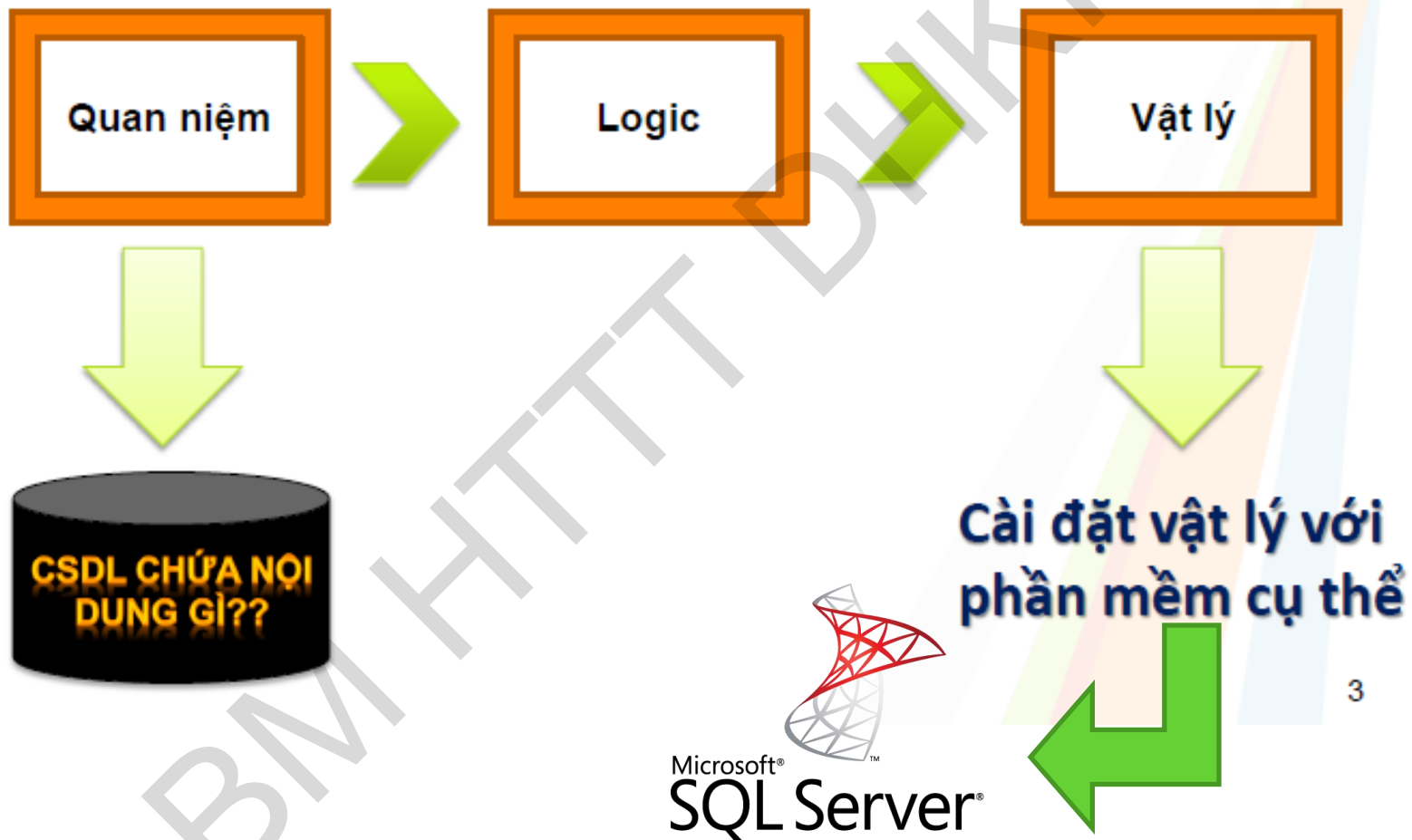




# Seminar: INDEX

Bộ môn HTTT – Khoa CNTT – ĐH KHTN

- 3 mức thiết kế CSDL:



# Physical Design Process

## Inputs

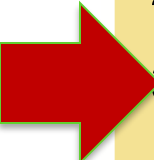
- Normalized relations
- Volume estimates
- Attribute definitions
- Response time expectations
- Data security needs
- Backup/recovery needs
- Integrity expectations
- DBMS technology used
- .....

Leads to

## Decisions

- Attribute data types
- Physical record descriptions (doesn't always match logical design)
- File organizations
- Indexes and database architectures
- Query optimization
- ....

# Cải thiện hiệu năng

- 
1. Optimizing use of existing resources.
  2. Using better or more resources.
  3. Creating indexes.
  4. Denormalizing the DB.
  5. Storing derived data.
  6. Creating procedures to archive data.

.....



**improve DB  
performance**

# Nội dung

1. Định nghĩa
2. Phân loại chỉ mục
3. Execution plan
4. Cài đặt chỉ mục với SQLserver

- ❑ Tổ chức dữ liệu cho phép tăng tốc độ truy xuất đến các dòng trong bảng dữ liệu
- ❑ Được cài đặt trên một hoặc một số cột thuộc tính
  - ➔ *chọn cột thuộc tính cài index?* ➔ *xét tần suất truy xuất*
- ❑ Chứa các giá trị khóa - key values cho phép trở tới các dòng thỏa mãn giá trị khóa này
- ❑ Ví dụ về index
  - Danh mục trong một cuốn sách
  - Index trong từ điển



# INDEX

- ☹ Không index phải duyệt tuần tự toàn bảng
- 😊 Có index chỉ duyệt những dòng index trở tới

## KHÔNG INDEX

→	1	Nguyễn Văn An	1986
→	2	Lê Ngọc Thanh	1987
→	3	Đinh Tiến Long	1988
→	4	Lê Minh Phước	1988
→	5	Hồ Hoàng Lan	1986

## CÓ INDEX

→	1986	→	1	Nguyễn Văn An	1986
			5	Hồ Hoàng Lan	1986
	1987	→	2	Lê Ngọc Thanh	1987
			3	Đinh Tiến Long	1988
	1988	→	4	Lê Minh Phước	1988

# Nội dung

1. Định nghĩa
2. Phân loại chỉ mục trong MSSQL
3. Execution plan
4. Cài đặt chỉ mục với SQLserver



# Phân loại Index trong SQL Server

## CLUSTERED INDEX

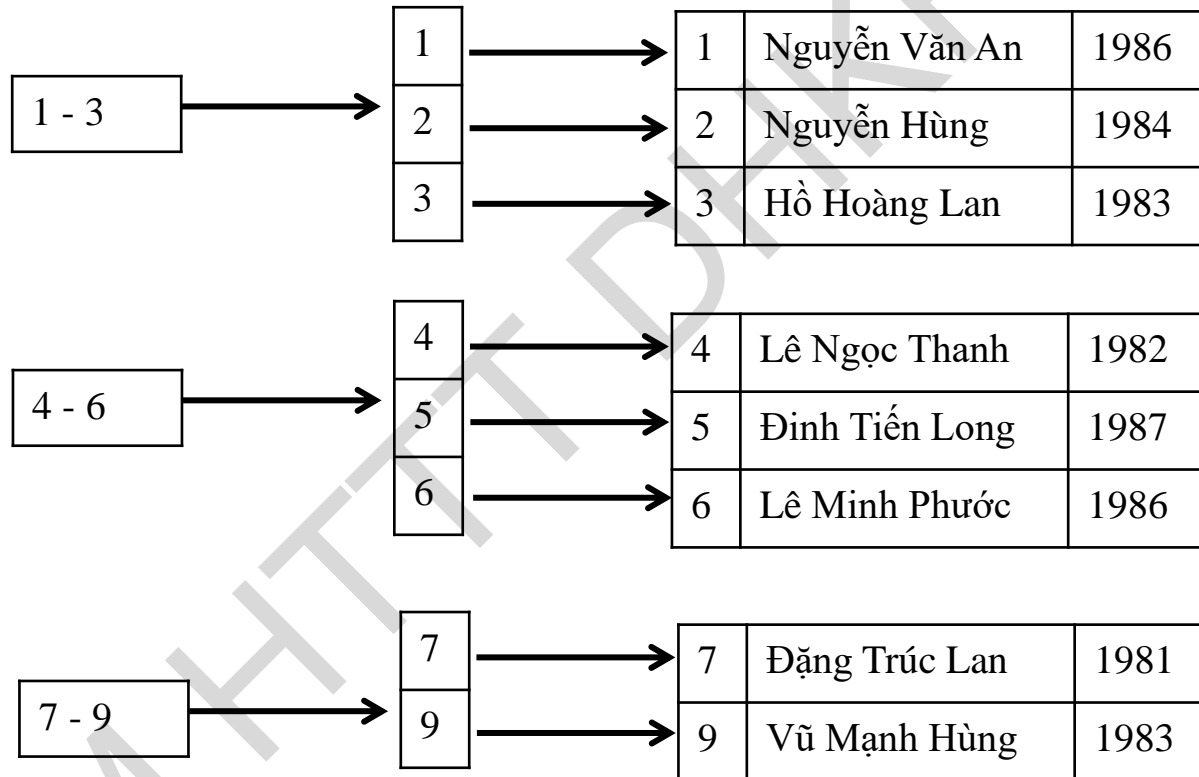
- ☐ Các dòng sắp xếp vật lý theo chỉ mục
- ☐ Chỉ mục trỏ đến dòng dữ liệu cần tìm
- ☐ Mỗi bảng chỉ được có 1 clustered index

## NON - CLUSTERED INDEX

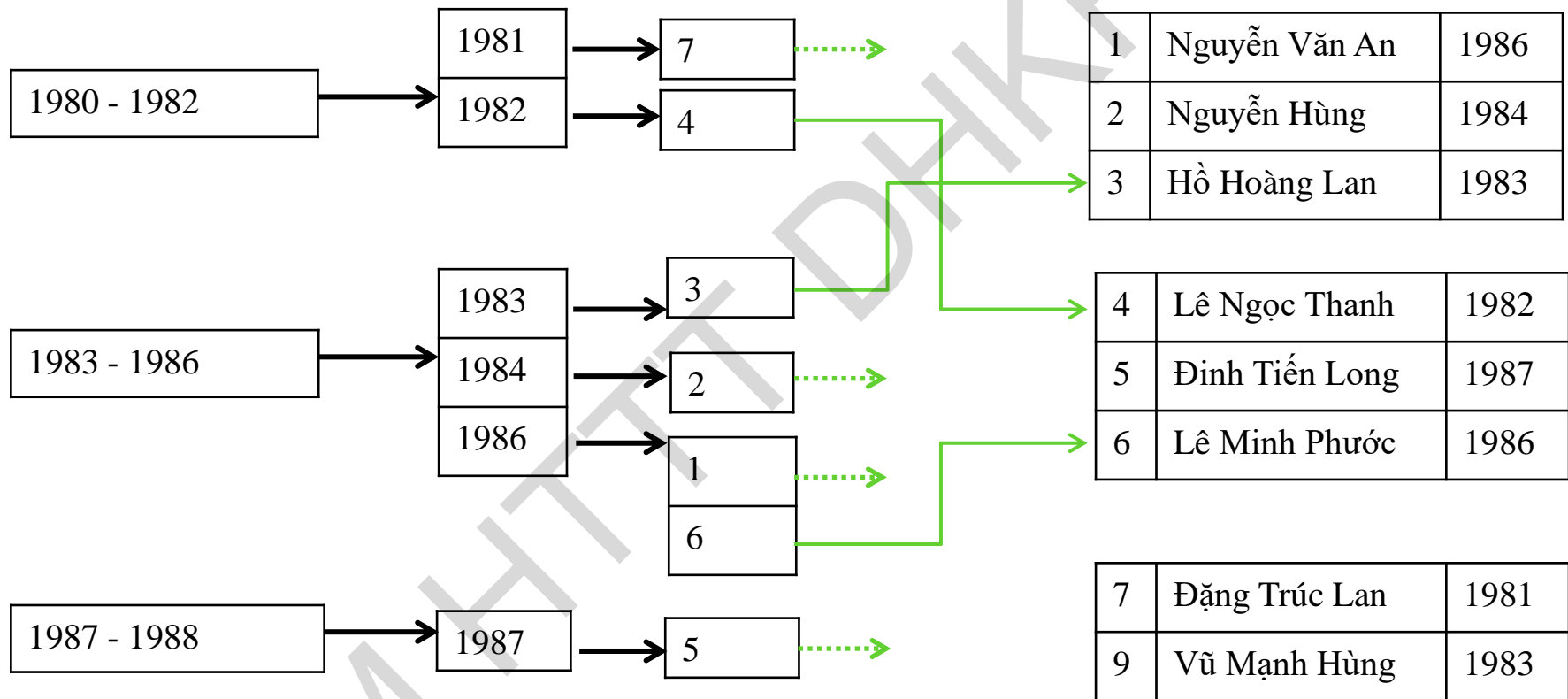
- ☐ Các dòng sắp xếp logic theo chỉ mục
- ☐ Chỉ mục trỏ đến id hoặc clustered index của dòng dữ liệu cần tìm
- ☐ Mỗi bảng có nhiều non-clustered index

