
GO
CREATE INDEX Idx_Customer_Index_CustomerID ON
Sales.Customer_Index(CustomerID)
```

Nay ta có hai câu lệnh SELECT sau để truy vấn hai bảng:

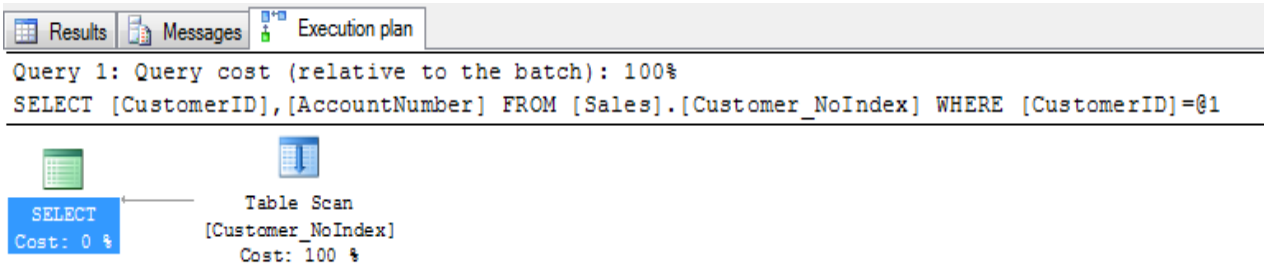
Code:

