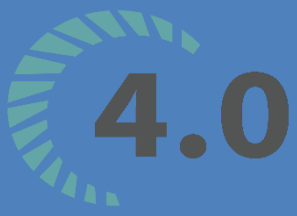
BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

MÔN CƠ SỞ DỮ LIỆU NÂNG CAO



Nhóm thực hiện: 21CLC1.CSDLNC.03

GV phụ trách: Nguyễn Trần Minh Thư

ĐỒ ÁN MÔN HỌC - CƠ SỞ DỮ LIỆU NÂNG CAO

HỌC KỲ I – NĂM HỌC 2023-2024

MỤC LỤC

Contents

YÊU CẦU ĐỒ ÁN- BÀI TẬP

Loại bài tập	**☑ Lý thuyết ☐ Thực hành ☐ Đồ án ☑ Bài tập**
Ngày bắt đầu	**08/12/2023**
Ngày kết thúc	**14/12/2023**

1. Yêu cầu của Đồ án/Bài tập

Hoạt động thiết kế dữ liệu mức vật lý, cho mô hình dữ liệu ở mức logic ORDER ENTRY đạt dạng chuẩn 3. Thực hiện giải quyết các vấn đề.

2. Kết quả

A. Giải pháp điều chỉnh cho mô hình hiện tại để cài đặt vật lý, đáp ứng các mục tiêu truy xuất dữ liệu hiệu quả cho các truy vấn dữ liệu.

1. Đối với một đơn đặt hàng của khách hàng cụ thể, tổng chi phí đơn đặt hàng là bao nhiêu?

Bổ sung thuộc tính **TotalCost** vào bảng Order, với giá trị khởi tạo là 0. Mỗi khi insert vào bảng Ordered_Item, cộng vào TotalCost giá trị bằng $QuantityOrdered \times SellingPrice$ cho Order với OrderNumber thích hợp. Tương tự, nếu update QuantityOrdered hoặc SellingPrice, hay delete trên bảng Ordered_Item thì thực hiện cập nhật cho TotalCost ở bảng Order một cách phù hợp.

2. Đối với một sản phẩm quảng cáo (Advertised_Item) cụ thể, giá thấp nhất mà nhà cung cấp hiện đang cung cấp là bao nhiêu?

Bổ sung thuộc tính **LowestPurchasePrice** vào bảng Advertised_Item, với giá trị khởi tạo là 0. Mỗi khi insert vào vào bảng Restock_Item, cập nhật LowestPurchasePrice của Advertised_Item với ItemNumber tương ứng nếu $PurchasePrice < LowestPurchasePrice$. Tương tự, nếu update PurchasePrice, hay delete trên bảng Restock_Item thì thực hiện cập nhật LowestPurchasePrice ở bảng Advertised_Item một cách phù hợp.

3. Khi thông tin khách hàng được truy xuất, hãy bao gồm tất cả số thẻ tín dụng của họ.

Bổ sung thuộc tính **CustomerIdentifier** – khóa ngoại trỏ đến bảng Customer – vào bảng Credit_Card. Từ đó, khi thông tin khách hàng được truy xuất, chỉ cần tìm thêm trên bảng Credit_Card những thẻ có CustomerIdentifier phù hợp.

4. Giả định bổ sung thuộc tính PreferredOption vào bảng Credit_Card để quản lý thẻ tín dụng yêu thích của khách hàng. Khi thông tin khách hàng truy xuất, cho biết thông tin thẻ tín dụng sử dụng yêu thích của họ.

Bổ sung thuộc tính **PreferredOption** vào bảng Credit_Card với giá trị mặc định là 0, khi khách hàng chọn một thẻ làm thẻ ưa thích thì giá trị này sẽ được đổi thành 1. Đồng thời bổ sung thêm thuộc tính **PreferredCard** vào bảng Customer là khóa ngoại trỏ đến bảng Credit_Card với giá trị ban đầu là NULL. Khi một thẻ được chọn làm thẻ ưa thích của một khách hàng thì giá trị này sẽ được đổi thành mã thẻ.

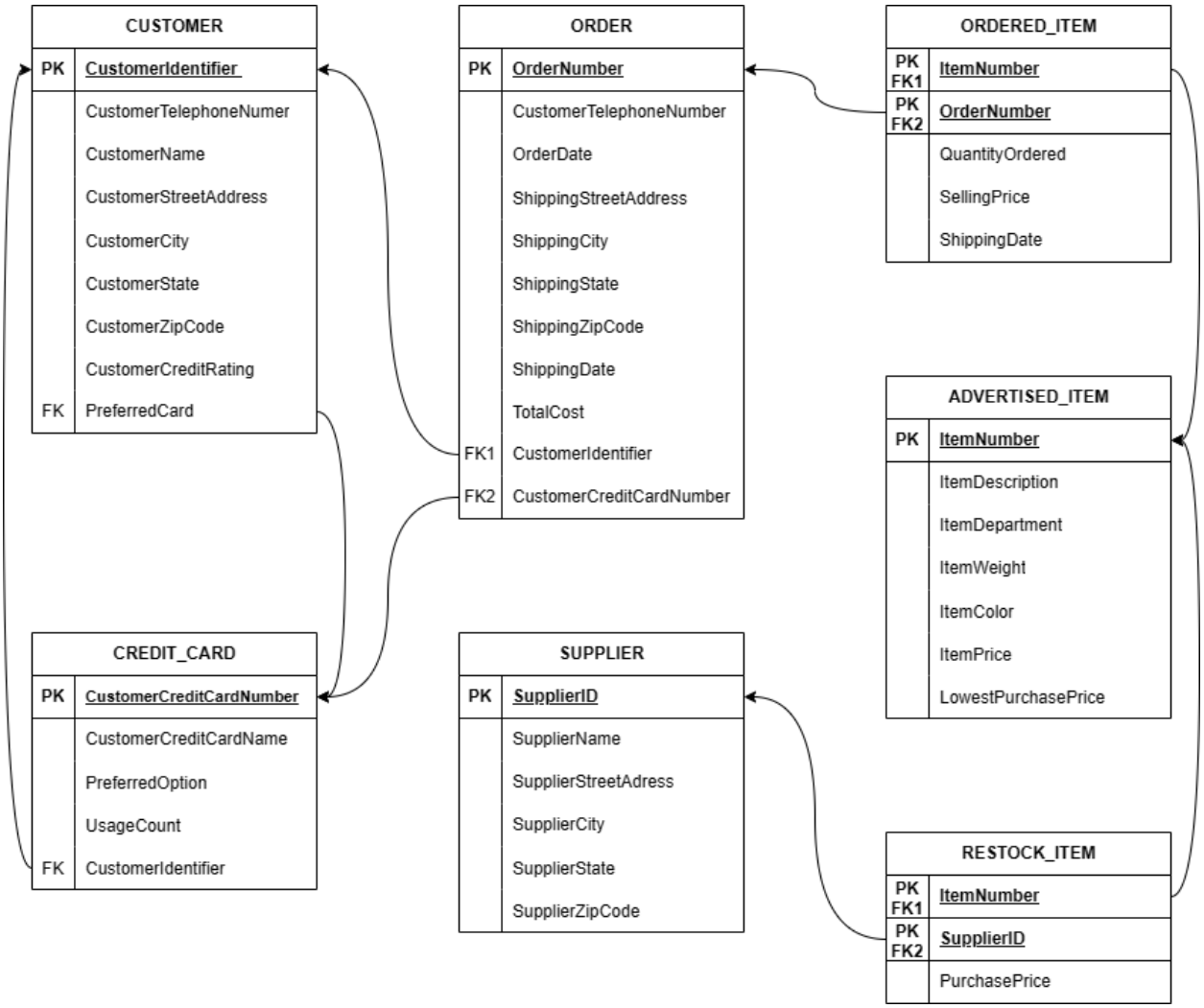
5. Cho biết thông tin khách hàng và số lần sử dụng trên mỗi thẻ tín dụng của họ.