**In SQL Server, indexes are organized as B-trees**

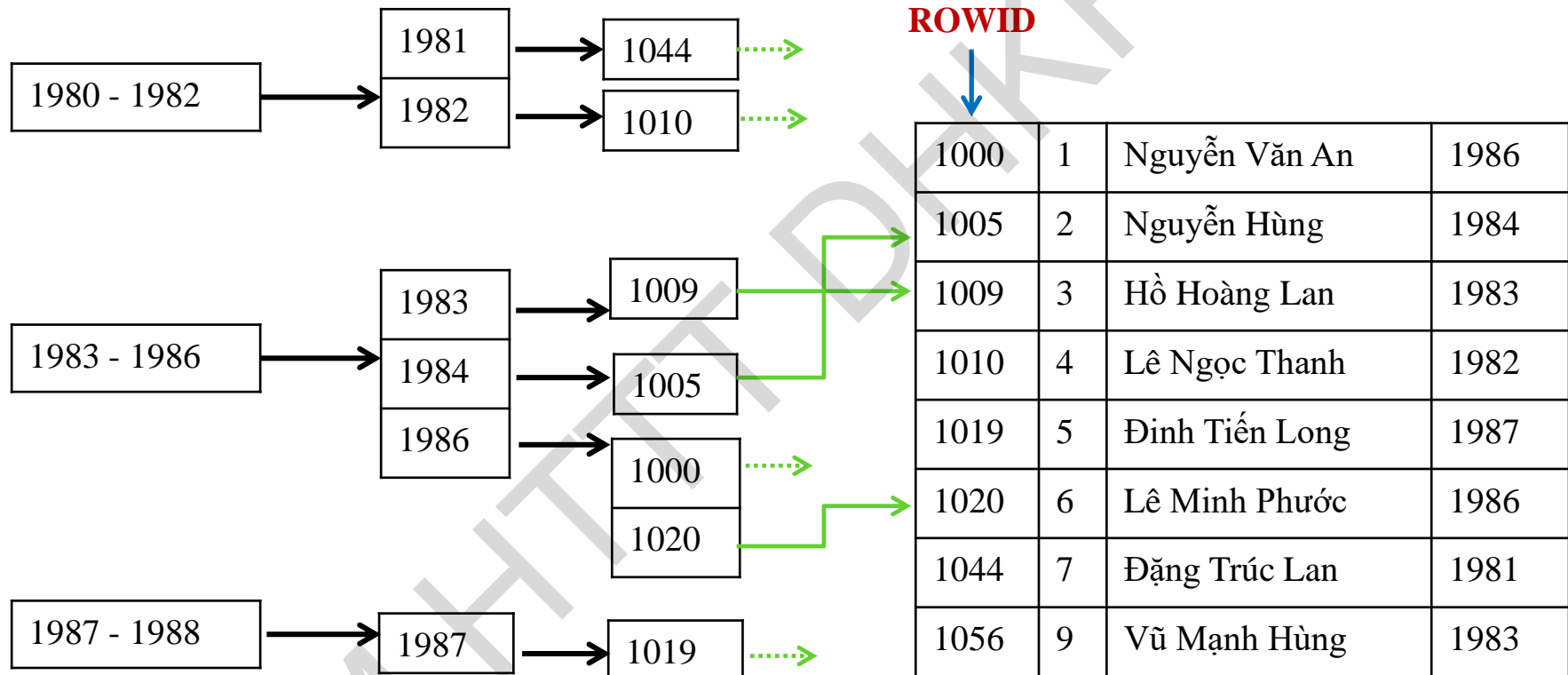
# CLUSTERED INDEX



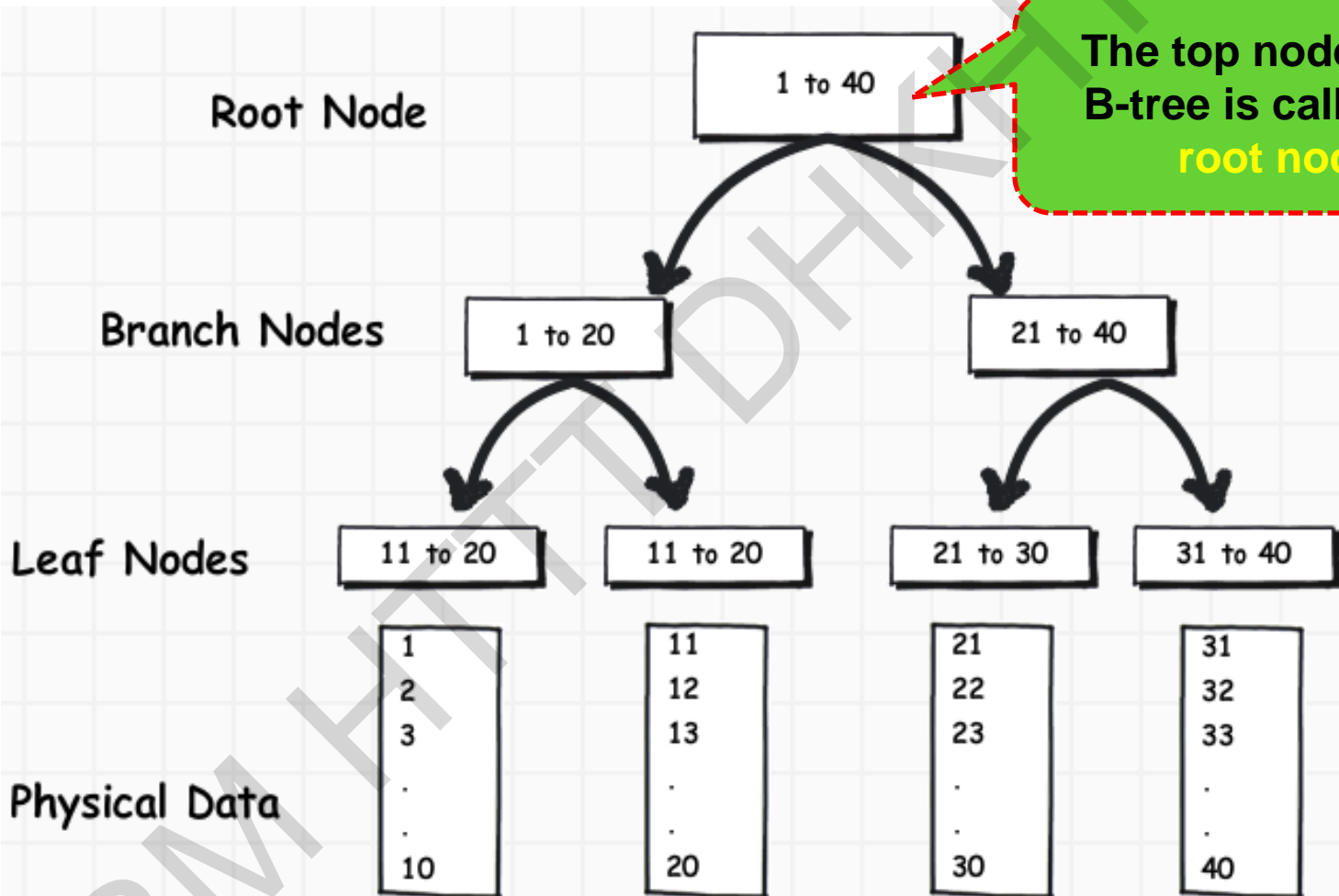
# NON - CLUSTERED INDEX



# NON - CLUSTERED INDEX

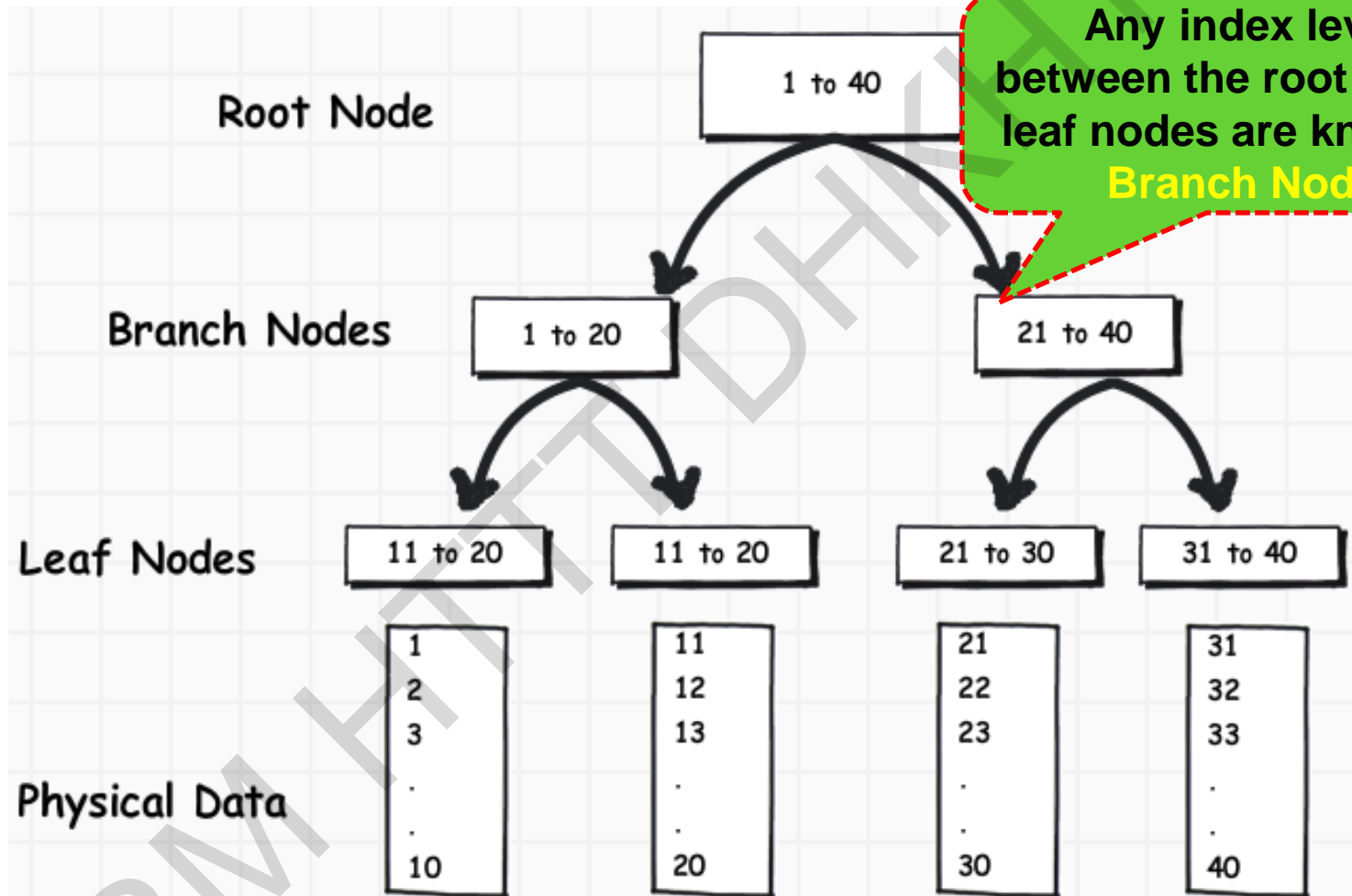


# GIỚI THIỆU VỀ B-TREE INDEX



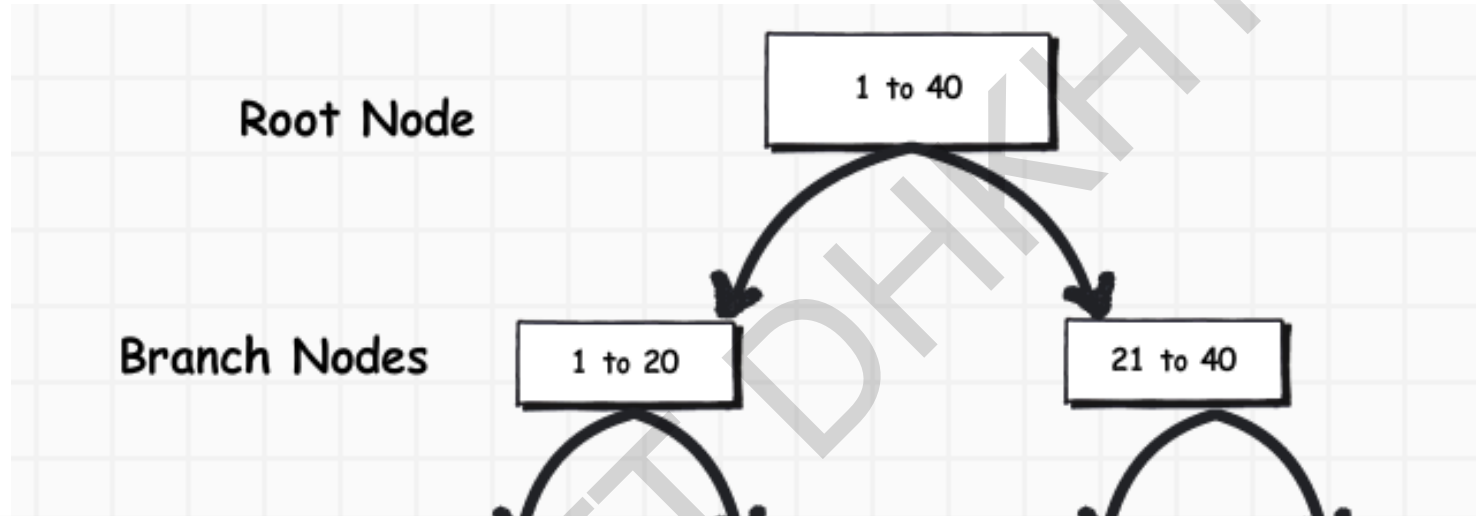
[http://msdn.microsoft.com/en-us/library/ms177443\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms177443(v=sql.105).aspx)