```
-- #1
SELECT CustomerID, AccountNumber
FROM Sales.Customer_NoIndex
WHERE CustomerID = 11001
-- #2
SELECT CustomerID, AccountNumber
FROM Sales.Customer_Index
WHERE CustomerID = 11001
```

Hai câu lệnh này sẽ cho cùng kết quả, khác biệt duy nhất là câu lệnh thứ hai truy vấn bảng Sales.Customer_Index có index trên trường cần tìm (CustomerID). Ta sẽ xem hai câu lệnh trên được thực hiện như thế nào bằng cách nhìn vào kế hoạch thực thi (execution plan) của chúng. Khi bắt đầu thực hiện một câu lệnh, SQL Server lên một kế hoạch gồm các bước sẽ tiến hành để thực thi câu lệnh đó, gọi là kế hoạch thực thi.**Trên hàng công cụ bạn hãy**

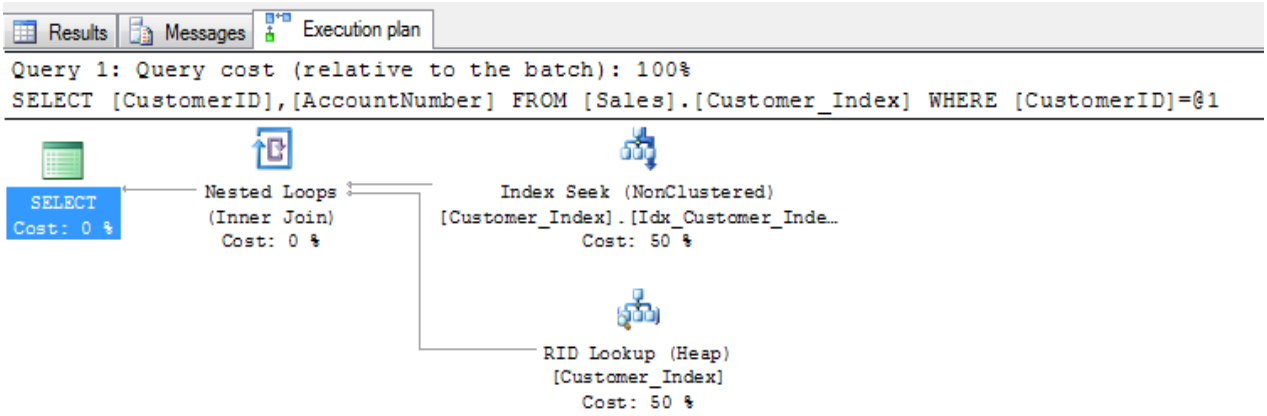
bấm vào nút "Include Actual Execution Plan". Khi đó, mỗi lần chạy câu lệnh hệ thống sẽ vừa thực hiện câu lệnh vừa đồng thời trả lại kế hoạch thực thi mà nó đã dùng để thực hiện câu lệnh đó.

Bôi đen câu lệnh thứ nhất và thực hiện nó, ở tab "Execution plan" hiện ra kế hoạch thực thi như thế này:



Như vậy hệ thống sẽ thực thi câu lệnh bằng cách duyệt qua cả bảng (table scan) và tìm ra các bản ghi thỏa mãn yêu cầu tìm kiếm. Thao tác duyệt bảng có nghĩa là hệ thống cần phải đọc tuần tự từng bản ghi từ đầu đến cuối để tìm ra kết quả. Trong trường hợp này, nó phải đọc toàn bộ 19820 bản ghi và tìm ra bản ghi có CustomerID=11011. Đây là một thao tác rất chậm vì nó phải xử lý tất cả các bản ghi trong bảng. Nên nhớ hệ thống sẽ không dừng lại khi nó tìm được bản ghi đầu tiên có CustomerID=11011, vì nó không biết liệu còn bản ghi nào khác có giá trị CustomerID tương tự hay không, cho nên để chắc chắn trả lại kết quả đầy đủ hệ thống vẫn phải tiếp tục đọc các bản ghi còn lại. Ta có thể nhận xét thấy chi phí của thao tác duyệt bảng tăng tuyến tính cùng với số lượng bản ghi trong bảng (độ phức tạp là O(n)).

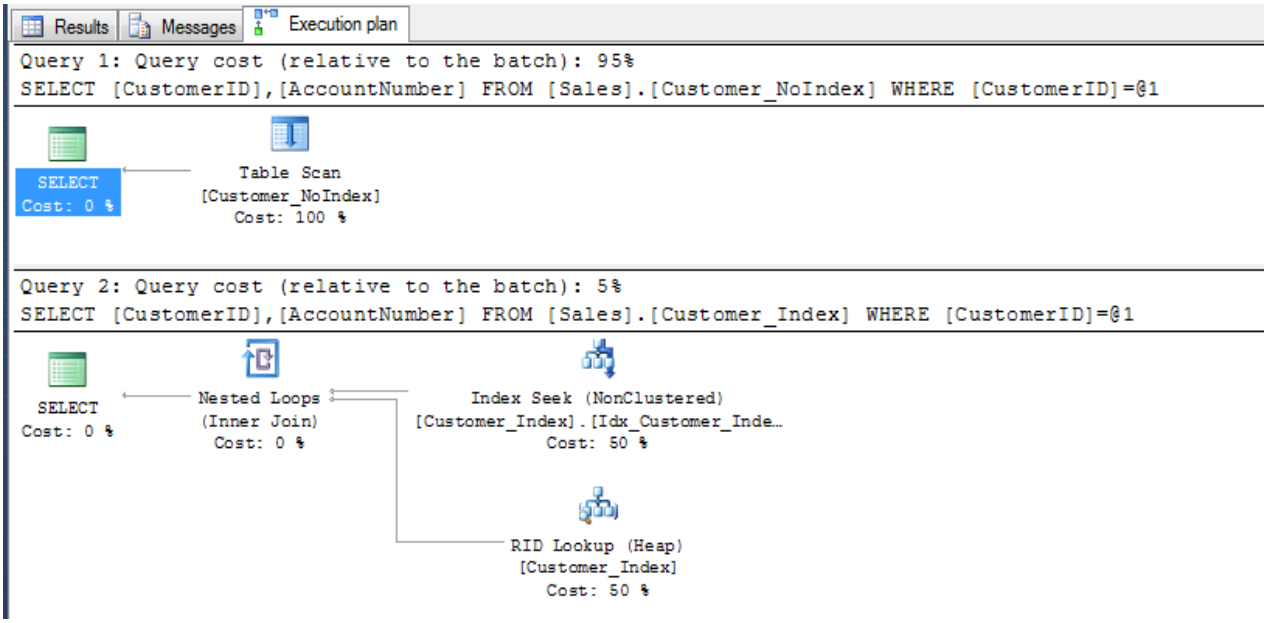
Thực hiện câu lệnh thứ hai, lần này kế hoạch thực thi sẽ như sau:



Lần này không thấy thao tác table scan nữa, mà thay vào đó là index seek và RID lookup. Index seek là khi hệ thống có thể nhảy đến được node trên cây index chứa khóa thỏa mãn yêu cầu tìm kiếm. Index là một cấu trúc dữ liệu có dạng B-tree, nên nó rất thích hợp với các thao tác tìm kiếm theo kiểu key=value, chỉ cần vài phép so sánh là hệ thống định vị

được node chứa khóa cần tìm. Node này chứa khóa (trường được index, ở đây là giá trị của CustomerID) và RID là ID của bản ghi tương ứng trong bảng (đây là giá trị nội bộ chỉ dùng bên trong hệ thống, ta không truy cập được giá trị này). Vì thế bước tiếp theo là dùng RID này để nhảy đến bản ghi tương ứng trong bảng (RID lookup) để lấy các trường dữ liệu cần thiết. Với index seek, độ phức tạp giảm xuống thành O(logn), một bước tiến vượt bậc so với table scan.

Ta có thể so sánh chi phí của hai câu lệnh trên bằng cách thực hiện cả hai cùng nhau:



Ta thấy câu lệnh thứ nhất chiếm tới 95% tổng chi phí, trong khi câu lệnh thứ hai chỉ chiếm có 5%. Nói cách khác, trong trường hợp này index trên trường CustomerID đã giúp cho câu lệnh thực hiện nhanh lên đến 19 lần. Index đã giúp cho lượng dữ liệu hệ thống cần phải xử lý để tìm ra kết quả giảm xuống đến mức tối thiểu, và điều đó đã tạo ra bước nhảy về tốc độ. Từ đây ta rút ra một bài học quan trọng: Các trường thường được dùng trong mệnh đề WHERE là các ứng cử viên đầu tiên cần được tạo index.

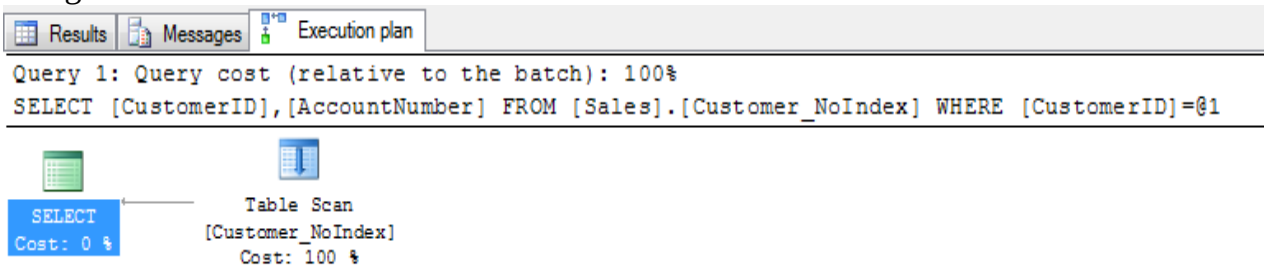
3. Các tiêu chí xem xét hiệu năng của câu truy vấn

Trước khi nghĩ đến việc tăng tốc câu truy vấn thì ta phải xem xét hiệu năng (Performance) của nó. Có 3 tiêu chí chính để chúng ta xem xét đó là: Query Cost (Chi phí của câu truy vấn), Page Reads (Số lượng trang được đọc) và Execution Time (Thời gian thực thi câu truy vấn).

**** Query Cost***

Thông số này càng thấp, câu truy vấn chạy càng nhanh. Thông thường ta có thể thấy được thông số này bằng cách thực thi câu truy vấn thông qua SQL management Studio, chọn Include Actual Execution Plan, khi rê chuột vào Execution Plan, ta thấy xuất hiện

thông số như hình sau:



* ***Page Reads:***

Chính là số lượng trang dữ liệu được truy cập bởi SQL Server khi thực thi câu truy vấn.

Sử dụng hàm: `SET STATISTICS IO { ON | OFF }`

Ví dụ:

```
USE AdventureWorks;
GO
SET STATISTICS IO ON;
GO
SELECT *
FROM Production.ProductCostHistory
WHERE StandardCost < 500.00;
GO
SET STATISTICS IO OFF;
GO
```

Kết quả trả về:

```
(269 row(s) affected)
Table 'ProductCostHistory'. Scan count 1, logical reads 5,
physical reads 0, read-ahead reads 0, lob logical reads 0,
lob physical reads 0, lob read-ahead reads 0.

(1 row(s) affected)
```

Chú giải:

Output item	Meaning
Table	Tên bảng
Scan count	Số lần quét(scan) các bảng(chỉ mục) được thực thi
logical reads	Số trang đọc từ cache dữ liệu

physical reads	Số trang đọc từ ổ đĩa
read-ahead reads	Số trang được đặt vào cache dành cho câu truy vấn
lob logical reads	Số các trang **text**, **ntext**, **image**, hay kiểu dữ liệu lớn (**varchar(max)**, **nvarchar(max)**, **varbinary(max)**) đọc từ cache dữ liệu
lob physical reads	Số các trang **text**, **ntext**, **image**, hay kiểu dữ liệu lớn (**varchar(max)**, **nvarchar(max)**, **varbinary(max)**) đọc từ ổ đĩa
lob read-ahead reads	Số các trang **text**, **ntext**, **image**, hay kiểu dữ liệu lớn (**varchar(max)**, **nvarchar(max)**, **varbinary(max)**) được đặt vào cache cho truy vấn

**** Query Execution Time:***

Tiêu chí này cho ta biết thời gian thực thi câu truy vấn, và nó chịu tác động của blocking(locks) và sự tranh giành tài nguyên của máy chủ .

Sử dụng hàm: `SET STATISTICS TIME { ON | OFF }`

Ví dụ:

```
SET STATISTICS TIME ON
GO
SELECT *
FROM Production.ProductCostHistory
WHERE StandardCost < 500.00;
GO
SET STATISTICS TIME OFF;
GO
```

Kết quả:

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(269 row(s) affected)

(1 row(s) affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 135 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
```

III. Bài tập thực hành

1. Sử dụng dữ liệu mẫu trên thực thi các câu truy vấn sau:

1. Hiển thị chi tiết của tất cả mọi người từ bảng Person.Person
2. Hiển thị Title, FirstName, MiddleName, LastName và EmailAddress từ bảng Person.Contact
3. Hiển thị Title, FirstName, LastName như là một chuỗi nối nhằm dễ đọc và cung cấp tiêu đề cho cột tên (PersonName).
4. Hiển thị chi tiết địa chỉ của tất cả các nhân viên trong bảng Person.Address
5. Liệt kê tên của các thành phố từ bảng Person.Address và bỏ đi phần lặp lại.
6. Hiển thị chi tiết của 10 bảng ghi đầu tiên của bảng Person.Address.
7. Hiển thị trung bình của tỷ giá (Rate) từ bảng HumanResources.EmployeePayHistory.
8. Hiển thị tổng số nhân viên từ bảng HumanResources.Employee
9. Đưa ra danh sách các khách hàng có trên 10 đơn hàng
10. Đưa ra danh sách các mặt hàng chưa từng được đặt hàng
11. Sử dụng tool **Execution Plan** để giải thích các bước xử lý của các câu truy vấn trên
12. Sử dụng index trên 1 bảng nào đấy, xem xét hiệu năng thực thi các câu truy vấn trên bảng đấy.
13. Sử dụng câu truy vấn 10, hãy viết ra 2 đến 3 câu lệnh SQL, đánh giá hiệu năng thực thi giữa các câu lệnh SQL trên