Bổ sung 2 thuộc tính vào bảng Credit_Card:

- **CustomerIdentifier** – Khóa ngoại trỏ đến bảng Customer.
- **UsageCount** – Đếm số lần thẻ này được sử dụng, với giá trị khởi tạo là 0. Mỗi khi insert vào bảng Order, nếu CustomerCreditCardNumber không là NULL thì cộng một vào UsageCount của thẻ có CustomerCreditCardNumber tương ứng ở bảng Credit_Card. Tương tự, nếu update CustomerCreditCardNumber, hay delete trên bảng Order thì thực hiện cập nhật cho UsageCount ở bảng Credit_Card một cách phù hợp.

B. Sau khi có mô hình ORDER ENTRY giải quyết các vấn đề ở A. Hãy kiểm tra và nêu ra tất cả các vấn đề cần lưu ý cũng như ràng buộc dữ liệu cho mô hình dữ liệu mới.

Mô hình mới



Ràng buộc toàn vẹn

1. Tổng chi phí đặt một đơn hàng là tổng giá trị các món hàng thuộc về đơn hàng. Giá trị các món hàng được tính bằng đơn giá nhân với số lượng món hàng được chọn mua.

R1	Thêm	Xóa	Sửa
ORDER	+	-	+(TotalCost)
ORDERED_ITEM	+	+	+(SellingPrice, QuantityOrdered)

2. Giá thấp nhất từ nhà cung cấp của một sản phẩm được quảng cáo là giá thấp nhất của sản phẩm đó ở bảng Restock_Item.

R2	Thêm	Xóa	Sửa
ADVERTISED_ITEM	+	-	+(LowestPurchasePrice)
RESTOCK_ITEM	+	+	+(PurchasePrice)

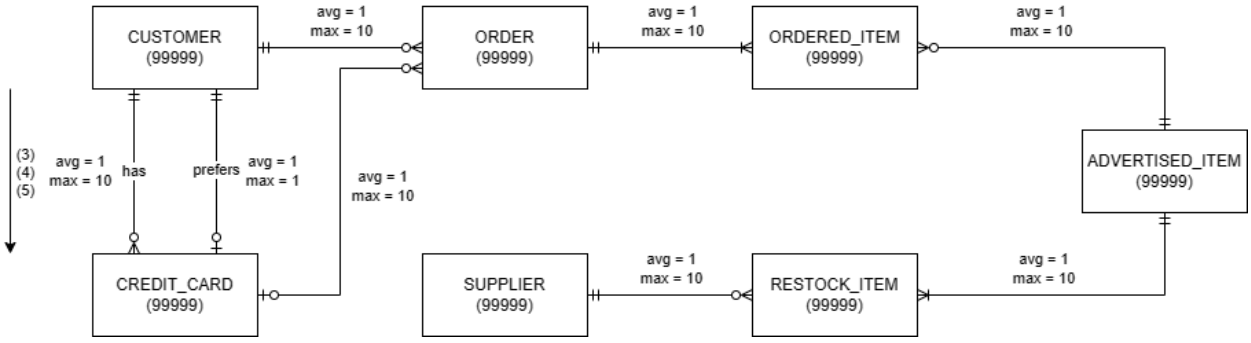
3. Tại một thời điểm, một khách hàng chỉ có một thẻ tín dụng yêu thích.

R3	Thêm	Xóa	Sửa
CREDIT_CARD	+	-	+(PreferredOption)

4. Số lần sử dụng của một thẻ là tổng số lần xuất hiện của thẻ đó ở các đơn hàng.

R4	Thêm	Xóa	Sửa
CREDIT_CARD	+	-	+(UsageCount)
ORDER	+	+	+(CustomerCreditCardNumber)

C. Phân tích ma trận tham chiếu và xây dựng bảng chỉ mục cần phải cài đặt cho các thuộc tính cũng như các bảng dữ liệu.



Truy vấn

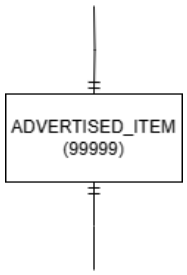
1. *Đối với một đơn đặt hàng của khách hàng cụ thể, tổng chi phí đơn đặt hàng là bao nhiêu?*

```
SELECT CUSTOMER_IDENTIFIER, ORDER_NUMBER, TOTAL_COST FROM ORDERS
WHERE CUSTOMER_IDENTIFIER = 'CS00001' AND ORDER_NUMBER = 'OD00001'
```

Transaction volume • **Average:** 100 per hour • **Peak:** 200 per hour	

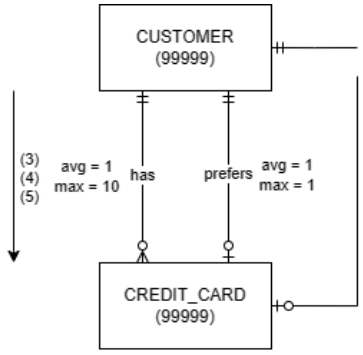
2. *Đối với một sản phẩm quảng cáo (Advertised_Item) cụ thể, giá thấp nhất mà nhà cung cấp hiện đang cung cấp là bao nhiêu?*

```
SELECT ITEM_NUMBER, LOWEST_PURCHASE_PRICE FROM ADVERTISED_ITEM
WHERE ITEM_NUMBER = 'IT00001'
```

Transaction volume • **Average:** 10 per hour • **Peak:** 20 per hour	

3. *Khi thông tin khách hàng được truy xuất, hãy bao gồm tất cả số thẻ tín dụng của họ.*

```
SELECT CS.*, CD.CUSTOMER_CREDIT_CARD_NUMBER, CD.CUSTOMER_CREDIT_CARD_NAME,
    CD.PREFERRED_OPTION, CD.USAGE_COUNT
FROM CUSTOMER CS JOIN CREDIT_CARD CD
    ON CS.CUSTOMER_IDENTIFIER = CD.CUSTOMER_IDENTIFIER
WHERE CS.CUSTOMER_IDENTIFIER = 'CS00001'
```

Transaction volume • **Average:** 100 per hour • **Peak:** 200 per hour	

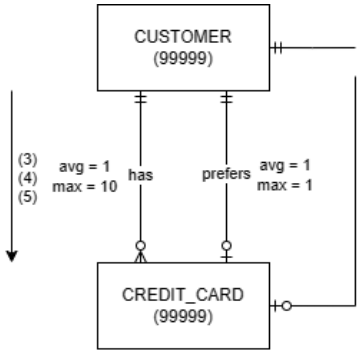
Access	Entity	Type	No. of references		
			Per trans	Avg/hour	Peak/hour
1	CREDIT_CARD (entry)	R	1 – 10	100 – 1000	200 – 2000
Total references			1 – 10	100 – 1000	200 – 2000

4. *Giả định bổ sung thuộc tính PreferredOption vào bảng Credit_Card để quản lý thẻ tín dụng yêu thích của khách hàng. Khi thông tin khách hàng truy xuất, cho biết thông tin thẻ tín dụng sử dụng yêu thích của họ.*

```
SELECT CS.*, CD.CUSTOMER_CREDIT_CARD_NAME, CD.USAGE_COUNT
FROM CUSTOMER CS JOIN CREDIT_CARD CD
```

```
    ON CS.PREFERRED_CARD = CD.CUSTOMER_CREDIT_CARD_NUMBER
WHERE CS.CUSTOMER_IDENTIFIER = 'CS00001'
```