# GIỚI THIỆU VỀ B-TREE INDEX

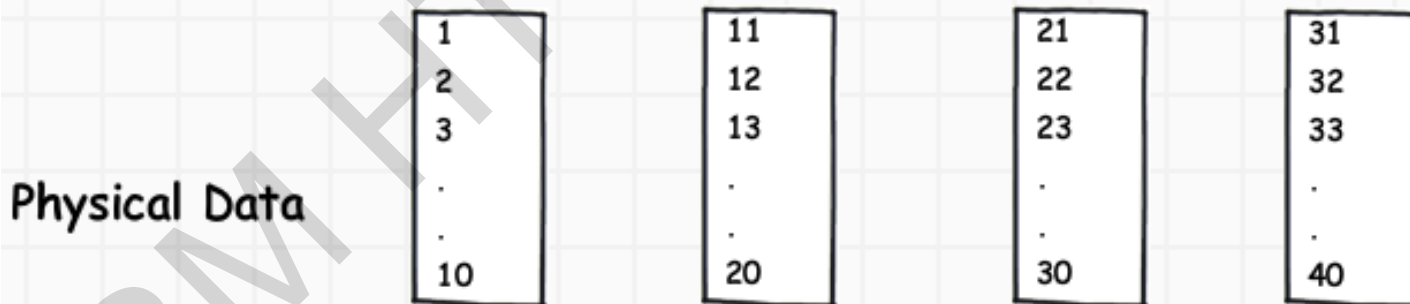




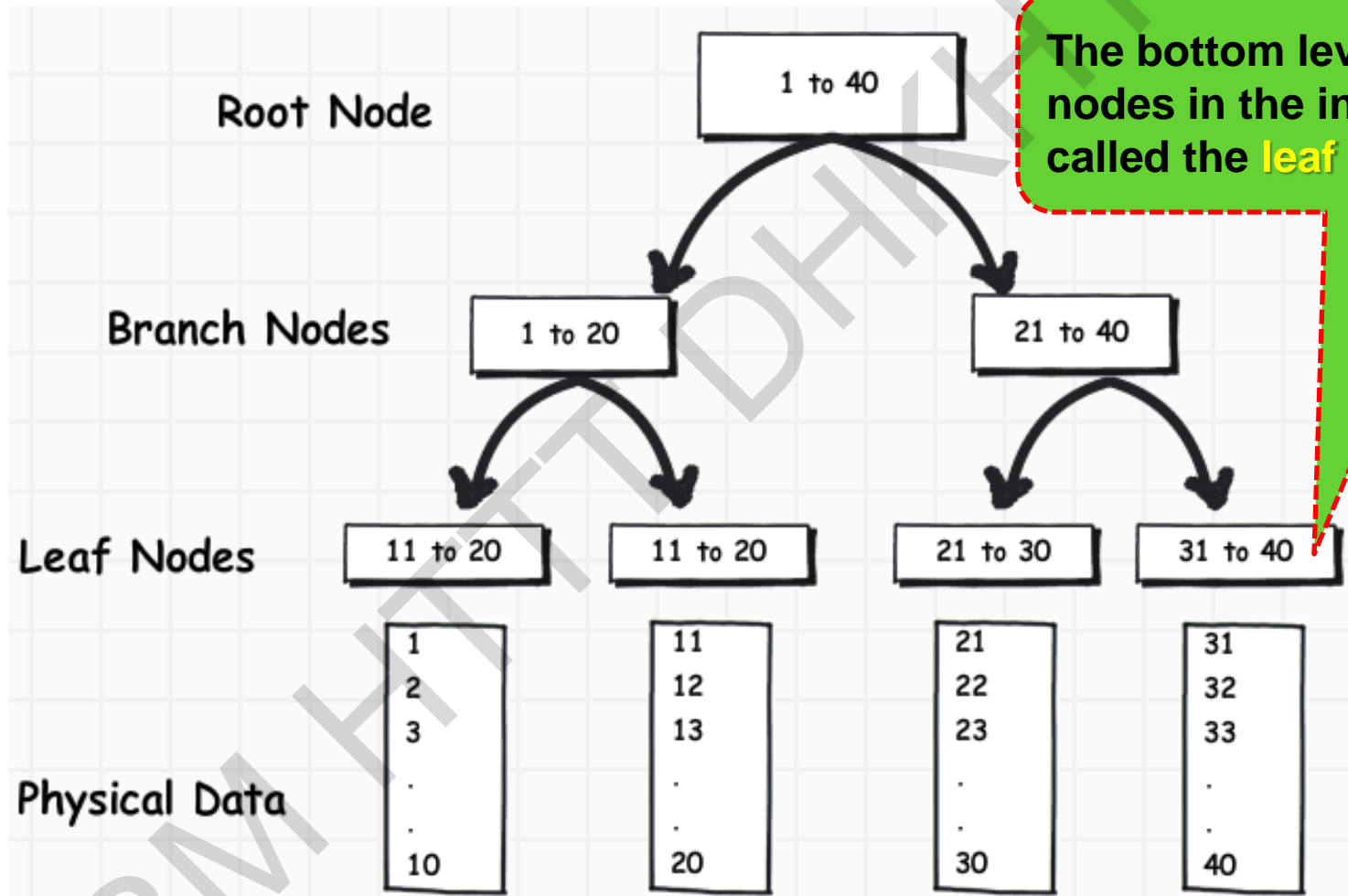
# GIỚI THIỆU VỀ B-TREE INDEX



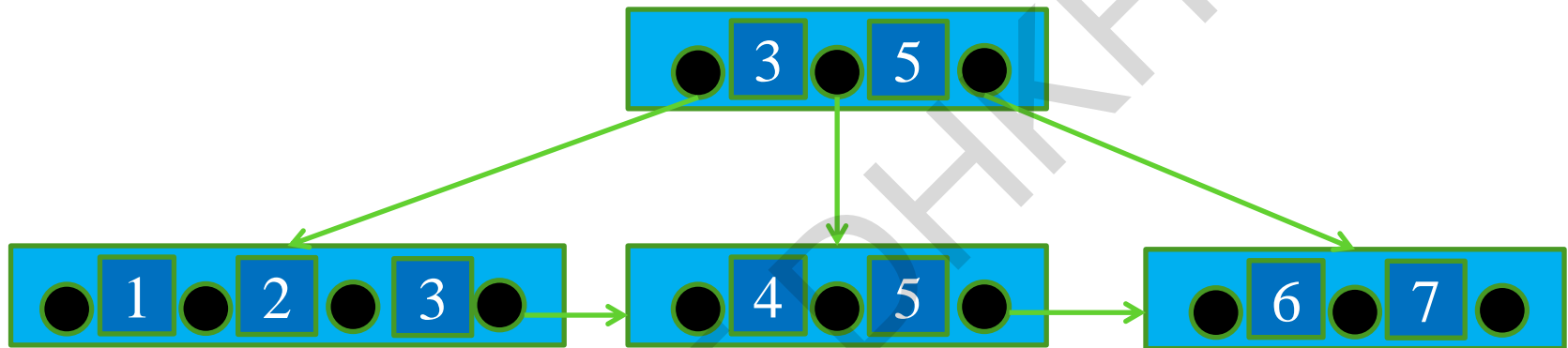
**Each page in an index B-tree is called an index node**  
**The root and Branch level nodes contain index pages holding index rows**



# GIỚI THIỆU VỀ B-TREE INDEX



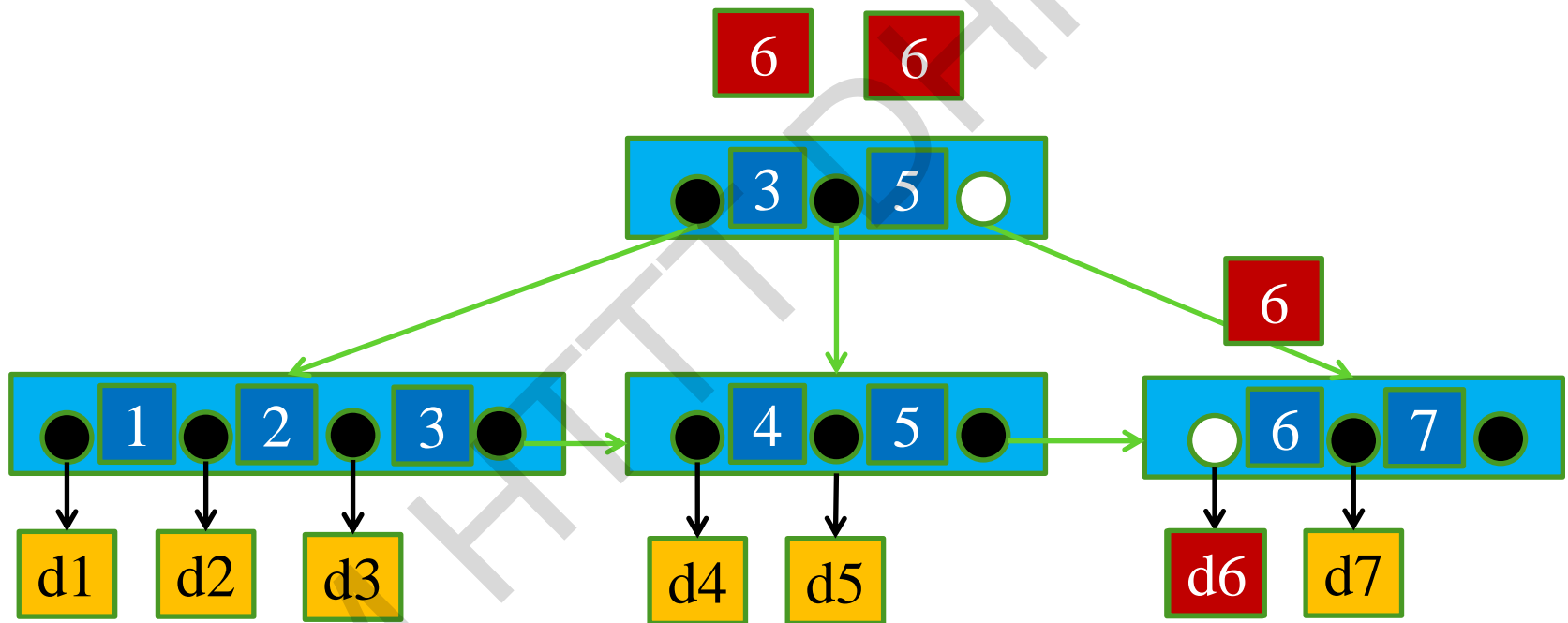
# GIỚI THIỆU VỀ B-TREE INDEX



- ☐ Dựa trên nguyên lý cây đa phân cân bằng
- ☐ Sử dụng phổ biến trong việc tìm kiếm trên dữ liệu lớn ít trùng lặp
- ☐ Được hỗ trợ bởi Oracle, MySQL, SQLServer, ...
- ☐ Khóa chính một bảng sẽ được cài B-Tree index

# NGUYÊN LÝ TÌM KIẾM

Tìm kiếm dòng dữ liệu có khóa là 6



# NHU CẦU

❑ Ví dụ: tìm các sinh viên nam lớp TH02 sinh năm 1984

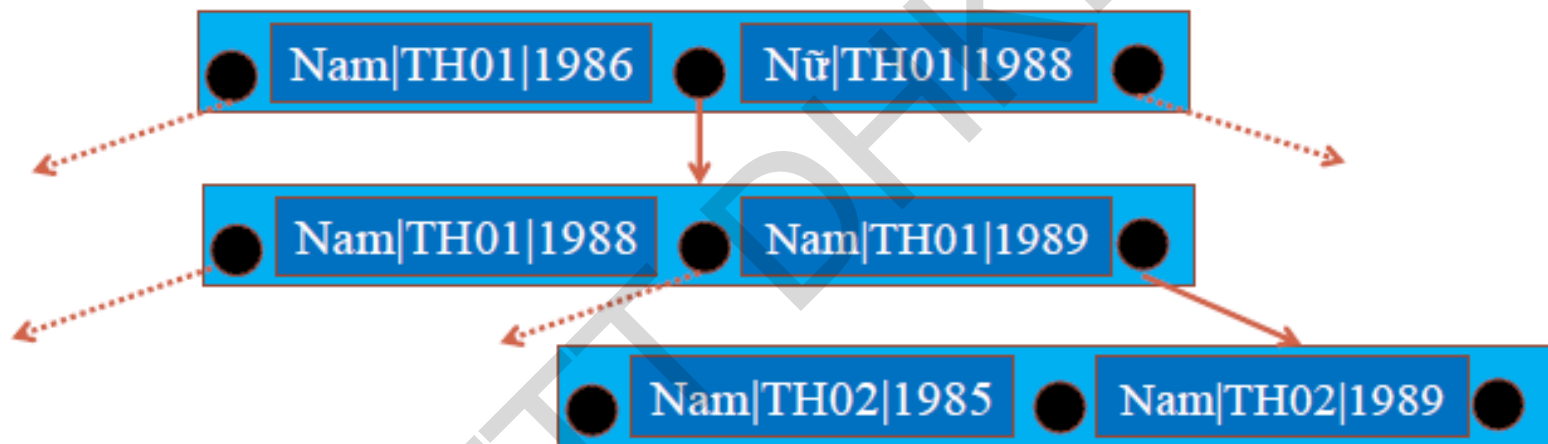
❑ *Giải pháp B-Tree Index*

- Duyệt lần lượt 3 B-Tree Index trên 3 cột thuộc tính “phái”, “lớp”, “năm sinh”
- Giao 3 kết quả để tìm dòng dữ liệu thỏa mãn 3 điều kiện

☹ *Nhiều bộ trung gian, tốn chi phí duyệt B-Tree 3 lần*

*➔ không hiệu quả*

# GIỚI THIỆU COMPOSITE INDEX



- ❑ Một dạng B-Tree cho phép tìm kiếm với nhiều tiêu chí
  - ❑ Khóa của index là tập hợp các thuộc tính được móc nối với nhau
- ➔ *giúp trả nhanh kết quả tìm kiếm cho những câu truy vấn có nhiều điều kiện*



# Chỉ mục trong SQL SERVER

- ❖ **Clustered index**
- ❖ **Non – clustered index**

**It can be configured in other ways**

- ❖ **Composite index**
- ❖ **Unique index**
  - is automatically created when you define a primary key or unique constraint:
    - **Primary key (cluster index)**
    - **Unique (non cluster index)**
- ❖ **Covering index:** chỉ mục trên tất cả các cột cần cho xử lý một câu truy vấn cụ thể

### 3. Composite index có ích khi nào ?

- ❖ Query Optimizer có thể sẽ sử dụng composite index nếu trong truy vấn có thuộc tính nằm trong composite index.

- **Ví dụ : composite index ( A, B, C )**

➔ Truy vấn ( A=1, B=2, C=3 ), ( B=2, C=3 ), ( B=2 ), ( C=3 ) đều có thể sử dụng composite index trên bất kể tăng hay giảm hiệu suất.

- ❖ Một composite index thông thường chỉ hữu ích cho một truy vấn nếu mệnh đề WHERE của truy vấn phù hợp với một hoặc nhiều thuộc tính tận cùng bên trái trong index.

- **Ví dụ: composite index ( A, B, C ).**

➔ truy vấn có lợi : ( A=1 ), ( A=1, B=2 ), ( A=1, B=2, C=3 ), ( A=1, B=2, C=3, D=4 ),

không có lợi: ( A=1, C=3 ), ( B=2, C=3 ), ( B=2 ), ...

# Tạo Composite index

❖ Là index gồm từ 2 thuộc tính trở lên.

