Quản Trị Dữ Liệu Với Microsoft SQL Server

Chương: 11

Indexes

Mục tiêu

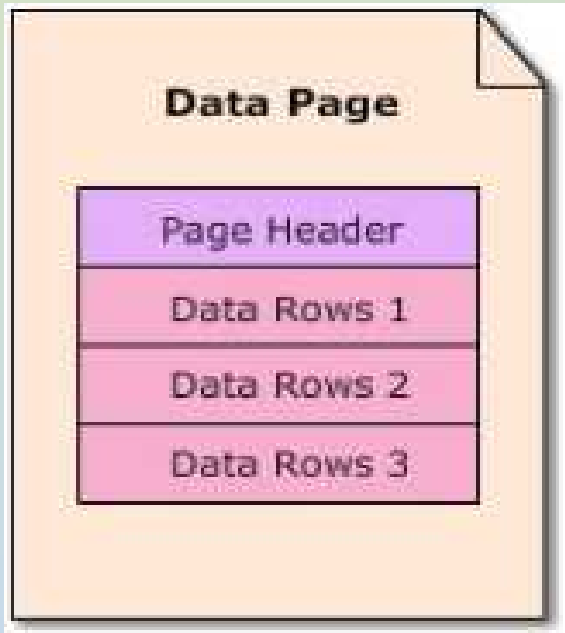
- Định nghĩa và giải thích về index
- Mô tả hiệu suất của index
- Giải thích kiến trúc của Index
  - Mô tả cấu trúc của clustered index và non clustered index
  - Mô tả cách SQL Server sử dụng index để truy xuất dữ liệu
- Giải thích về phân vùng dữ liệu
- Giải thích các bước để hiển thị hiệu suất truy vấn dữ liệu bằng index

Giới thiệu

- Khi thực hiện một truy vấn, SQL Server 2012 sử dụng các index(chỉ mục) để tìm dữ liệu.
- Nếu không định nghĩa bất kỳ một index nào cho việc tìm kiếm, thì bộ máy của SQL Server sẽ phải duyệt từng dòng trong bảng.
- Khi dữ liệu của bảng lên tới hàng ngàn, hàng triệu dòng và có thể hơn thế nữa, công việc quét(scan) toàn bộ bảng sẽ trở lên chậm hơn và làm tăng thêm các chi phí(thời gian, tài nguyên…).
- Vì vậy, mà index được khuyến khích nên dùng.
- Việc tạo hay gỡ bỏ các index không làm ảnh hưởng gì đến code của ứng dụng.
- Các thao tác index là ở phía backend(chạy phía sau) với hỗ trợ của bộ máy csdl(database engine).
- Hơn nữa, việc tạo ra index phù hợp có thể làm tăng hiệu suất đáng kể cho ứng dụng.

Tổng quan lưu trữ dữ liệu của SQL Server

- Một quyển sách có nhiều trang, trong mỗi trang có chứa các đoạn văn được hình nên bởi các câu. Tương tự, SQL Server 2012 lưu trữ dữ liệu trong đơn vị lưu trữ là data page (trang dữ liệu).
- SQL Server dùng data page để lưu trữ các dòng dữ liệu của bảng. Mỗi data page có kích thước 8 Kilo Bytes (KB).
- Mỗi Mega Byte của csdl sẽ có 128 trang dữ liệu.
- Mỗi trang được bắt đầu bằng 96 byte header lưu các thông tin về trang như:
  - Số trang (Page number)
  - Kiểu trang (Page type)
  - Không gian còn trống trên trang
  - Con trỏ trỏ đến các trang trước hoặc kế tiếp



Tổng quan lưu trữ dữ liệu của SQL Server

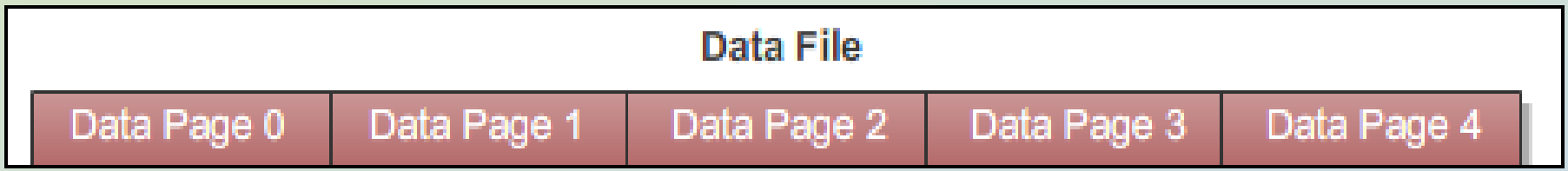
- SQL Server sử dụng một trong hai phương pháp sau để tổ chức các trang của bảng:
  - Các bảng có một clustered index, còn gọi là bảng clustered.
  - Các bảng không có clustered index, còn gọi là heap.
- Khi tạo clustered hoặc nonclustered index cho bảng, các trang index cũng được tạo ra cho bảng để lưu các khóa.
- Các trang index giúp cho có thể truy xuất trực tiếp tới dòng bất kỳ trong bảng

Data Pages

Page 4		Page 5		Page 6		Page 7		Page 8		Page 9	
Con	...	Rudd	...	Akhtar	...	Smith	...	Martin	...	Ganio	...
Funk	...	White	...	Funk	...	Ota	...	Phua	...	Jones	...
White	...	Barr	...	Smith	...	Jones	...	Jones	...	Hall	...
...	...	...	...	Martin	...	...	...	Smith	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...