Transaction volume
- **Average:** 100 per hour
- **Peak:** 200 per hour

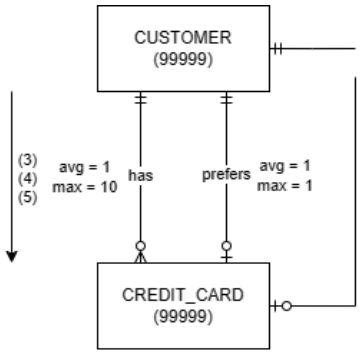


Access	Entity	Type	No. of references		
			Per trans	Avg/hour	Peak/hour
1	CREDIT_CARD (entry)	R	1 – 10	100 – 1000	200 – 2000
Total references			1 – 10	100 – 1000	200 – 2000

5. Cho biết thông tin khách hàng và số lần sử dụng trên mỗi thẻ tín dụng của họ.

```
SELECT CS.*, CD.CUSTOMER_CREDIT_CARD_NUMBER, CD.USAGE_COUNT
FROM CUSTOMER CS JOIN CREDIT_CARD CD
    ON CS.CUSTOMER_IDENTIFIER = CD.CUSTOMER_IDENTIFIER
WHERE CS.CUSTOMER_IDENTIFIER = 'CS00001'
```

Transaction volume
- **Average:** 100 per hour
- **Peak:** 200 per hour



Access	Entity	Type	No. of references		
			Per trans	Avg/hour	Peak/hour
1	CREDIT_CARD (entry)	R	1 – 10	100 – 1000	200 – 2000
Total references			1 – 10	100 – 1000	200 – 2000

Ma trận tham chiếu

Bảng/Query	1				2				3				4				5			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Customer										X				X				X		
Order		X																		
Adv_Item						X														
Supplier																				

Ord_Item																			
Rest_Item																			
Credit_Card										X				X				X		

Xác định index lược đồ

Bảng	Loại	Cột	Lý do
Customer	Clustered	Customer_Identifier	Khóa chính 3, 5: Join 3, 4, 5: Where
	Nonclus	Preferred_Card	Khóa ngoại 4: Join
Order	Nonclus	Order_Number	Khóa chính
	Clustered	Shipping_Date	Chia partition
	(Composite) Covering	(Customer_Identifier, Order_Number) Total_Cost	(1: Where) 1: Select
Advertised_Item	Clustered	Item_Number	Khóa chính 1: Where
Supplier	Clustered	Supplier_ID	Khóa chính
Ordered_Item	Clustered	(Order_Number, Item_Number)	Khóa chính
Restock_Item	Clustered	(Supplier_ID, Item_Number)	Khóa chính
Credit_Card	Clustered	Customer_Credit_Card_Number	Khóa chính
	Nonclus (Covering)	Customer_Identifier (Customer_Credit_Card_Name, Preferred_Option, Usage_Count)	Khóa ngoại 3, 5: Join (3, 4, 5: Select)

D. Hãy nghiên cứu một vài giải pháp thiết kế vật lý hiện nay cho CSDL quan hệ, và thực hiện áp dụng vào bài toán này.

Kỹ thuật Index

Index là một cấu trúc dữ liệu cho phép tăng tốc độ truy xuất đến dòng trên một bảng, đổi lại cần phải tốn thêm chi phí ghi và lưu trữ cấu trúc index. Index có thể được cài trên một hay nhiều cột, hỗ trợ cho sự nhanh chóng trong cả việc tra cứu ngẫu nhiên, và truy cập dữ liệu có thứ tự.

Index có thể được phần thành 2 loại cơ bản là **Clustered** và **Nonclustered**.

- **Clustered Index** lưu trữ và sắp xếp dữ liệu vật lý trong bảng dựa trên giá trị khóa. Mỗi bảng chỉ có một Clustered Index vì bản chất các dòng dữ liệu sẽ bị thay đổi thứ tự vật lý.
- **Nonclustered Index** chứa các giá trị của cột được chọn, và mỗi giá trị được kèm them một con trỏ tới dòng dữ liệu tương ứng trong bảng. Do index này và dữ liệu gốc được lưu tách biệt nhau, nên một bảng có thể có nhiều Nonclustered Index.

Hơn nữa, mỗi loại index trên còn có thể được mở rộng thành nhiều kiểu,

- **Composite Index** là index được cài trên nhiều hơn một cột.
- **Unique Index** đảm bảo tính không trùng lặp về giá trị trên các cột được cài.
- **Covering Index** bao gồm tất cả các cột cần thiết để xử lý một truy vấn cụ thể.

Mặc dù index đóng vai trò quan trọng trong việc tối ưu tốc độ truy vấn, nhưng nó lại tốn thêm bộ nhớ để lưu trữ. Do đó, khi index cần phải tính toán và lưu ý một số vấn đề,

- Không index trên bảng nhỏ.
- Tạo index trên khóa chính nếu nó không phải là khóa để tổ chức dữ liệu.
- Tạo index trên cột thường được truy xuất, hoặc dùng thường xuyên trong các phép select, join, order by, group by, và các thao tác sắp xếp như union hay distinct.
- Tạo index trên cột thường xuất hiện trong các hàm tổng hợp có sẵn.
- Tránh index trên cột hoặc bảng được cập nhật thường xuyên.
- Tránh index trên cột mà truy vấn sẽ trả về phần lớn dữ liệu trên bảng.
- Tránh index trên cột có giá trị là chuỗi kí tự dài.

Áp dụng

Index đã được áp dụng bằng việc cài các index đã xác định được ở mục C.

Kỹ thuật Partition

Partition, hay phân mảnh, là một kĩ thuật giúp phân mảnh một dataset lớn thành nhiều subset nhỏ hơn sao cho máy tính có thể thực hiện việc lưu trữ và xử lý trên các subset này một cách độc lập. Kĩ thuật này giúp giảm bớt gánh nặng lưu trữ với ổ cứng, tăng tốc độ đọc ghi dữ liệu và trên hết là giúp cải thiện hiệu suất xử lý của các truy vấn.

Hiện nay partition được phân làm 2 loại chính,

- **Vertical Partition** – Việc phân mảnh dữ liệu được tiến hành bằng cắt dọc dataset theo các cột.
- **Horizontal Partition** – Việc phân mảnh dữ liệu được tiến hành bằng việc cắt các dataset theo hàng, giữ nguyên tính toàn vẹn của cột. Đây là một kĩ thuật được ưa thích vì các partition vẫn giữ nguyên được số lượng cột và ngữ nghĩa của bản gốc, ngoài ra kĩ thuật này tương đối khá dễ cài.

Có nhiều phương pháp phổ biến được áp dụng để thực hiện phân mảnh ngang như:

- **Key-based Partitioning** – Việc phân từng hàng dữ liệu vào partition được quyết định dựa trên khóa, các hàng dữ liệu thuộc cùng một partition có điểm chung về khóa.
- **Range Partitioning** – Việc phân từng hàng dữ liệu vào partition được quyết định bằng cách xem xét hàng đó thuộc khoảng dữ liệu nào. Phần lớn thời gian việc này được quyết định vào khoảng thời gian mà dữ liệu đó thuộc về (Ví dụ: Hóa đơn của từng năm sẽ được phân vào từng partition khác nhau). Khi đó các hàng dữ liệu trong cùng một partition thuộc về cùng một khoảng thời gian.