❖ **Cú pháp:**

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX [idex1]  
ON [ Table ] (key1, key2, key3);
```

❖ Giúp cho việc tìm kiếm dữ liệu nhanh hơn nhờ sắp xếp dữ liệu theo kiểu B-tree.

# Unique & Covering index

- **Composite index** (chỉ mục phức hợp): chứa 1 hoặc nhiều cột
  - SQL 2005, 2008 có thể bao gồm 16 columns cho 1 index ( chỉ mục không vượt quá 900 byte)
  - Cả cluster/ noncluster index có thể là composite index
- **Unique index**: đảm bảo mỗi giá trị trong cột được chỉ mục là duy nhất
- **Covering index**: là 1 loại chỉ mục bao gồm tất cả các cột cần thiết cho xử lý 1 câu truy vấn cụ thể.
  - Ví dụ: câu truy vấn cần rút trích cột Firstname, lastname từ 1 bảng dựa vào giá trị cột ContactID. Có thể tạo chỉ mục phủ bao gồm 3 cột trên

# Nội dung

1. Định nghĩa
2. Phân loại chỉ mục
3. Execution plan
4. Cài đặt chỉ mục với SQLserver

# Execution plan

- ❖ Làm thế nào để đánh giá lợi ích của việc lựa chọn thiết kế ?



ORACLE®

Microsoft®  
SQL Server®



# Execution plan

The graphical plans in SQL Server Management Studio:

## ❖ **Display Estimated Execution Plan**

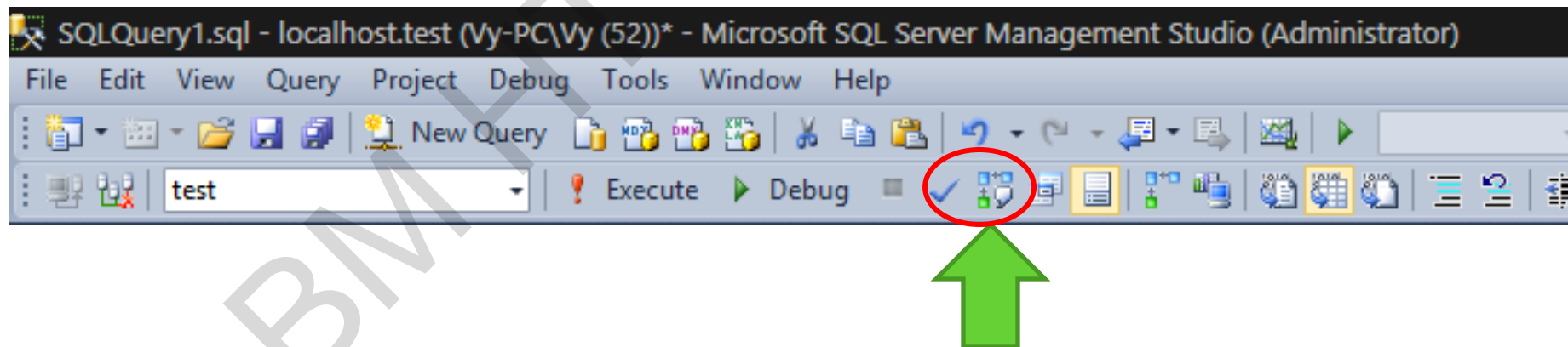
- show the plan immediately without executing the query

## ❖ **Include Actual Execution Plan**

- both click on **Include Actual Execution Plan** and then execute the query.

# Query Execution Plan

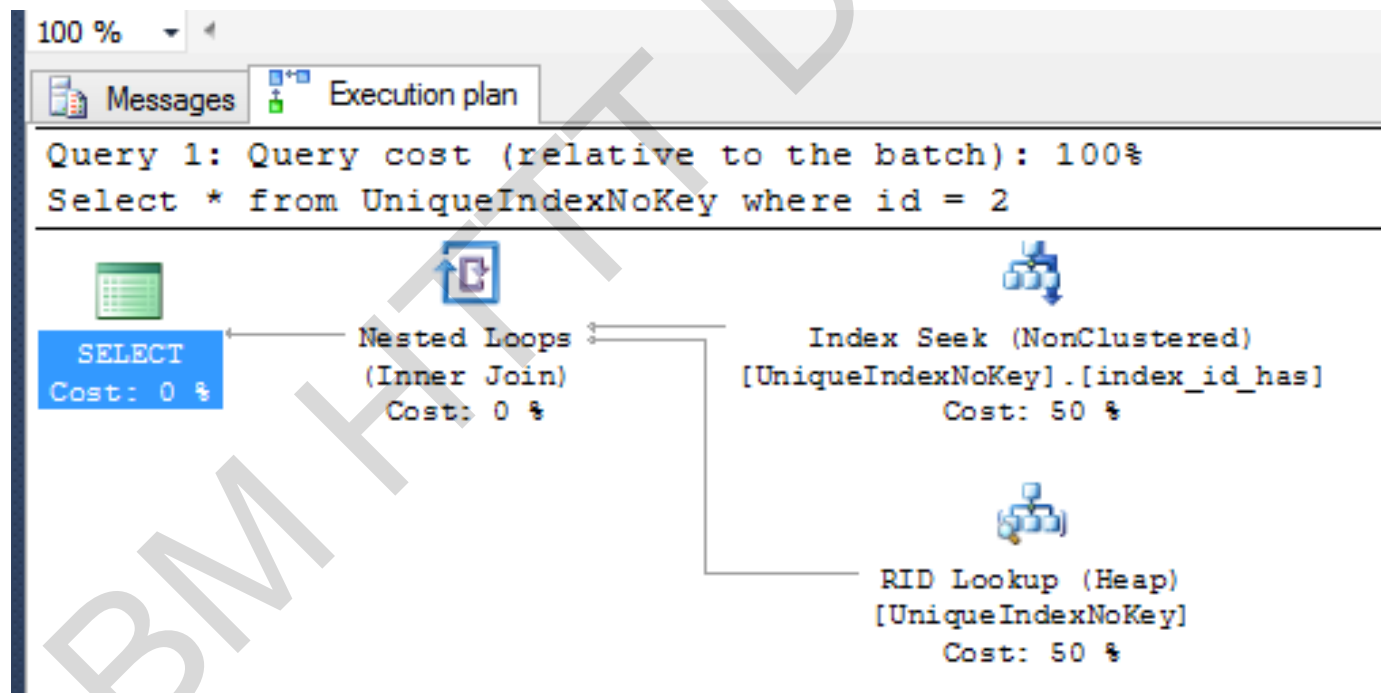
- ❖ TG thực thi truy vấn = TG tạo Execution Plan + TG thực thi truy vấn dựa trên execution plan đó
- ❖ Dựa vào Execution Plan có thể biết được một câu truy vấn có sử dụng index hay không, và mức hiệu của việc sử dụng index đó.



# Query Execution Plan

## ❖ Ví dụ

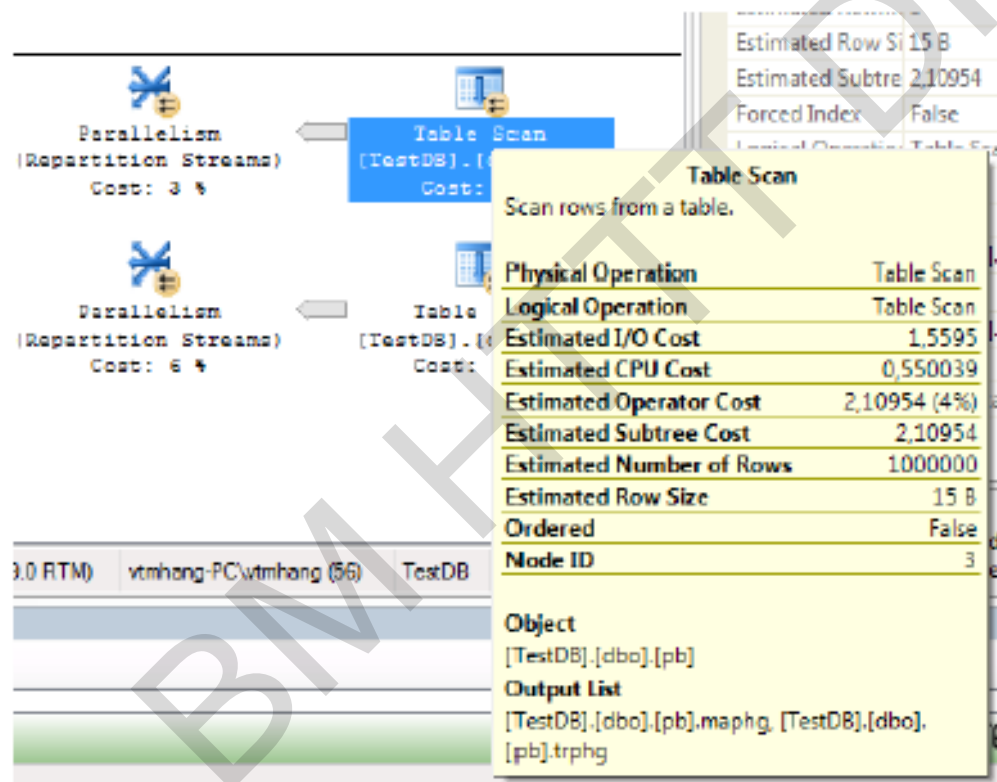
Select \* from UniqueIndexNoKey where id = 2



# Xem execution plan

**SELECT \* FROM nv, pb WHERE nv.manv = pb.trphg**

Chọn *Display  
Estimated  
Execution Plan*



# Ví dụ

```
-- #1
SELECT CustomerID, CustomerType
FROM Sales.Customer_NoIndex
WHERE CustomerID = 11001

-- #2
SELECT CustomerID, CustomerType
FROM Sales.Customer_Index
WHERE CustomerID = 11001
```

1

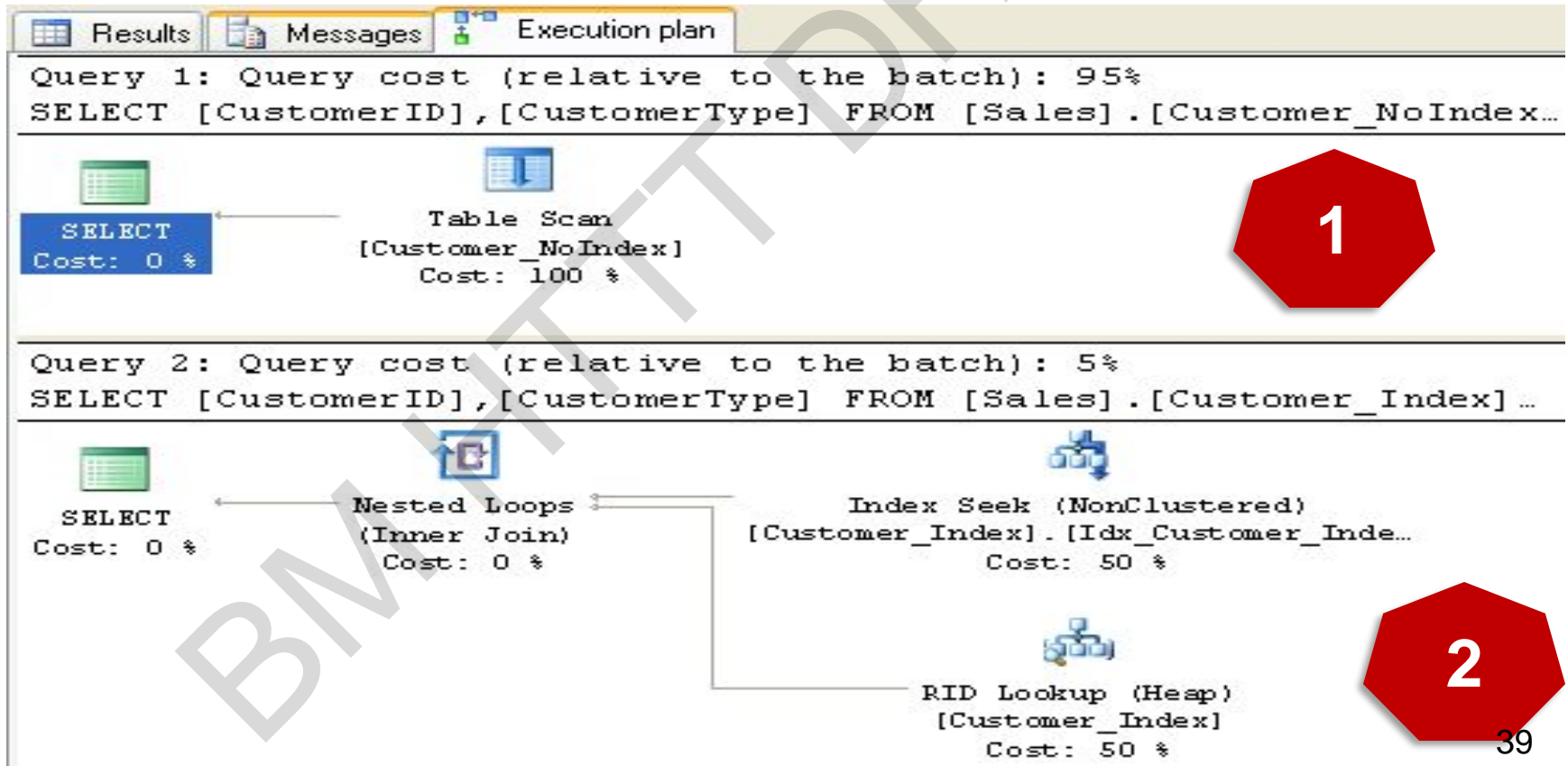
2

- ❖ 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”**.

# Query Execution Plan

❖ Được thể hiện như 1 biểu đồ hướng dẫn:

- Đối tượng: table, index, materialized views, catalogs.
- Thao tác: join (hash join, nested-loop, semijoin), sort..





# Query Execution Plan

- ❖ Table scan vs Index scan vs index seek
  - An **index scan** or **table scan** is when SQL Server has to scan the data or index pages to find the appropriate records.
  - Index seek uses the index to pinpoint the records that are needed to satisfy the query.

# Query Execution Plan

❖ Table có cluster index → index scan

The screenshot displays the SQL Server Enterprise Manager interface. At the top, there are two tabs: 'index types.sql - H...SPECTRE\hthvy (53))\*' and 'Filter index.sql - H...eWorks2012 ('. The main window shows a SQL query with line numbers 44 to 49. The query is as follows:

```
44 --(OrderDate ASC) WHERE ShipDate IS NULL;  
45 --CREATE NONCLUSTERED INDEX IX_SalesOrder  
46 --    ON Sales.SalesOrderHeader (ShipDate)  
47  
48 --index scan/seek  
49 SELECT * FROM Person.Person
```

Below the query editor, there is a 'Messages' and 'Execution plan' tab. The 'Execution plan' tab is active, showing the execution plan for 'Query 1: Query cost (relative to the batch): 100%'. The query text is 'SELECT \* FROM Person.Person'. The execution plan shows a single operator: 'Clustered Index Scan (C... [Person].[PK\_Person\_Bus... Cost: 100 %'. A blue box highlights the 'SELECT' operator with a cost of 0 %.

# Query Execution Plan

❖ Table không có cluster index → table scan

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query script with the following content:

```
45 --CREATE NONCLUSTERED INDEX IX_SalesOrderH
46 --    ON Sales.SalesOrderHeader (ShipDate)
47
48 --index scan/seek
49 SELECT * FROM Person.Person
50 SELECT * FROM Person.Person_NoKey
51
```

The bottom pane shows the execution plan for the query. The plan consists of a single operator: a Table Scan for the table [Person\_NoKey]. The cost of this operator is 100%.

Query 1: Query cost (relative to the batch): 100%

SELECT \* FROM Person.Person\_NoKey

Table Scan  
[Person\_NoKey]  
Cost: 100 %

# Query Execution Plan

- ❖ Vẫn là Clustered Index Scan → nhưng SQL server cảnh báo thiếu index