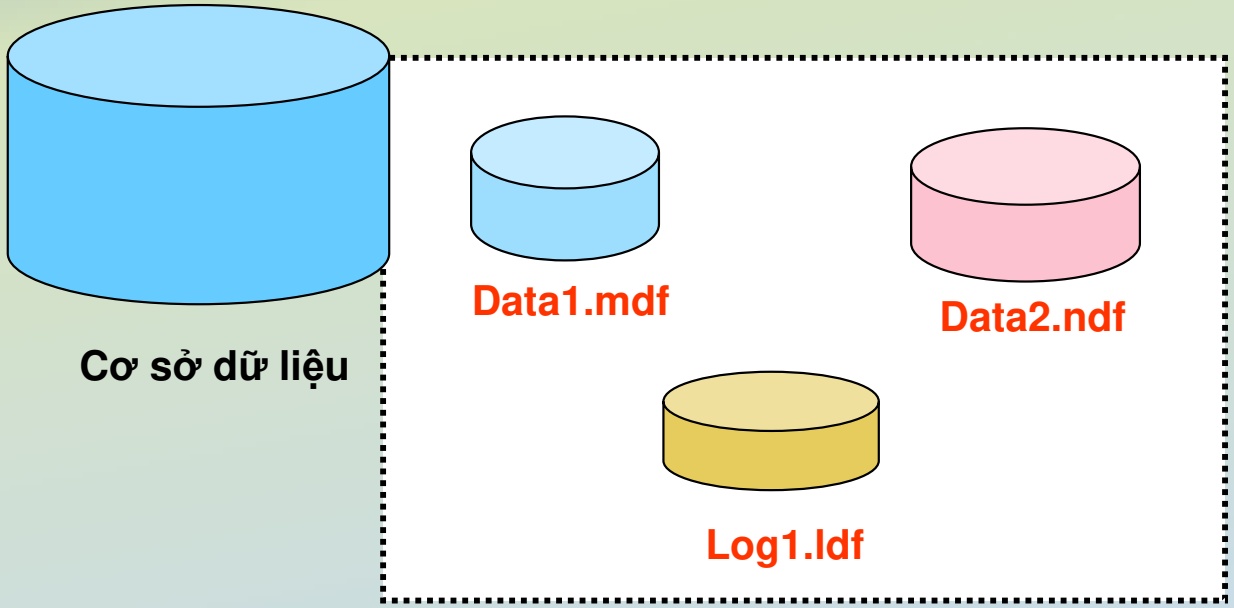
Các tập tin dữ liệu

- Tất cả thao tác nhập/xuất trong csdl được thực hiện ở mức trang. Có nghĩa là bộ máy csdl đọc hoặc ghi các trang dữ liệu.
- Một tập 8 trang dữ liệu(data page) liên tiếp được xem như là một extend. SQL Server 2012 lưu các trang dữ liệu trong các tập tin data file.
- Không gian cấp phát cho một data file được chia thành một dãy các trang dữ liệu được đánh số bắt đầu từ 0