- **Hash-based Partitioning** – Việc phân từng hàng dữ liệu vào partition được quyết định dựa trên giá trị hash các thuộc tính của hàng đó. Vì vậy việc phân chia này có tính ngẫu nhiên và thường giúp các partition có số lượng hàng tương đương nhau.

Tuy nhiên partition cũng mang các hạn chế nhất định,

- Việc phân chia các partition cần phải được tính toán kĩ lưỡng ngay từ đầu, các bước cập nhật, thêm và sửa partition tốn rất nhiều chi phí.
- Việc phân chia partition chỉ có ý nghĩa khi các truy vấn được hiện gói gọn trong một partition, nếu các truy vấn làm việc liên quan tới dữ liệu trên nhiều partition khác nhau thì hệ thống phải thực hiện merge các partition lại với nhau để hình thành nên dataset gốc, và khi hệ thống thực hiện merge thì các hàng dữ liệu sẽ được sắp xếp không có thứ tự. Việc này tốn rất nhiều chi phí hệ thống để thực hiện merge và truy vấn trên bảng dataset được ghép.

Áp dụng

Sử dụng kĩ thuật Range-Partitioning, bảng Order có thể được phân mảnh ngang trên thuộc tính ShippingDate. Khi đó, Order sẽ được chia thành các partition khác nhau lưu các đơn hàng đã được giao theo từng năm.

Kỹ thuật viết Query

Có một số kĩ thuật giúp tối ưu hóa các truy vấn như,

- Hạn chế sử dụng phép so sánh LIKE “%” vì khi đó các thuật toán index sẽ không hoạt động hiệu quả do không có cách so sánh các khóa của các dòng dữ liệu với mục tiêu tìm kiếm.
- Sử dụng Indexes để tăng tốc truy vấn, Indexes sẽ thực hiện việc mapping từng đoạn data để các giao tác tìm kiếm có thể hoàn thành nhanh chóng.
- Sử dụng hiệu quả tài nguyên của kiểu dữ liệu, giả sử với dữ liệu là tuổi của một người thì ta chỉ cần sử dụng kiểu dữ liệu tinyInt chứ không cần kiểu dữ liệu Int. Điều này giúp tiết kiệm một lượng khá lớn tài nguyên đối với một bảng dữ liệu có số dòng lớn.
- Tránh viết truy vấn lồng, cố gắng giải quyết yêu cầu của truy vấn bằng các phép truy vấn đơn vì truy vấn lồng sẽ khiến thời gian truy vấn tăng theo cấp số mũ.
- Chỉ truy vấn những dòng dữ liệu cần thiết và chỉ truy vấn những cột dữ liệu cần thiết, giả sử bảng CUSTOMER gồm nhiều thuộc tính nhưng truy vấn của ta chỉ cần tìm tên của khách hàng thì chỉ cần SELECT NAME thay vì SELECT *. Điều này giúp tăng đáng kể hiệu suất của I/O.
- Sử dụng EXISTS thay cho IN, luôn cố gắng sử dụng EXISTS thay cho IN vì điều kiện của Exist chỉ cần xét tồn tại và sẽ trả về kết quả ngay khi tìm thấy 1 dòng thay vì phải kiểm tra thêm một lần nữa trong các kết quả trả về như của IN.

Áp dụng

Những kĩ thuật trên đã được áp dụng trong quá trình viết các truy vấn được yêu cầu.

E. Thực thi khai báo CSDL vào DBMS để khai thác, và kiểm chứng hiệu quả của bảng thiết kế.

Cấu trúc thư mục SQL

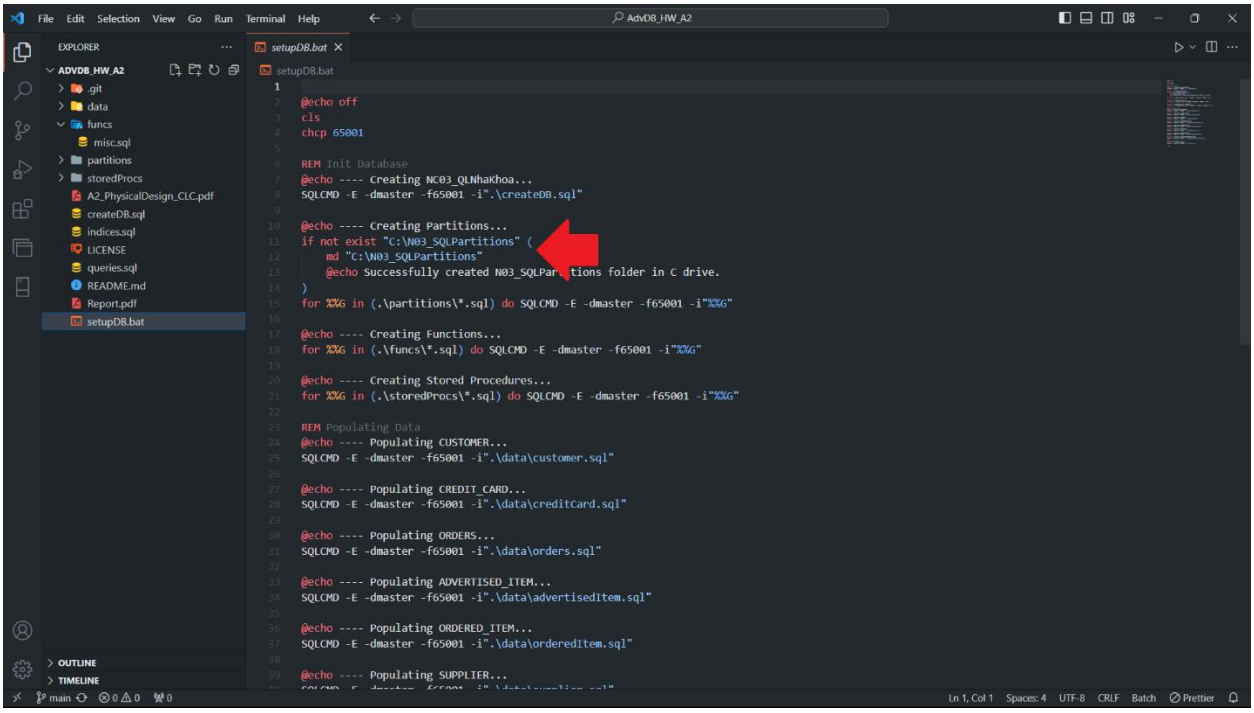
Nhằm phục vụ cho sự dễ dàng trong việc quản lý, hợp tác, và cải tiến trong quá trình cài đặt cơ sở dữ liệu, source code SQL được chia thành nhiều thành phần riêng lẻ nhau.