```
SELECT * FROM Person.Contact  
WHERE LastName = 'Russell'
```

48 --Index scan/seek  
49 SELECT \* FROM Person.Person  
50 where LastName = 'Russell'  
51 SELECT \* FROM Person.Person\_NoKey

100 %

Messages Execution plan

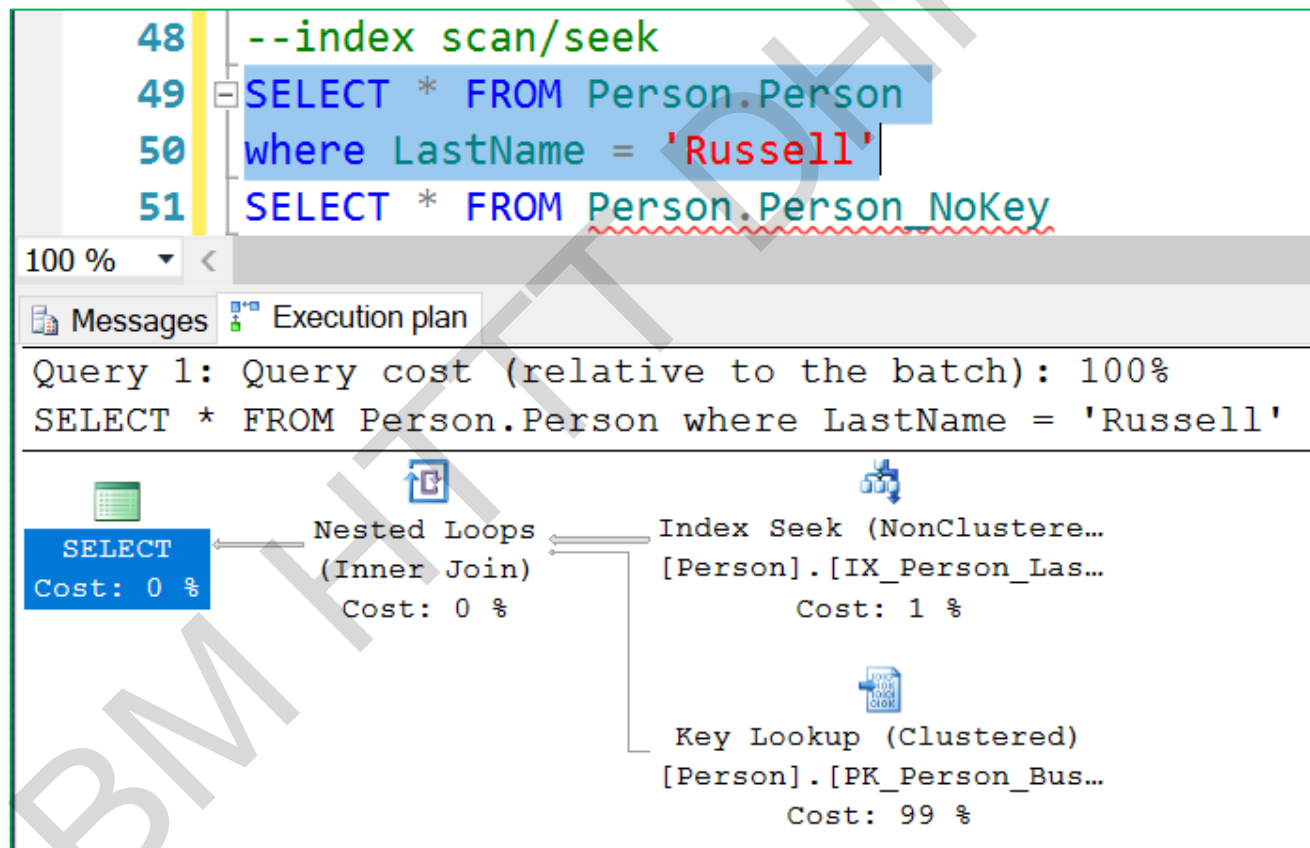
Query 1: Query cost (relative to the batch): 100%  
SELECT \* FROM Person.Person where LastName = 'Russell'  
Missing Index (Impact 99.0864): CREATE NONCLUSTERED INDEX [<Name of Missing Index, >]

Clustered Index Scan (C...  
[Person].[PK\_Person\_Bus...  
Cost: 100 %

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]  
ON [Person].[Person] ([LastName])

# Query Execution Plan

❖ Sau khi tạo chỉ mục phù hợp → index seek



# Nội dung

1. Định nghĩa
2. Phân loại chỉ mục
3. Execution plan
4. Cài đặt chỉ mục với SQLserver

# TẠO INDEX VỚI SQLSERVER

## ❖ Cluster index

- Clustered indexes sort and store the data rows in the table or view based on their key values

## ❖ Non-cluster index

- have a structure separate from the data rows
- contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.
- **Unique index**
  - when you create a table with a UNIQUE constraint, Database Engine automatically creates a nonclustered index
- **Covering index**
- **Index with Include column**
- **Filter index**



# TẠO INDEX VỚI SQLSERVER

- ❖ Khi tạo bảng có khoá chính, chỉ mục cluster index được tự tạo

```
USE [test]
GO

/***** Object: Table [dbo].[PrimaryKey]      Script Date: 8/10/2013 12:15:41 PM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PrimaryKey](
    [id] [int] NOT NULL,
    [name] [nvarchar](30) NULL,
    [addr] [nvarchar](30) NULL,
    PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALL
    ) ON [PRIMARY]
GO
```

# TẠO INDEX VỚI SQLSERVER

❖ Liệt kê danh sách chỉ mục có trong csdl

```
USE test
GO
```

```
SELECT
```

```
    so.name AS TableName
  , si.name AS IndexName
  , si.type_desc AS IndexType
```

```
FROM
```

```
    sys.indexes si
  JOIN sys.objects so ON si.[object_id] = so.[object_id]
```

```
WHERE
```

```
    so.type = 'U'      --Only get indexes for User Created Tables
  AND si.name IS NOT NULL
```

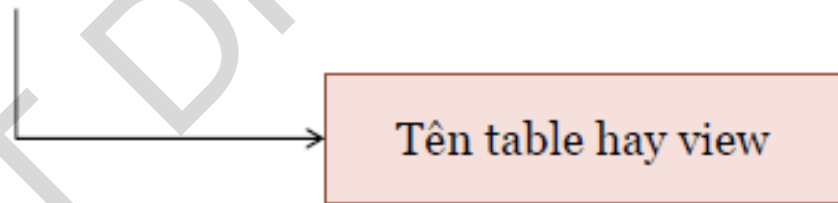
```
ORDER BY
```

```
    so.name, si.type |
```

# TẠO INDEX VỚI SQLSERVER

## ❖ Cú pháp

```
CREATE [UNIQUE | CLUSTERED | NONCLUSTERED]  
INDEX ind_name ON <OBJECT> (COL1, COL2, ...)
```



**Có thể tạo chỉ mục trên hầu hết các cột, ngoại trừ những cột có kiểu dữ liệu lớn (large object): image, text và varchar(max)**

# TẠO INDEX VỚI SQLSERVER

❖ Tạo chỉ mục trên 1 bảng. Cho phép trùng giá trị:

- **CREATE INDEX** index\_name  
**ON** table\_name (column\_name)

❖ Tạo chỉ mục unique, không cho phép trùng giá trị:

- **CREATE UNIQUE INDEX** index\_name  
**ON** table\_name (column\_name)

# TẠO INDEX VỚI SQLSERVER

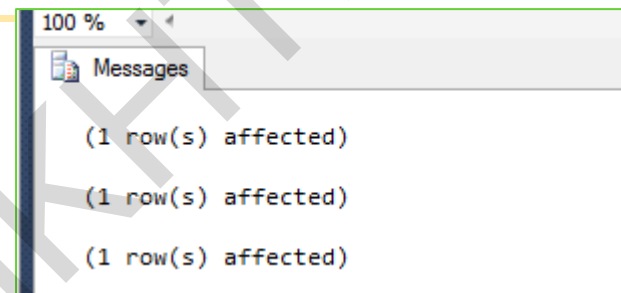
```
create table NoPrimaryKey  
(  
    id int,  
    name nvarchar(30),  
    addr nvarchar(30)  
)
```

--Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_id  
ON NoPrimaryKey (id)
```

```
insert into NoPrimaryKey values(1,N'Thiên Ân', N'Điện Biên Phủ')  
insert into NoPrimaryKey values(1,N'Tường Vi', N'Phan Văn Hân')  
insert into NoPrimaryKey values(2,N'Trúc Ly', N'Điện Biên Phủ')
```

```
Select * from NoPrimaryKey
```



# TẠO INDEX VỚI SQLSERVER

--Tạo chỉ mục unique, không cho phép trùng giá trị

```
create table UniqueIndexNoKey
```

```
(  
    id int,  
    name nvarchar(30),  
    addr nvarchar(30)|  
)
```

```
CREATE UNIQUE INDEX index_id_has ON UniqueIndexNoKey (id)
```

```
insert into UniqueIndexNoKey values(1,N'Thiên Ân', N'Điện Biên Phủ')
```

```
insert into UniqueIndexNoKey values(1,N'Tường Vi', N'Phan Văn Hân')
```

```
insert into UniqueIndexNoKey values(2,N'Trúc Ly', N'Điện Biên Phủ')
```

Messages

(1 row(s) affected)

Msg 2601, Level 14, State 1, Line 2

Cannot insert duplicate key row in object 'dbo.HasPrimaryKey' with unique index 'index\_id\_has'. The duplicate key value is (1).  
The statement has been terminated.

(1 row(s) affected)

# TẠO INDEX VỚI SQLSERVER

## ❖ Lưu ý:

- SQL server không cho phép tạo unique index trên một cột mà đã tồn tại giá trị trùng nhau
- Duplicate value phải được xoá bỏ
- Và cột được tạo chỉ mục unique phải là NOT NULL

100 %

Messages

Msg 1505, Level 16, State 1, Line 1  
The CREATE UNIQUE INDEX statement terminated because a duplicate key was found for the object name 'dbo.UniqueIndexNoKey' and the  
The statement has been terminated.



# TẠO INDEX VỚI SQLSERVER

## ❖ **NON-CLUSTER** index

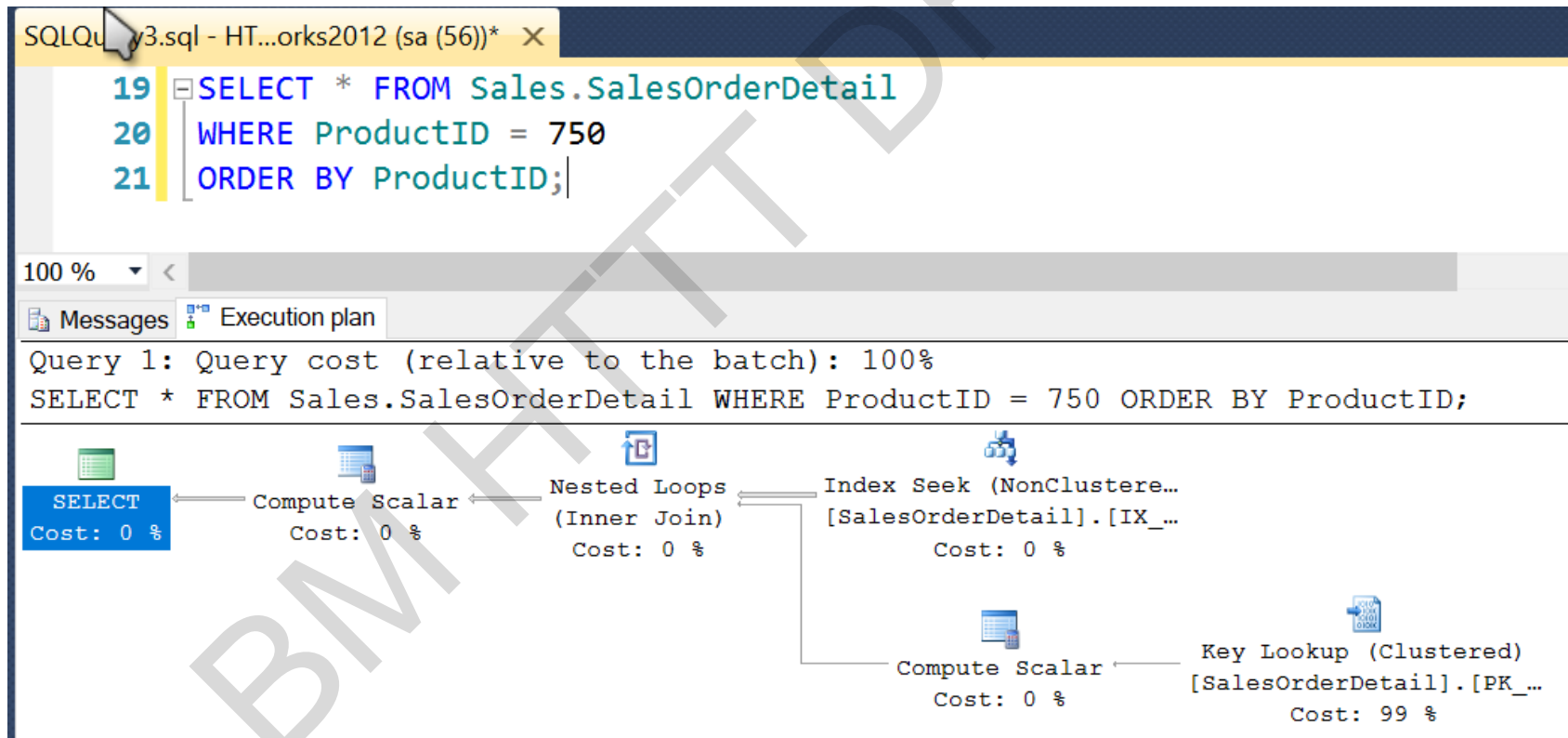
-- Adding non-clustered index

```
CREATE NONCLUSTERED INDEX Person_LastFirstName  
ON Person.Person(LastName ASC, FirstName ASC);
```

```
CREATE INDEX Person_FirstName  
ON Person.Person (FirstName ASC);
```

# TẠO INDEX VỚI SQLSERVER

- ❖ Không cần table scan để lấy data → index seek của non-cluster index và lookup cluster index để lấy dữ liệu, không cần sort vì dữ liệu đã theo thứ tự



# TẠO INDEX VỚI SQLSERVER

```
SELECT ProductNumber, Name  
FROM Production.Product  
WHERE Name = 'Cable Lock';
```

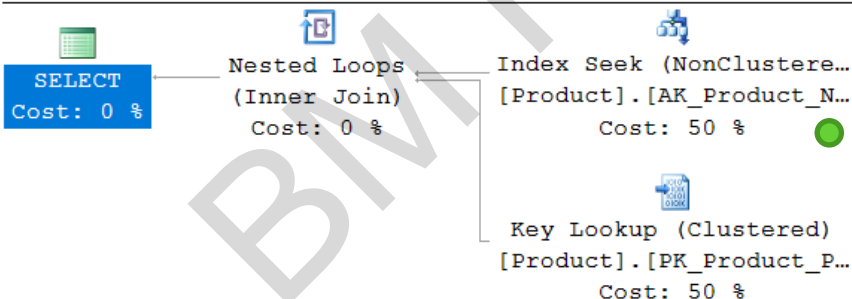
Đã tạo 2 index  
pro\_ID và  
pro\_name

100 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
SELECT ProductNumber, Name FROM Production.Product WHERE Name = 'Cable Lock';
```



# TẠO INDEX VỚI SQLSERVER

## ❖ Chỉ mục phủ (covering index)

```
CREATE NONCLUSTERED INDEX P_PNumber_Name  
ON Production.Product (Name ASC, ProductNumber ASC);
```

```
SELECT ProductNumber, Name  
FROM Production.Product  
WHERE Name = 'Cable Lock';
```

# TẠO INDEX VỚI SQLSERVER

- ❖ Sau khi tạo 1 chỉ mục phủ → Câu truy vấn được thực hiện dựa trên truy cập chỉ mục vừa tạo → không cần thực hiện lookup cho việc tìm kiếm

SQLQuery3.sql - HT...orks2012 (sa (52))\* X

```
24  
25 SELECT ProductNumber, Name  
26 FROM Production.Product  
27 WHERE Name = 'Cable Lock';  
28 --CREATE NONCLUSTERED INDEX IX_Production_ProductNumber_Name  
29 --    ON Production.Product (Name ASC,ProductNumber ASC);
```

100 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT ProductNumber, Name FROM Production.Product WHERE Name = 'Cable Lock';

SELECT  
Cost: 0 %

Index Seek (NonClustere...  
[Product].[IX\_Productionio...  
Cost: 100 %

# TẠO INDEX VỚI SQLSERVER

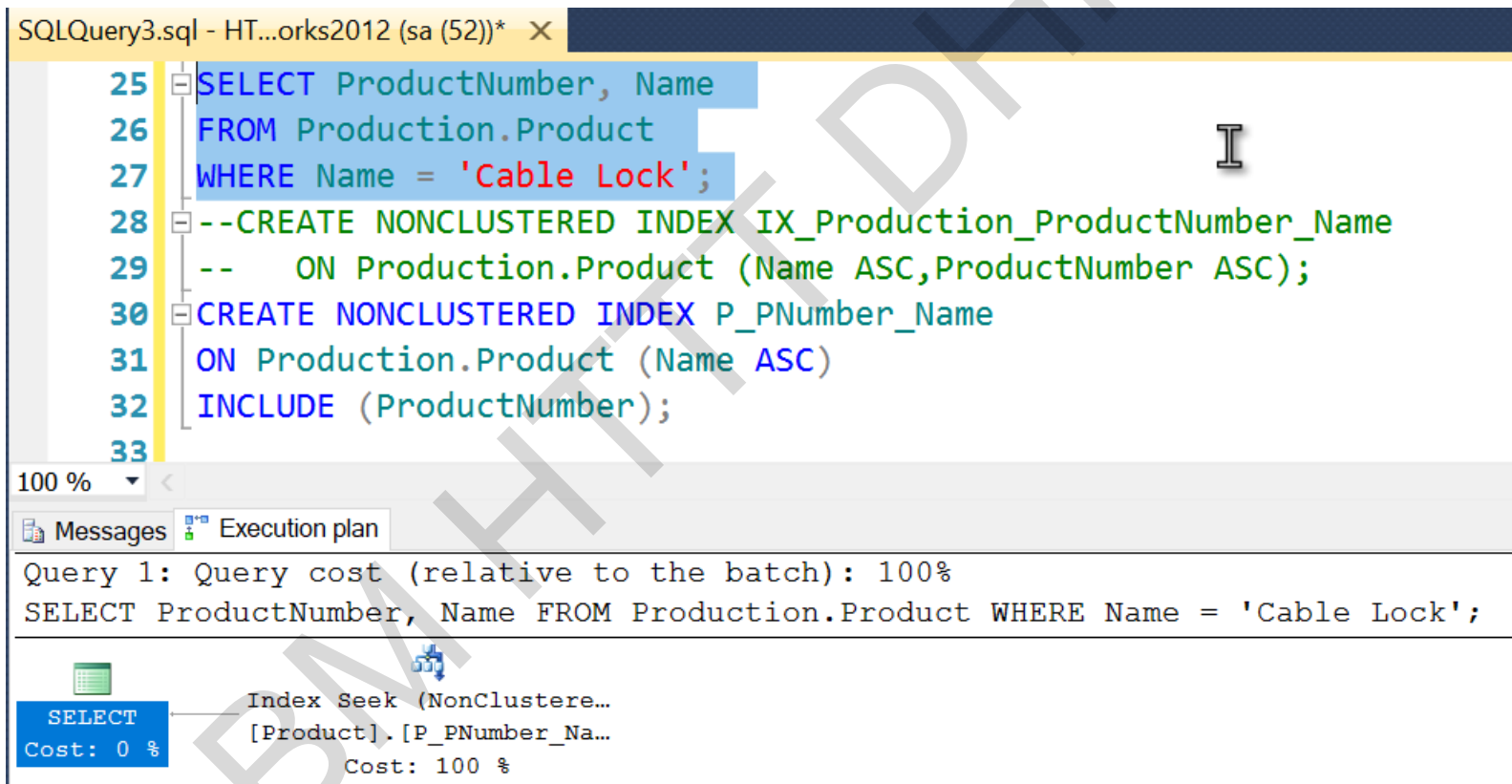
## ❖ Index với include column

```
CREATE NONCLUSTERED INDEX P_PNumber_Name  
ON Production.Product (Name ASC)  
INCLUDE (ProductNumber);
```

```
SELECT ProductNumber, Name  
FROM Production.Product  
WHERE Name = 'Cable Lock';
```

# TẠO INDEX VỚI SQLSERVER

❖ Câu truy vấn được thực hiện mà không cần thao tác lookup



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query in a text editor. The query consists of a SELECT statement followed by two CREATE INDEX statements. The SELECT statement is highlighted in blue. The bottom pane shows the execution plan for the query, which is a single 'Index Seek' operation. The cost of the query is 100%.

```
SQLQuery3.sql - HT...orks2012 (sa (52))* X
25 SELECT ProductNumber, Name
26 FROM Production.Product
27 WHERE Name = 'Cable Lock';
28 --CREATE NONCLUSTERED INDEX IX_Production_ProductNumber_Name
29 --  ON Production.Product (Name ASC,ProductNumber ASC);
30 CREATE NONCLUSTERED INDEX P_PNumber_Name
31 ON Production.Product (Name ASC)
32 INCLUDE (ProductNumber);
33
```

100 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT ProductNumber, Name FROM Production.Product WHERE Name = 'Cable Lock';

Index Seek (NonClustered...  
[Product].[P\_PNumber\_Na...  
Cost: 100 %

SELECT  
Cost: 0 %



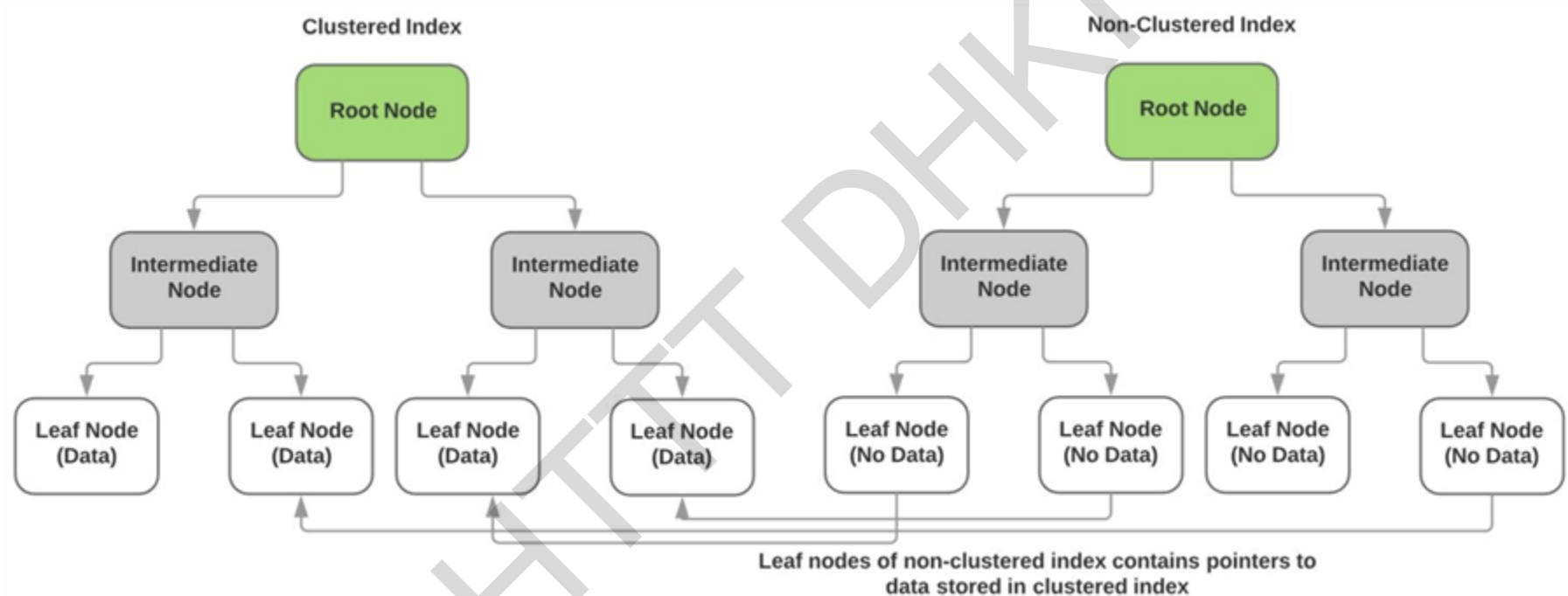
# TẠO INDEX VỚI SQLSERVER

## ❖ Index với Include column:

- Là 1 non-cluster index cũng bao gồm non-key column trong node lá của chỉ mục tương tự như cluster index
- Lợi ích: cho phép bao gồm các cột không được cho phép trong index keys → tất cả các column trong câu query / index key / include column sẽ không cần phải thực hiện lookup để lấy data thoả yêu cầu truy vấn
- → giảm thao tác truy xuất đĩa
- → tương tự như covering index

# TẠO INDEX VỚI SQLSERVER

## ❖ Non-clustered indexes vs clustered index



<https://www.mssqltips.com/sqlservertutorial/9133/sql-server-nonclustered-indexes/>

# TẠO INDEX VỚI SQLSERVER

## ❖ FILTERED INDEX

- A filtered index is a special index type where only a certain portion of the rows of the table are indexed.
  - Based on the filter criteria that is applied when the index is created only the remaining rows are indexed which can save on space, improve on query performance and reduce maintenance overhead as the index is much smaller

➔ *Hữu ích khi tạo index trên các column có nhiều giá trị **NULL** trong các cột nhất định*

# TẠO INDEX VỚI SQLSERVER

## ❖ FILTERED INDEX

### ■ Cú pháp

```
CREATE INDEX index_name ON  
table_name(column_list)  
WHERE predicate;
```

```
CREATE NONCLUSTERED INDEX OrderDate_ShipDate  
ON Sales.SalesOrderHeader (OrderDate ASC)  
WHERE ShipDate IS NULL;
```

If the rows returned from the query are beyond the filtered index criteria, the optimizer will not use the filtered index

# Ví dụ filter index

```
51
52 select CustomerID, PersonID, AccountNumber
53 FROM [Sales].[Customer]_NC
54 where PersonID='16215'
```

133 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

select CustomerID, PersonID, AccountNumber FROM [Sales].[Customer]\_NC where PersonID='16215'

Missing Index (Impact 96.9669): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname>] ON [Sales].[Customer]\_NC (PersonID)

```
graph LR
    SELECT[SELECT  
Cost: 0 %] --> Filter[Filter  
Cost: 7 %]
    Filter --> CS1[Compute Scalar  
Cost: 2 %]
    CS1 --> CS2[Compute Scalar  
Cost: 2 %]
    CS2 --> CIS[Clustered Index Scan (C...  
[Customer].[PK_Customer...]  
Cost: 89 %]
```

```
49 create nonclustered index ix_phone
50 ON [Sales].[Customer_NC](customerID)
51 Where customerID is not null
52
53 select CustomerID, PersonID, AccountNumber
54 FROM [Sales].[Customer_NC]
55 where customerID='16215'
```

133 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

select CustomerID, PersonID, AccountNumber FROM [Sales].[Customer\_NC] where customerID='16215'

```
graph LR
    SELECT[SELECT  
Cost: 0 %] --> NL[Nested Loops  
(Inner Join)  
Cost: 0 %]
    NL --> IS[Index Seek (NonClustered...  
[Customer_NC].[Idx_Cust...]  
Cost: 50 %]
    NL --> RL[RID Lookup (Heap)  
[Customer_NC]  
Cost: 50 %]
```

# TẠO INDEX VỚI SQLSERVER

## ❖ Trước khi tạo filter index

SQLQuery4.sql - HT...orks2012 (sa (53))    SQLQuery3.sql - HT...orks2012 (sa (52))\* X

```
33
34 --filter index
35 SELECT OrderDate FROM Sales.SalesOrderHeader
36 WHERE ShipDate IS NULL
37 ORDER BY OrderDate ASC;
```

100 %

Messages    Execution plan

Query 1: Query cost (relative to the batch): 100%  
SELECT OrderDate FROM Sales.SalesOrderHeader WHERE ShipDate IS NULL  
Missing Index (Impact 97.3334): CREATE NONCLUSTERED INDEX

SELECT  
Cost: 0 %

Sort  
Cost: 5 %

Clustered Index Scan (C...  
[SalesOrderHeader].[PK\_...]  
Cost: 95 %

Null rows: 0  
Not null rows: 31.465



# TẠO INDEX VỚI SQLSERVER

## ❖ Sau khi tạo filter index

SQLQuery3.sql - HT...PECTRE\hthvy (53))\* X Filter index.sql - H...eWorks2012 (sa (52))

```
34 --filter index
35 SELECT OrderDate FROM Sales.SalesOrderHeader
36 WHERE ShipDate IS NULL
37 ORDER BY OrderDate ASC;
38
39 --CREATE NONCLUSTERED INDEX IX_SalesOrderHeader
40 --(OrderDate ASC) WHERE ShipDate IS NULL;
41
```

100 % <

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

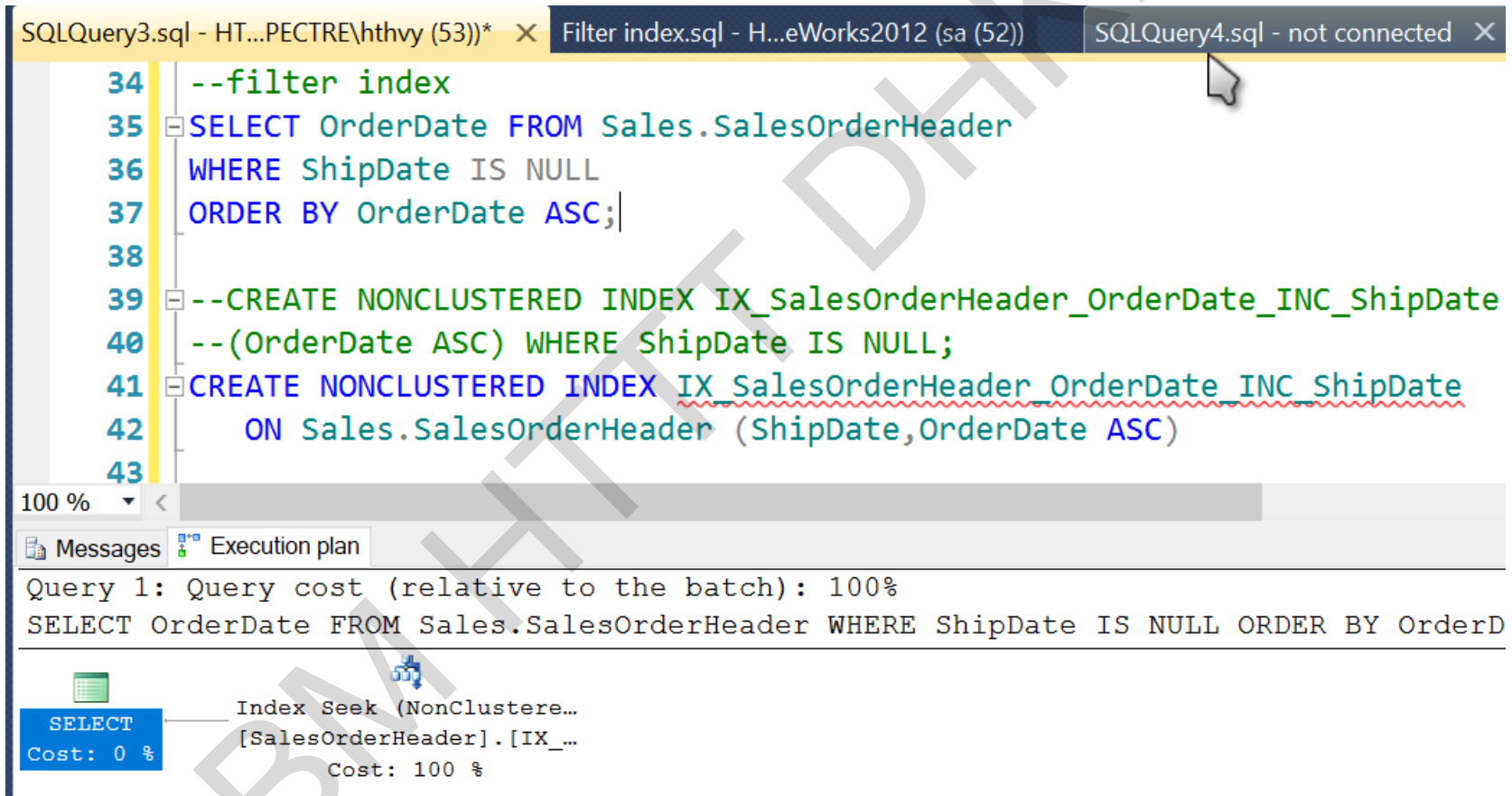
SELECT OrderDate FROM Sales.SalesOrderHeader WHERE ShipDate