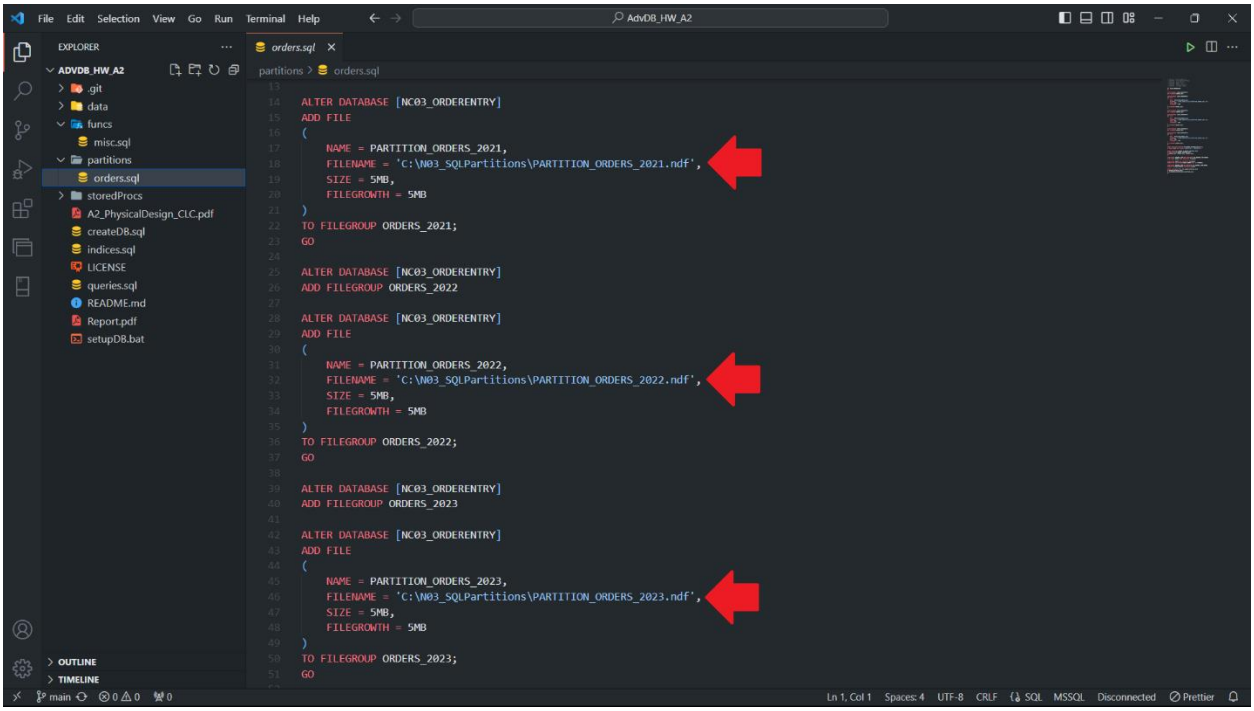
- **createDB.sql** – Script xây dựng database và tạo các bảng.
- **partitions** – Thư mục chứa script partition cho mỗi bảng.
- **funcs** – Thư mục chứa script function cho mỗi bảng, và **misc.sql** cho function dùng chung.
- **storedProcs** – Thư mục chứa script stored procedure cho mỗi bảng, và **misc.sql** cho stored procedure dùng chung.
- **indices.sql** – Script khai báo các index cho database.
- **data** – Thư mục chứa dữ liệu cho mỗi bảng.

Và, đơn giản hóa việc chạy tất cả thành phần trên với **setupDB.bat**. Đây là batch file sử dụng SQLCMD để gọi chạy các script SQL theo thứ tự đã được sắp xếp sẵn.

Lưu ý

1. **Chú ý bật SQL Server** trước khi chạy **setupDB.bat**.
2. Các partition file của database sẽ được lưu tại ổ **C** trong thư mục **N03_SQLPartitions**, tạo bởi **setupDB.bat**. Nếu thay đổi vị trí lưu, cần cập nhật đường dẫn cho **setupDB.bat**, và giá trị của tất cả trường **FILENAME** trong mọi script thuộc thư mục **partitions**.
    - Nếu không cập nhật cho setupDB.bat, cần phải tạo thủ công thư mục lưu mới. **SQL Server sẽ không thực hiện tạo giúp thư mục này nếu nó chưa tồn tại.**





3. Thời gian trung bình để **setupDB.bat** hoàn thành rơi vào từ **45 phút - 1 tiếng**. Việc chạy thủ công từng script là hoàn toàn có thể, tuy nhiên vẫn nên tuân theo thứ tự đã được tính toán trong **setupDB.bat**. Dưới đây là hiển thị cho thấy đã thực hiện xong.

```
C:\Windows\system32\cmd.e:
(1 rows affected)
(1 rows affected)
(1 rows affected)
(0 rows affected)
(1 rows affected)
(1 rows affected)
(1 rows affected)
(1 rows affected)
(1 rows affected)
(0 rows affected)
(1 rows affected)
(1 rows affected)
(0 rows affected)
(1 rows affected)
---- Creating Indices...
Changed database context to 'NC03_ORDERENTRY'.
Press any key to continue . . .
```

Dữ liệu

Việc chọn kiểu dữ liệu phù hợp được thể hiện trong script createDB.sql. Đối với dữ liệu, mỗi bảng được phát sinh 99,999 dòng và các tình huống đã thiết kế được chạy trên chính tập dữ liệu này. Tuy nhiên, sẽ có khác biệt về các giá trị cụ thể so với truy vấn mẫu đã viết ở mục C.

Partition ShippingDate trên bảng Order

Dữ liệu trên bảng này được tạo giới hạn trong năm 2021 và 2022. Do đó, sẽ có 3 partition ORDERS_2021, ORDERS_2022, ORDERS_2023 tương ứng với 3 biểu thức $x < 2022/01/01$, $2022/01/01 \leq x < 2023/01/01$, $x \geq 2023/01/01$.

Results | Messages | Execution plan

	PARTITION_NUMBER	PARTITION_FILEGROUP	NUMBER_OF_ROWS
1	1	ORDERS_2021	69437
2	2	ORDERS_2022	30561
3	3	ORDERS_2023	0

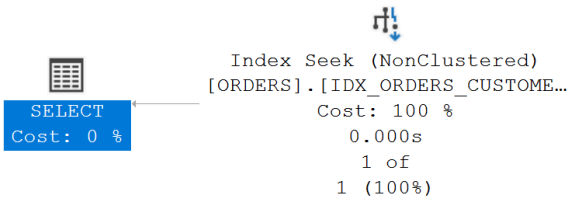
1. Đối với một đơn đặt hàng của khách hàng cụ thể, tổng chi phí đơn đặt hàng là bao nhiêu?

```
SELECT CUSTOMER_IDENTIFIER, ORDER_NUMBER, TOTAL_COST FROM ORDERS
WHERE CUSTOMER_IDENTIFIER = 'CS03392' AND ORDER_NUMBER = 'OD18350'
```

Results | Messages | Execution plan

	CUSTOMER_IDENTIFIER	ORDER_NUMBER	TOTAL_COST
1	CS03392	OD18350	14000

```
Query 1: Query cost (relative to the batch): 100%
SELECT [CUSTOMER_IDENTIFIER],[ORDER_NUMBER],[TOTAL_COST] FROM [ORDERS] WHERE [CUSTOMER_IDENTIFIER]=@1 AND [ORDER_NUMBER]=@2
```



Câu truy vấn tận dụng được index **(CUSTOMER_IDENTIFIER, ORDER NUMBER) INCLUDE TOTAL_COST** thể hiện qua Index Seek.

Không cần dùng Key, hay RID Lookup do các cột cần truy xuất đã được phủ bởi index. Đáp ứng được yêu cầu đề và hầu như không tiêu hao thời gian truy xuất.

Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	1
Actual Number of Rows for All Executions	1
Actual Number of Batches	0
Estimated Operator Cost	0.0098471 (100%)
Estimated I/O Cost	0.009375
Estimated Subtree Cost	0.0098471
Estimated CPU Cost	0.0004721
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	1
Estimated Number of Rows to be Read	1
Estimated Number of Rows Per Execution	1
Estimated Row Size	25 B
Actual Rebinds	0
Actual Rewinds	0
Partitioned	True
Actual Partition Count	1
Ordered	True
Node ID	0

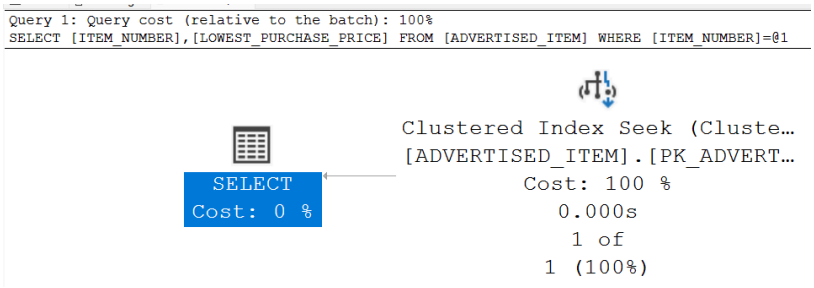
Object
[NC03_ORDERENTRY].[dbo].[ORDERS].
[IDX_ORDERS_CUSTOMER_IDENTIFIER_ORDER_NUMBER]
Output List
[NC03_ORDERENTRY].[dbo].[ORDERS].ORDER_NUMBER,
[NC03_ORDERENTRY].[dbo].[ORDERS].TOTAL_COST,
[NC03_ORDERENTRY].[dbo].[ORDERS].CUSTOMER_IDENTIFIER
Seek Predicates
Seek Keys[1]: Start: PtnId1000 >= Scalar Operator((1)), End: PtnId1000 <= Scalar Operator((3)), Seek Keys[2]: Prefix: [NC03_ORDERENTRY].[dbo].[ORDERS].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[ORDERS].ORDER_NUMBER = Scalar Operator([@1]), Scalar Operator([@2])

2. Đối với một sản phẩm quảng cáo (Advertised_Item) cụ thể, giá thấp nhất mà nhà cung cấp hiện đang cung cấp là bao nhiêu?

```
SELECT ITEM_NUMBER, LOWEST_PURCHASE_PRICE FROM ADVERTISED_ITEM
WHERE ITEM_NUMBER = 'IT70000'
```

Results | Messages | Execution plan

	ITEM_NUMBER	LOWEST_PURCHASE_PRICE
1	IT70000	9000



Với ITEM_NUMBER là khóa chính clustered index làm điều kiện where, đáp ứng được yêu cầu đề và hầu như không tiêu hao thời gian truy xuất.	**Clustered Index Seek (Clustered)** Scanning a particular range of rows from a clustered index. **Physical Operation**: Clustered Index Seek **Logical Operation**: Clustered Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 1 **Actual Number of Rows for All Executions**: 1 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032831 (100%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032831 **Estimated CPU Cost**: 0.0001581 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1 **Estimated Number of Rows to be Read**: 1 **Estimated Number of Rows Per Execution**: 1 **Estimated Row Size**: 18 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 0 **Object** [NC03_ORDERENTRY].[dbo].[ADVERTISED_ITEM].[PK_ADVERTISED_ITEM] **Output List** [NC03_ORDERENTRY].[dbo].[ADVERTISED_ITEM].ITEM_NUMBER, [NC03_ORDERENTRY].[dbo].[ADVERTISED_ITEM].LOWEST_PURCHASE_PRICE **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[ADVERTISED_ITEM].ITEM_NUMBER = Scalar Operator([@1])

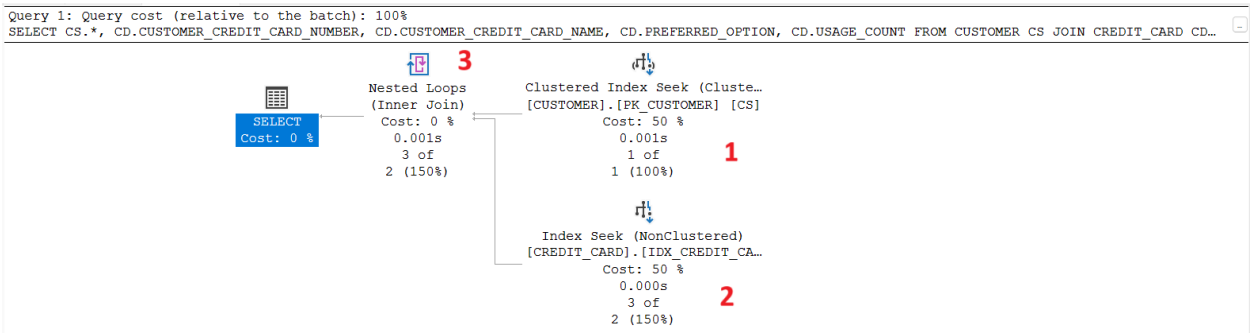
3. Khi thông tin khách hàng được truy xuất, hãy bao gồm tất cả số thẻ tín dụng của họ.

```
SELECT CS.*, CD.CUSTOMER_CREDIT_CARD_NUMBER, CD.CUSTOMER_CREDIT_CARD_NAME,
    CD.PREFERRED_OPTION, CD.USAGE_COUNT
FROM CUSTOMER CS JOIN CREDIT_CARD CD
    ON CS.CUSTOMER_IDENTIFIER = CD.CUSTOMER_IDENTIFIER
WHERE CS.CUSTOMER_IDENTIFIER = 'CS66736'
```

Results | Messages | Execution plan

	CUSTOMER_IDENTIFIER	CUSTOMER_TELEPHONE_NUMBER	CUSTOMER_NAME	CUSTOMER_STREET_ADDRESS	CUSTOMER_CITY	CUSTOMER_STATE	CUSTOMER_ZIP_CODE	CUSTOMER_CREDIT_RATING	PREFERRED_CARD
1	CS66736	84298537006	Ruby Houston	19259 North 61st Drive	Lynn Haven	AZ	85308	63	CD99950
2	CS66736	84298537006	Ruby Houston	19259 North 61st Drive	Lynn Haven	AZ	85308	63	CD99950
3	CS66736	84298537006	Ruby Houston	19259 North 61st Drive	Lynn Haven	AZ	85308	63	CD99950

CUSTOMER_CREDIT_CARD_NUMBER	CUSTOMER_CREDIT_CARD_NAME	PREFERRED_OPTION	USAGE_COUNT
CD59270	Ruby Houston	0	2
CD85334	Ruby Houston	0	1
CD99950	Ruby Houston	1	0



1	Với CUSTOMER_IDENTIFIER là khóa chính clustered index làm điều kiện where, truy xuất được một dòng thỏa yêu cầu CUSTOMER_IDENTIFIER = 'CS66736', mà hầu như không tiêu hao thời gian.	**Clustered Index Seek (Clustered)** Scanning a particular range of rows from a clustered index. **Physical Operation**: Clustered Index Seek **Logical Operation**: Clustered Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 1 **Actual Number of Rows for All Executions**: 1 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032831 (50%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032831 **Estimated CPU Cost**: 0.0001581 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1 **Estimated Number of Rows to be Read**: 1 **Estimated Number of Rows Per Execution**: 1 **Estimated Row Size**: 450 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 1 **Object** [NC03_ORDERENTRY].[dbo].[CUSTOMER].[PK_CUSTOMER] [CS] **Output List** [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_TELEPHONE_NUMBER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_NAME, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STREET_ADDRESS, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CITY, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STATE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_ZIP_CODE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CREDIT_RATING, [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER = Scalar Operator('CS66736')