```
graph LR
    SELECT[SELECT  
Cost: 0 %] --> NL[Nested Loops  
(Inner Join)  
Cost: 0 %]
    NL --> IS[Index Scan (NonClustered)  
[SalesOrderHeader].[IX_...]  
Cost: 25 %]
    IS --> KL[Key Lookup (Clustered)  
[SalesOrderHeader].[PK_...]  
Cost: 75 %]
```



# TẠO INDEX VỚI SQLSERVER

❖ Chỉ mục phủ tận dụng được index seek



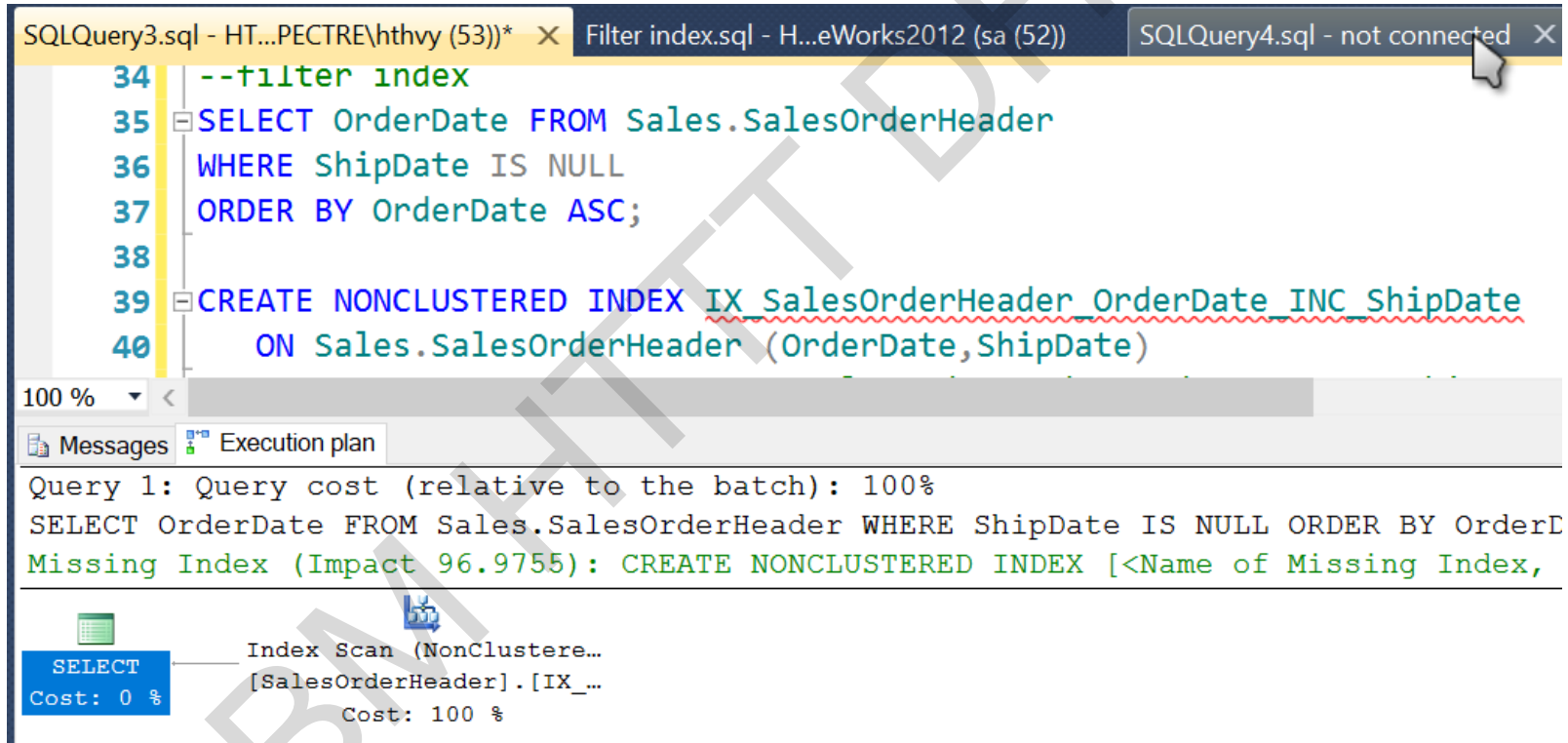
The screenshot displays the SQL Server Enterprise Manager interface. At the top, there are three tabs: 'SQLQuery3.sql - HT...PECTRE\hthvy (53)\*', 'Filter index.sql - H...eWorks2012 (sa (52))', and 'SQLQuery4.sql - not connected'. The active tab shows the following SQL code:

```
34 --filter index
35 SELECT OrderDate FROM Sales.SalesOrderHeader
36 WHERE ShipDate IS NULL
37 ORDER BY OrderDate ASC;
38
39 --CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_OrderDate_INC_ShipDate
40 --(OrderDate ASC) WHERE ShipDate IS NULL;
41 CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_OrderDate_INC_ShipDate
42 ON Sales.SalesOrderHeader (ShipDate, OrderDate ASC)
43
```

Below the code editor, the 'Execution plan' tab is selected. It shows the execution plan for 'Query 1: Query cost (relative to the batch): 100%'. The plan consists of a single step: 'Index Seek (NonClustered... [SalesOrderHeader].[IX\_... Cost: 100 %'. A tooltip for this step shows 'SELECT Cost: 0 %'.

# TẠO INDEX VỚI SQLSERVER

❖ Chỉ mục phủ không tận dụng được chỉ mục → index scan



The screenshot displays the SQL Server Enterprise Manager interface. At the top, there are three tabs: 'SQLQuery3.sql - HT...PECTRE\hthvy (53))\*', 'Filter index.sql - H...eWorks2012 (sa (52))', and 'SQLQuery4.sql - not connected'. The active tab shows a SQL script with the following lines:

```
34 --filter index
35 SELECT OrderDate FROM Sales.SalesOrderHeader
36 WHERE ShipDate IS NULL
37 ORDER BY OrderDate ASC;
38
39 CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_OrderDate_INC_ShipDate
40 ON Sales.SalesOrderHeader (OrderDate,ShipDate)
```

Below the script, the 'Execution plan' tab is selected. It shows the following text:

Query 1: Query cost (relative to the batch): 100%

SELECT OrderDate FROM Sales.SalesOrderHeader WHERE ShipDate IS NULL ORDER BY OrderDate ASC

Missing Index (Impact 96.9755): CREATE NONCLUSTERED INDEX [<Name of Missing Index,

At the bottom, the execution plan is visualized. It shows a single step: 'Index Scan (NonClustered... [SalesOrderHeader].[IX\_...'. The cost for this step is 100%. A legend on the left indicates that the 'SELECT' operation has a cost of 0%.

# TẠO INDEX VỚI SQLSERVER

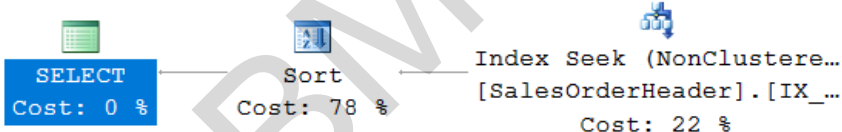
```
SQLQuery3.sql - HT...PECTRE\hthvy (53))* X Filter index.sql - H...eWorks2012 (sa (52)) SQ
34 --filter index
35 SELECT OrderDate FROM Sales.SalesOrderHeader
36 WHERE ShipDate IS NULL
37 ORDER BY OrderDate ASC;
38
39 --CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_Or
40 --(OrderDate ASC) WHERE ShipDate IS NULL;
41 CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_Orde
42 ON Sales.SalesOrderHeader (ShipDate) include(Or
43
```

100 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT OrderDate FROM Sales.SalesOrderHeader WHERE ShipDate IS



## Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032864 (22%)
Estimated Subtree Cost	0.0032864
Estimated CPU Cost	0.0001614
Estimated Number of Executions	1
Estimated Number of Rows	4
Estimated Row Size	15 B
Ordered	True
Node ID	1

## Object

[AdventureWorks2012].[Sales].[SalesOrderHeader].  
[IX\_SalesOrderHeader\_OrderDate\_INC\_ShipDate]

## Output List

[AdventureWorks2012].[Sales].  
[SalesOrderHeader].OrderDate

## Seek Predicates

Seek Keys[1]: Prefix: [AdventureWorks2012].[Sales].  
[SalesOrderHeader].ShipDate = Scalar Operator(NULL)

# KẾT LUẬN

- ☺ Index giúp tăng tốc độ truy vấn CSDL, tiết kiệm chi phí và thời gian khi scan trên bảng dữ liệu lớn
  - ☹ Tốn không gian lưu trữ
  - ☹ Tốn thời gian tạo dựng index và update dữ liệu
- ➔ *cần nhắc lựa chọn index thế nào cho tối ưu?*





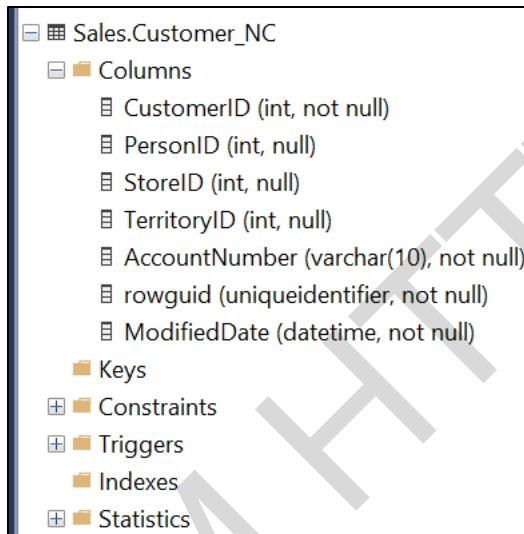
# Thank You !

 Bộ môn HTTT – Khoa CNTT – ĐH KHTN

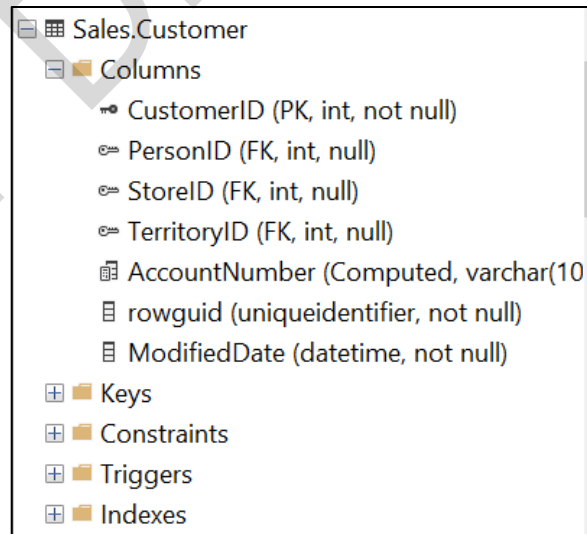
# Ví dụ về chỉ mục

## ❖ Cho hai table:

- Sales.Customer\_NC (chưa tạo khoá, chỉ mục)
- Sales.Customer (đã tạo khoá chính)



Sales.Customer_NC	
Columns	
CustomerID	(int, not null)
PersonID	(int, null)
StoreID	(int, null)
TerritoryID	(int, null)
AccountNumber	(varchar(10), not null)
rowguid	(uniqueidentifier, not null)
ModifiedDate	(datetime, not null)
Keys	
Constraints	
Triggers	
Indexes	
Statistics	



Sales.Customer	
Columns	
CustomerID	(PK, int, not null)
PersonID	(FK, int, null)
StoreID	(FK, int, null)
TerritoryID	(FK, int, null)
AccountNumber	(Computed, varchar(10))
rowguid	(uniqueidentifier, not null)
ModifiedDate	(datetime, not null)
Keys	
Constraints	
Triggers	
Indexes	

# Ví dụ về chỉ mục

SQLQuery1.sql - HT...PECTRE\hthvy (61))\*

```
9      -- #1
10     SELECT CustomerID,[StoreID]
11     FROM Sales.Customer WHERE CustomerID = 27684
12
13     -- #2
14     SELECT CustomerID,[StoreID]
15     FROM Sales.Customer_NC WHERE CustomerID = 27684
```

146 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 2%  
SELECT CustomerID,[StoreID] FROM Sales.Customer WHERE CustomerID = 27684

SELECT  
Cost: 0 %

Clustered Index Seek (C...  
[Customer].[PK\_Customer...  
Cost: 100 %

Query 2: Query cost (relative to the batch): 98%  
-- #2 SELECT CustomerID,[StoreID] FROM Sales.Customer\_NC WHERE CustomerID = 27684

SELECT  
Cost: 0 %

Table Scan  
[Customer\_NC]  
Cost: 100 %



# Ví dụ về chỉ mục

SQLQuery1.sql - HT...PECTRE\hthvy (61)\*

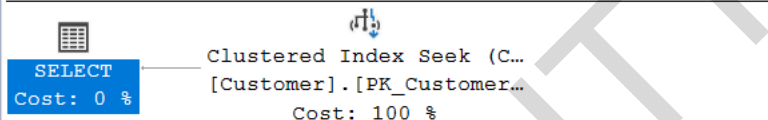
```
6 GO
7 CREATE INDEX Idx_CustomerID_NC ON Sales.Customer_NC(CustomerID)
8 GO
9 -- #1
10 SELECT CustomerID,[StoreID]
11 FROM Sales.Customer WHERE CustomerID = 27684
12 -- #2
13 SELECT CustomerID,[StoreID]
14 FROM Sales.Customer_NC WHERE CustomerID = 27684
15
```

133 %

Messages Execution plan

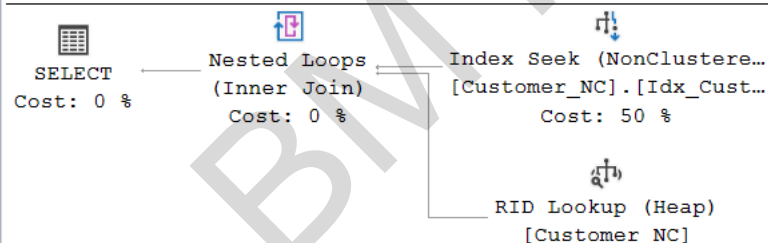
Query 1: Query cost (relative to the batch): 33%

SELECT CustomerID,[StoreID] FROM Sales.Customer WHERE CustomerID = 27684



Query 2: Query cost (relative to the batch): 67%

-- #2 SELECT CustomerID,[StoreID] FROM Sales.Customer\_NC WHERE CustomerID = 27684

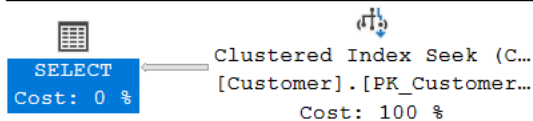


# Ví dụ về chỉ mục

10 Clustered Index là một covering index với độ bao phủ là toàn bộ các cột trong bảng, nhưng khi  
17 SELECT CustomerID,[StoreID]  
18 FROM Sales.Customer WHERE CustomerID BETWEEN 20000 and 30000  
19  
20 SELECT CustomerID,[StoreID]  
21 FROM Sales.Customer\_NC WHERE CustomerID BETWEEN 20000 and 30000

Query 1: Query cost (relative to the batch): 30%

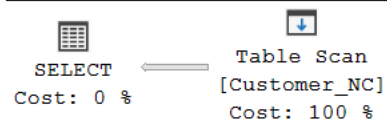
SELECT CustomerID,[StoreID] FROM Sales.Customer WHERE CustomerID BETWEEN 20000 and 30000



Query 2: Query cost (relative to the batch): 70%

SELECT CustomerID,[StoreID] FROM Sales.Customer\_NC WHERE CustomerID BETWEEN 20000 and 30000

Missing Index (Impact 90.9092): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Sales].[Cus



CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]  
ON [Sales].[Customer\_NC] ([CustomerID])  
INCLUDE ([StoreID])

# Ví dụ về chỉ mục

133 %

Messages

Execution

Query 1: Query (

select Customer

SELECT

Cost: 0 %

Table Scan

Cost: 0 %

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.117199
Estimated Operator Cost	0.139158 (100%)
Estimated CPU Cost	0.021959
Estimated Subtree Cost	0.139158
Estimated Number of Executions	1
Estimated Number of Rows for All Executions	19820
Estimated Number of Rows Per Execution	19820
Estimated Number of Rows to be Read	19820
Estimated Row Size	15 B
Ordered	False
Node ID	0

Object

[AdventureWorks2012].[Sales].[Customer\_NC]

Output List

[AdventureWorks2012].[Sales].[Customer\_NC].CustomerID,

[AdventureWorks2012].[Sales].[Customer\_NC].StoreID

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.117199
Estimated Operator Cost	0.139158 (100%)
Estimated CPU Cost	0.021959
Estimated Subtree Cost	0.139158
Estimated Number of Executions	1
Estimated Number of Rows for All Executions	19820
Estimated Number of Rows Per Execution	19820
Estimated Number of Rows to be Read	19820
Estimated Row Size	15 B
Ordered	False
Node ID	0

Object

[AdventureWorks2012].[Sales].[Customer\_NC]

Output List

[AdventureWorks2012].[Sales].[Customer\_NC].CustomerID,

[AdventureWorks2012].[Sales].[Customer\_NC].StoreID

Query executed successfully

# Ví dụ về chỉ mục

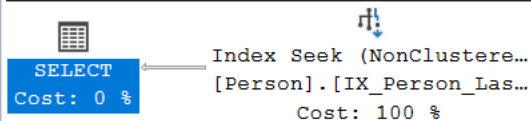
```
26 select [FirstName],[MiddleName],[LastName]
27 from [Person].[Person]
28 where LastName like 'A1%'
29
30 select [FirstName],[MiddleName],[LastName]
31 from [Person].[Person]
32 where LastName like '%A1%'
```

133 %

Messages Execution plan

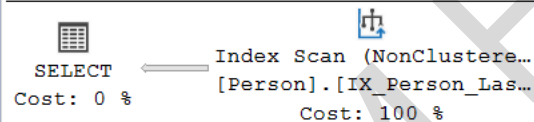
Query 1: Query cost (relative to the batch): 5%

select [FirstName],[MiddleName],[LastName] from [Person].[Person] where LastName like 'A1%'



Query 2: Query cost (relative to the batch): 95%

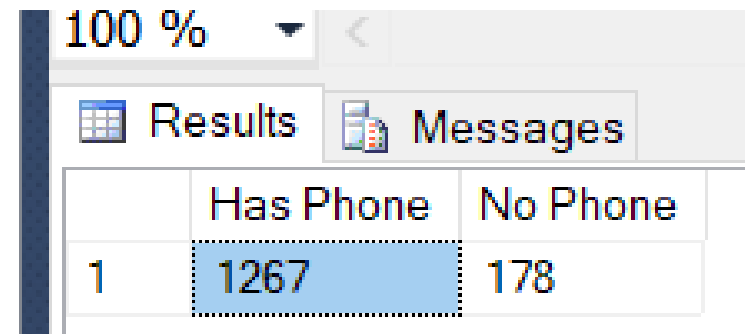
select [FirstName],[MiddleName],[LastName] from [Person].[Person] where LastName like '%A1%'



# DEMO FILTER INDEX

- ❖ The sales.customers table has the phone column which contains many NULL values:

```
SELECT
    SUM(CASE
        WHEN phone IS NULL
        THEN 1
        ELSE 0
    END) AS [Has Phone],
    SUM(CASE
        WHEN phone IS NULL
        THEN 0
        ELSE 1
    END) AS [No Phone]
FROM sales.customers
```



100 %		
Results Messages		
	Has Phone	No Phone
1	1267	178

# DEMO FILTER INDEX

- ❖ This **phone** column is a good candidate for the filtered index.
- ❖ This statement creates a filtered index for the phone column of the sales.customers table:

```
CREATE INDEX ix_cust_phone  
ON sales.customers(phone)  
WHERE phone IS NOT NULL;
```

# DEMO FILTER INDEX

Filter index-2.sql -...-SPECTRE\hthvy (55)\* X Filter index.sql - H...-SPECTRE\hthvy (52))

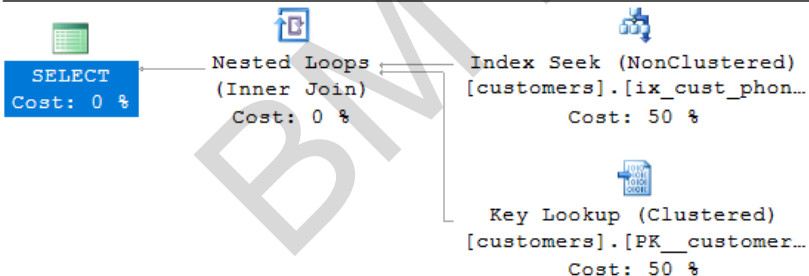
```
16 FROM sales.customers
17 go
18 CREATE INDEX ix_cust_phone
19 ON sales.customers(phone)
20 WHERE phone IS NOT NULL;
21 --The following query finds the customer whose phone number
22 SELECT
23     first_name,
24     last_name,
25     phone
26 FROM
27     sales.customers
28 WHERE phone = '(281) 363-3309';
```

100 % <

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT first\_name, last\_name, phone FROM sales.customers WHERE phone = '(281) 363-3309';





# DEMO FILTER INDEX

```
customers(phone)
IS NOT NULL;
wing query finds
```

ame,  
me,

tive to the batch): 10  
ame, phone FROM sales.

Index Seek (NonClustered)  
[customers].[ix\_cust\_phone]  
Cost: 50 %

Key Lookup (Clustered)  
[customers].[PK\_customer]  
Cost: 50 %

## Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032831 (50%)
Estimated Subtree Cost	0.0032831
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	16 B
Ordered	True
Node ID	1

## Object

[BIKESTORES].[sales].[customers].[ix\_cust\_phone]

## Output List

[BIKESTORES].[sales].[customers].customer\_id,  
[BIKESTORES].[sales].[customers].phone

## Seek Predicates

Seek Keys[1]: Prefix: [BIKESTORES].[sales].  
[customers].phone = Scalar Operator("(281) 363-3309")

```
ix_cust_phone  
customers(phone)  
IS NOT NULL;  
ing query find
```

me,  
e,

ve to the batch):  
e, phone FROM sale

Index Seek (NonClustered)  
[customers].[ix\_cust\_p]  
Cost: 50 %

Key Lookup (Clustered)  
[customers].[PK\_custo]  
Cost: 50 %

## Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032831 (50%)
Estimated Subtree Cost	0.0032831
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	267 B
Ordered	True
Node ID	3

## Object

[BIKESTORES].[sales].[customers].  
[PK\_customer\_CD65CB851DAC0316]

## Output List

[BIKESTORES].[sales].[customers].first\_name,  
[BIKESTORES].[sales].[customers].last\_name

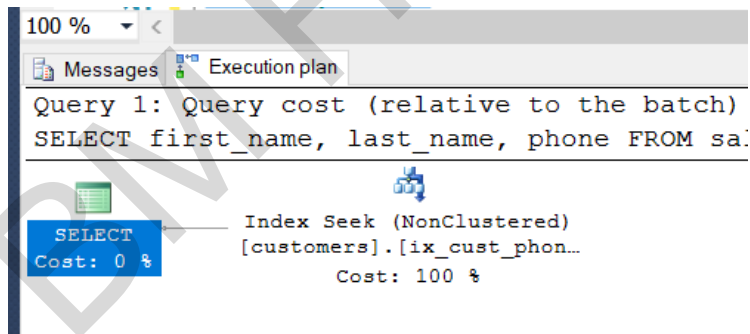
## Seek Predicates

Seek Keys[1]: Prefix: [BIKESTORES].[sales].  
[customers].customer\_id = Scalar Operator  
([BIKESTORES].[sales].[customers].[customer\_id])

# DEMO FILTER INDEX

- ❖ to **improve** the **key lookup**, you can use an **index with included columns**, which includes both first\_name and last\_name columns in the index:

```
CREATE INDEX ix_cust_phone  
ON sales.customers(phone)  
INCLUDE (first_name, last_name)  
WHERE phone IS NOT NULL;
```



# DEMO FILTER INDEX

```
27 sales.cu
28 WHERE phone
29 --to improve
30 --which incl
31 CREATE INDEX
32 ON sales.cus
33 INCLUDE (C)
```

100 %

Messages Execution plan

Query 1: Query cost (relative to query plan)

SELECT first\_name, last\_name

Index Seek (NonClustered)

Cost: 0 %

Cost: 100 %

Query executed successfully.

**Index Seek (NonClustered)**

Scan a particular range of rows from a nonclustered index.

<b>Physical Operation</b>	Index Seek
<b>Logical Operation</b>	Index Seek
<b>Estimated Execution Mode</b>	Row
<b>Storage</b>	RowStore
<b>Estimated I/O Cost</b>	0.003125
<b>Estimated Operator Cost</b>	0.0032831 (100%)
<b>Estimated Subtree Cost</b>	0.0032831
<b>Estimated CPU Cost</b>	0.0001581
<b>Estimated Number of Executions</b>	1
<b>Estimated Number of Rows</b>	1
<b>Estimated Row Size</b>	270 B
<b>Ordered</b>	True
<b>Node ID</b>	0

**Object**

[BIKESTORES].[sales].[customers].[ix\_cust\_phone]

**Output List**

[BIKESTORES].[sales].[customers].first\_name,  
[BIKESTORES].[sales].[customers].last\_name,  
[BIKESTORES].[sales].[customers].phone

**Seek Predicates**

Seek Keys[1]: Prefix: [BIKESTORES].[sales].  
[customers].phone = Scalar Operator([@1])

# To compare execution plans

❖ <https://sqlserverfast.com/epr/plan-properties/>

SQLQuery1.sql - HT...PECTRE\hthvy (64)\*

Execution plan

select \* from THAMGIADT where magv in ( select magv from GIANGVIEN gv...

SELECT Cost:...

Nested... (Inner... Cost: 1... 0.002s 40 of 24 (166...)

Index Seek (NonClu... [GIANGVIEN].[i\_GV\_... Cost: 40 % 0.002s 10 of 10 (100%)

Clustered Index Seek...

C:\Users\hthvy\Desktop\QueryPlan\_1.sqlplan

select \* from GIANGVIEN gv, THAMGIADT tg where gv.magv = tg.MAGV and...

SELECT Cost:...

Nested... (Inner... Cost: 1... 0.000s 40 of 24 (166...)

Clustered Index Sca... [GIANGVIEN].[PK\_gv]... Cost: 40 % 0.000s 10 of 10 (100%)

Showplan Analysis

Statement Options Multi Statement Scenarios

Properties

Top Plan Bottom Plan

SELECT SELECT

Actual Number 40 Actual Num 40

Cached plan si 32 KB Cached plan 40 KB

CardinalityEstir 70 CardinalityE 70

CompileCPU 2 CompileCPI 7

CompileMemor 240 CompileMer 264

CompileTime 2 CompileTim 7

Degree of Para 1 Degree of P 1

Estimated Nurr 0 Estimated N 0

Estimated Nurr 24.4898 Estimated N 24.4898

Estimated Ope 0 (0%) Estimated C 0 (0%)

Estimated Sub 0.0081625 Estimated S 0.0081701

MemoryGrantlr MemoryGra

Optimization Le FULL Optimization FULL

OptimizerHardv OptimizerH

QueryHash 0xB577A8E4CA33 QueryHash 0x7B13326361

QueryPlanHasl 0xAF5E08F20FA3 QueryPlanH 0x07ABE28B1

QueryTimeStat QueryTimeS

Reason For Ea Good Enough Plan Four Reason For Good Enough Plan I

RetrievedFrom true RetrievedFr true

SecurityPolicyA False SecurityPoli False

Set Options ANSI\_NULLS: True, AN Set Options ANSI\_NULLS: True,

Statement select \* from THAM Statement select \* from G

# TIPS

## ❖ Sử dụng hàm trong điều kiện WHERE

- `SELECT * FROM Employees WHERE YEAR(HireDate) = 2022;`
- Sửa:
- `SELECT * FROM Employees WHERE HireDate BETWEEN '2022-01-01' AND '2022-12-31';`

# TIPS

## ❖ Sử dụng toán tử không thể lập chỉ mục

- `SELECT * FROM Products WHERE ProductName LIKE '%Phone';`

## ❖ Sử dụng chỉ mục trên cột có tính phân biệt thấp

- ví dụ: cột Gender có hai giá trị 'Nam' và 'Nữ'
- `SELECT * FROM Employees WHERE Gender = 'Nam';`

## ❖ Cập nhật lượng dữ liệu lớn hoặc thao tác hàng loạt

- `UPDATE Orders SET Status = 'Processed' WHERE OrderDate < '2023-01-01';`

## ❖ Truy vấn trả về quá nhiều bản ghi

- 30-40% tổng số bản ghi của bảng → table scan





## ❖ Tập hợp chỉ mục không đầy đủ hoặc không phù hợp

- Nếu chỉ mục không bao gồm tất cả các cột cần thiết cho một truy vấn, SQL Server sẽ cần phải tham chiếu thêm bảng gốc để tìm nạp các cột không được lập chỉ mục, điều này có thể làm giảm lợi ích của chỉ mục.