2	Câu truy vấn tận dụng được index **CUSTOMER_IDENTIFIER INCLUDE(CUSTOMER_CREDIT_CARD_NAME, PREFERRED_OPTION, USAGE_COUNT)** thể hiện qua Index Seek tìm được 3 dòng thỏa yêu cầu CUSTOMER_IDENTIFIER = 'CS66736'. Không cần dùng Key, hay RID Lookup do các cột cần truy xuất đã được phủ bởi index. Đáp ứng được yêu cầu đề và hầu như không tiêu hao thời gian truy xuất.	**Index Seek (NonClustered)** Scan a particular range of rows from a nonclustered index. **Physical Operation**: Index Seek **Logical Operation**: Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 3 **Actual Number of Rows for All Executions**: 3 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032837 (50%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032837 **Estimated CPU Cost**: 0.0001587 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1.53453 **Estimated Number of Rows to be Read**: 1.53453 **Estimated Number of Rows Per Execution**: 1.53453 **Estimated Row Size**: 123 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 2 **Object** [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].[IDX_CREDIT_CARD_CUSTOMER_IDENTIFIER] [CD] **Output List** [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_CREDIT_CARD_NUMBER, [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_CREDIT_CARD_NAME, [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].PREFERRED_OPTION, [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].USAGE_COUNT **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_IDENTIFIER = Scalar Operator ('CS66736')

3	Với mỗi dòng kết quả từ 1, operator này quét các dòng nhận từ 2 để tìm ra những dòng phù hợp nhau. Sau đó thực hiện hợp nhất mỗi cặp dòng đó thành một dòng dữ liệu và xuất vào Output List.	**Nested Loops** For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows. **Physical Operation**: Nested Loops **Logical Operation**: Inner Join **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Actual Number of Rows for All Executions**: 3 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0000064 (0%) **Estimated I/O Cost**: 0 **Estimated CPU Cost**: 0.0000064 **Estimated Subtree Cost**: 0.0065732 **Number of Executions**: 1 **Estimated Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1.53453 **Estimated Number of Rows Per Execution**: 1.53453 **Estimated Row Size**: 564 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Node ID**: 0 **Output List** [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_TELEPHONE_NUMBER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_NAME, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STREET_ADDRESS, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CITY, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STATE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_ZIP_CODE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CREDIT_RATING, [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD, [NC0...

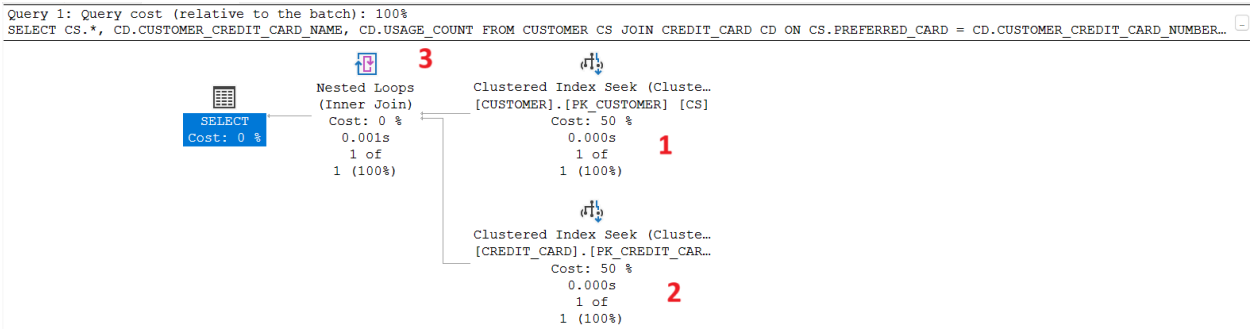
4. Giả định bổ sung thuộc tính PreferredOption vào bảng Credit_Card để quản lý thẻ tín dụng yêu thích của khách hàng. Khi thông tin khách hàng truy xuất, cho biết thông tin thẻ tín dụng sử dụng yêu thích của họ.

```
SELECT CS.*, CD.CUSTOMER_CREDIT_CARD_NAME, CD.USAGE_COUNT
FROM CUSTOMER CS JOIN CREDIT_CARD CD
    ON CS.PREFERRED_CARD = CD.CUSTOMER_CREDIT_CARD_NUMBER
WHERE CS.CUSTOMER_IDENTIFIER = 'CS45531'
```

Results | Messages | Execution plan

	CUSTOMER_IDENTIFIER	CUSTOMER_TELEPHONE_NUMBER	CUSTOMER_NAME	CUSTOMER_STREET_ADDRESS	CUSTOMER_CITY	CUSTOMER_STATE	CUSTOMER_ZIP_CODE
1	CS45531	99713847426	Sam Hogan	2001 Van Hoose Drive	Anchorage	AR	72701

CUSTOMER_CREDIT_RATING	PREFERRED_CARD	CUSTOMER_CREDIT_CARD_NAME	USAGE_COUNT
80	CD70045	Sam Hogan	4



1	Với CUSTOMER_IDENTIFIER là khóa chính clustered index làm điều kiện where, truy xuất được một dòng thỏa yêu cầu CUSTOMER_IDENTIFIER = ‘CS45531’, mà hầu như không tiêu hao thời gian.	**Clustered Index Seek (Clustered)** Scanning a particular range of rows from a clustered index. **Physical Operation**: Clustered Index Seek **Logical Operation**: Clustered Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 1 **Actual Number of Rows for All Executions**: 1 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032831 (50%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032831 **Estimated CPU Cost**: 0.0001581 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1 **Estimated Number of Rows to be Read**: 1 **Estimated Number of Rows Per Execution**: 1 **Estimated Row Size**: 450 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 1 **Object** [NC03_ORDERENTRY].[dbo].[CUSTOMER].[PK_CUSTOMER] [CS] **Output List** [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_TELEPHONE_NUMBER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_NAME, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STREET_ADDRESS, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CITY, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STATE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_ZIP_CODE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CREDIT_RATING, [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER = Scalar Operator('CS45531')

2	Với CUSTOMER_CREDIT_CARD_NUMBER là khóa chính clustered index làm điều kiện join, truy xuất được một dòng thỏa yêu cầu CD. CUSTOMER_CREDIT_CARD_NUMBER = CS.PREFERRED_CARD, mà hầu như không tiêu hao thời gian.	**Clustered Index Seek (Clustered)** Scanning a particular range of rows from a clustered index. **Physical Operation**: Clustered Index Seek **Logical Operation**: Clustered Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 1 **Actual Number of Rows for All Executions**: 1 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032831 (50%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032831 **Estimated CPU Cost**: 0.0001581 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1 **Estimated Number of Rows to be Read**: 1 **Estimated Number of Rows Per Execution**: 1 **Estimated Row Size**: 115 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 2 **Object** [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].[PK_CREDIT_CARD] [CD] **Output List** [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_CREDIT_CARD_NAME, [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].USAGE_COUNT **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_CREDIT_CARD_NUMBER = Scalar Operator ([NC03_ORDERENTRY].[dbo].[CUSTOMER].[PREFERRED_CARD] as [CS].[PREFERRED_CARD])

3	Với mỗi dòng kết quả từ 1, operator này quét các dòng nhận từ 2 để tìm ra những dòng phù hợp nhau. Sau đó thực hiện hợp nhất mỗi cặp dòng đó thành một dòng dữ liệu và xuất vào Output List.	**Nested Loops** For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows. **Physical Operation**: Nested Loops **Logical Operation**: Inner Join **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Actual Number of Rows for All Executions**: 1 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0000042 (0%) **Estimated I/O Cost**: 0 **Estimated CPU Cost**: 0.0000042 **Estimated Subtree Cost**: 0.0065704 **Number of Executions**: 1 **Estimated Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1 **Estimated Number of Rows Per Execution**: 1 **Estimated Row Size**: 556 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Node ID**: 0 **Output List** [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_TELEPHONE_NUMBER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_NAME, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STREET_ADDRESS, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CITY, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STATE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_ZIP_CODE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CREDIT_RATING, [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD, [NC0... **Outer References** [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD

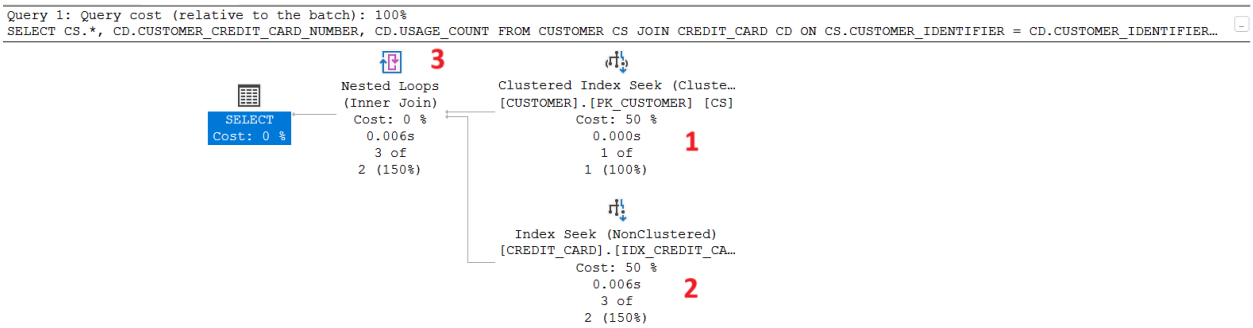
5. Cho biết thông tin khách hàng và số lần sử dụng trên mỗi thẻ tín dụng của họ.

```
SELECT CS.*, CD.CUSTOMER_CREDIT_CARD_NUMBER, CD.USAGE_COUNT
FROM CUSTOMER CS JOIN CREDIT_CARD CD
    ON CS.CUSTOMER_IDENTIFIER = CD.CUSTOMER_IDENTIFIER
WHERE CS.CUSTOMER_IDENTIFIER = 'CS45531'
```

Results | Messages | Execution plan

	CUSTOMER_IDENTIFIER	CUSTOMER_TELEPHONE_NUMBER	CUSTOMER_NAME	CUSTOMER_STREET_ADDRESS	CUSTOMER_CITY	CUSTOMER_STATE	CUSTOMER_ZIP_CODE
1	CS45531	99713847426	Sam Hogan	2001 Van Hoose Drive	Anchorage	AR	72701
2	CS45531	99713847426	Sam Hogan	2001 Van Hoose Drive	Anchorage	AR	72701
3	CS45531	99713847426	Sam Hogan	2001 Van Hoose Drive	Anchorage	AR	72701

CUSTOMER_CREDIT_RATING	PREFERRED_CARD	CUSTOMER_CREDIT_CARD_NUMBER	USAGE_COUNT
80	CD70045	CD13313	0
80	CD70045	CD70045	4
80	CD70045	CD97696	1



1	Với CUSTOMER_IDENTIFIER là khóa chính clustered index làm điều kiện where, truy xuất được một dòng thỏa yêu cầu CUSTOMER_IDENTIFIER = 'CS45531', mà hầu như không tiêu hao thời gian.	**Clustered Index Seek (Clustered)** Scanning a particular range of rows from a clustered index. **Physical Operation**: Clustered Index Seek **Logical Operation**: Clustered Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 1 **Actual Number of Rows for All Executions**: 1 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032831 (50%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032831 **Estimated CPU Cost**: 0.0001581 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1 **Estimated Number of Rows to be Read**: 1 **Estimated Number of Rows Per Execution**: 1 **Estimated Row Size**: 450 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 1 **Object** [NC03_ORDERENTRY].[dbo].[CUSTOMER].[PK_CUSTOMER] [CS] **Output List** [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_TELEPHONE_NUMBER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_NAME, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STREET_ADDRESS, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CITY, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STATE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_ZIP_CODE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CREDIT_RATING, [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER = Scalar Operator('CS45531')

2	Câu truy vấn tận dụng được index **CUSTOMER_IDENTIFIER INCLUDE(CUSTOMER_CREDIT_CARD_NAME, PREFERRED_OPTION, USAGE_COUNT)** thể hiện qua Index Seek tìm được 3 dòng thỏa yêu cầu CUSTOMER_IDENTIFIER = 'CS45531'. Không cần dùng Key, hay RID Lookup do các cột cần truy xuất đã được phủ bởi index. Đáp ứng được yêu cầu đề và hầu như không tiêu hao thời gian truy xuất.	**Index Seek (NonClustered)** Scan a particular range of rows from a nonclustered index. **Physical Operation**: Index Seek **Logical Operation**: Index Seek **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Storage**: RowStore **Actual Number of Rows Read**: 3 **Actual Number of Rows for All Executions**: 3 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.0032839 (50%) **Estimated I/O Cost**: 0.003125 **Estimated Subtree Cost**: 0.0032839 **Estimated CPU Cost**: 0.0001589 **Estimated Number of Executions**: 1 **Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1.68197 **Estimated Number of Rows to be Read**: 1.68197 **Estimated Number of Rows Per Execution**: 1.68197 **Estimated Row Size**: 18 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Ordered**: True **Node ID**: 2 **Object** [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].[IDX_CREDIT_CARD_CUSTOMER_IDENTIFIER] [CD] **Output List** [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_CREDIT_CARD_NUMBER, [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].USAGE_COUNT **Seek Predicates** Seek Keys[1]: Prefix: [NC03_ORDERENTRY].[dbo].[CREDIT_CARD].CUSTOMER_IDENTIFIER = Scalar Operator ('CS45531')

3	Với mỗi dòng kết quả từ 1, operator này quét các dòng nhận từ 2 để tìm ra những dòng phù hợp nhau. Sau đó thực hiện hợp nhất mỗi cặp dòng đó thành một dòng dữ liệu và xuất vào Output List.	**Nested Loops** For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows. **Physical Operation**: Nested Loops **Logical Operation**: Inner Join **Actual Execution Mode**: Row **Estimated Execution Mode**: Row **Actual Number of Rows for All Executions**: 3 **Actual Number of Batches**: 0 **Estimated Operator Cost**: 0.000007 (0%) **Estimated I/O Cost**: 0 **Estimated CPU Cost**: 0.000007 **Estimated Subtree Cost**: 0.006574 **Number of Executions**: 1 **Estimated Number of Executions**: 1 **Estimated Number of Rows for All Executions**: 1.68197 **Estimated Number of Rows Per Execution**: 1.68197 **Estimated Row Size**: 461 B **Actual Rebinds**: 0 **Actual Rewinds**: 0 **Node ID**: 0 **Output List** [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_IDENTIFIER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_TELEPHONE_NUMBER, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_NAME, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STREET_ADDRESS, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CITY, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_STATE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_ZIP_CODE, [NC03_ORDERENTRY].[dbo].[CUSTOMER].CUSTOMER_CREDIT_RATING, [NC03_ORDERENTRY].[dbo].[CUSTOMER].PREFERRED_CARD, [NC0...

Nguồn tham khảo mục D

Kỹ thuật Index

- https://en.wikipedia.org/wiki/Database_index
- https://viblo.asia/p/su-dung-index-trong-sql-query-1ZnbRlPQR2Xo
- **Tài liệu do giảng viên cung cấp**
  - Seminar 2 – index.pdf
  - Chap07 – Physical Design-en

Kỹ thuật Partition

- https://www.geeksforgeeks.org/data-partitioning-techniques/

Kỹ thuật viết Query

- https://medium.com/learning-sql/12-tips-for-optimizing-sql-queries-for-faster-performance-8c6c092d7af1