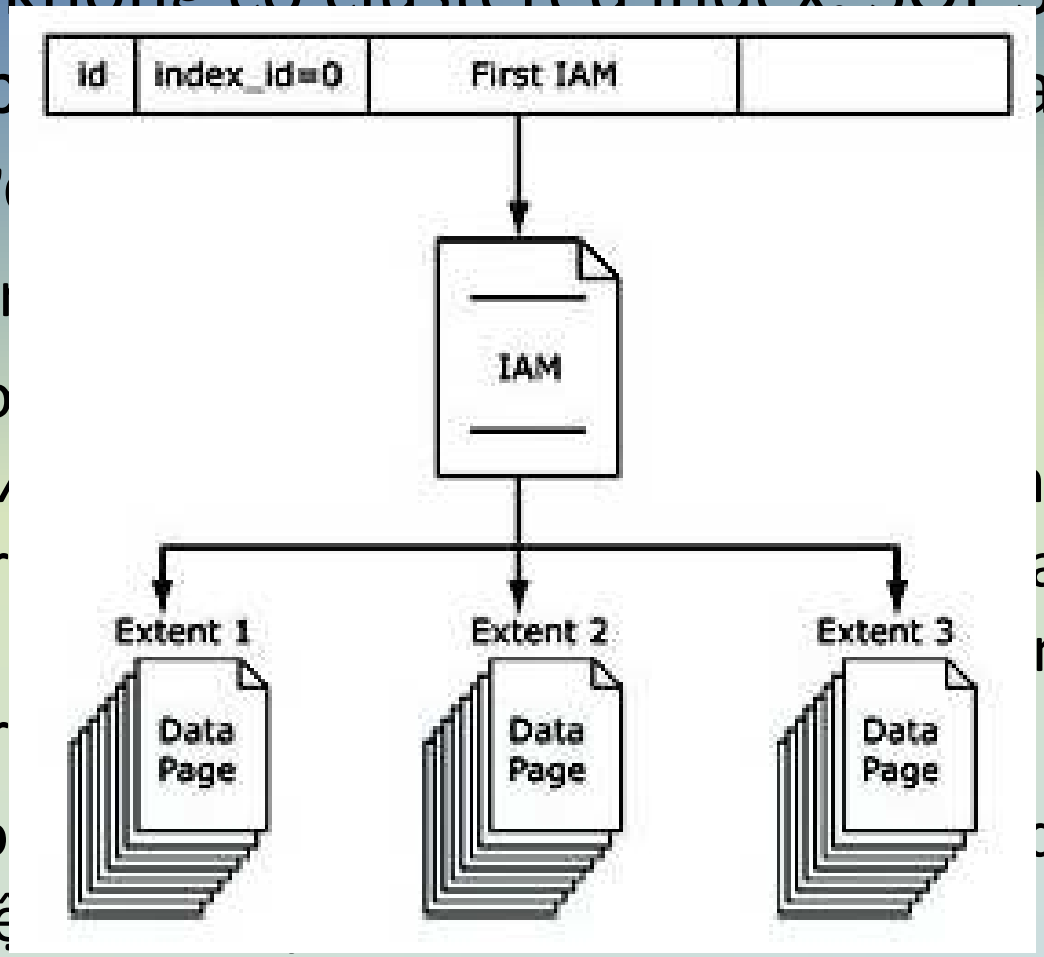
Các tập tin dữ liệu

- Một csdl SQL Server 2012 có ba kiểu tập tin :
  - Primary Data Files (.mdf)
  - Secondary Data Files (.ndf)
  - Transaction Log Files (.ldf)



SQL Server truy xuất dữ liệu như thế nào

- Với các bảng không có clustered index, SQL Server truy xuất dữ liệu b…able scan). Một bảng đư…
  - Bắt đầu từ t…
  - Quét lần lượ…ng khác của bảng. *Index …*g để di chuyển từ tr…ap. Với bảng có clustered …ng để di chuyển từ tr…
  - Đọc từng dò…dòng thỏa mãn điều kiệ…
  - Là cách tốt nhất cho việc truy xuất các bảng nhỏ.

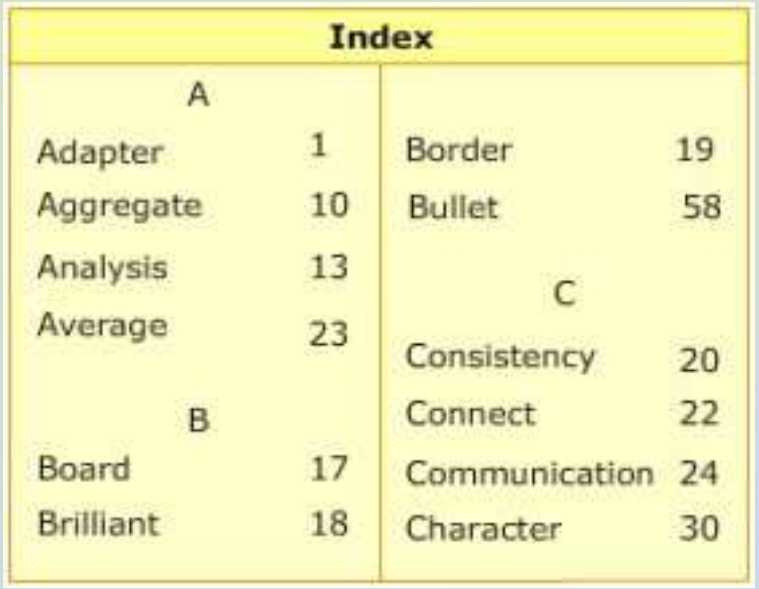


SQL Server truy xuất dữ liệu như thế nào

- Với bảng có clustered index, SQL Server sử dụng truy xuất index *(indexed access)* để truy xuất dữ liệu. Truy xuất index được thực hiện như sau:
  - Duyệt cấu trúc cây index để tìm các dòng mà truy vấn yêu cầu.
  - Đọc và lấy(extract) ra các dòng thỏa mãn điều kiện của truy vấn.
  - Là cách tốt nhất để truy xuất các dòng và dãy các dòng từ một bảng lớn.

Sự cần thiết của index(chỉ mục)

- Để thuận tiện, nhanh chóng cho việc lấy dữ liệu từ csdl, SQL Server 2012 cung cấp tính năng tạo index.
- Một index trong csdl SQL Server 2012 có chứa các thông tin cho phép bạn tìm dữ liệu được chỉ ra mà không phải quét toàn bộ bảng như hình sau:
- SQL Server sử dụng index tương tự như cách chúng ta sử dụng mục lục của quyển sách.



Index

- Trong một bảng, các bản ghi được lưu trữ theo thứ tự mà chúng được nhập vào. Việc lưu trữ chúng trong cơ sở dữ liệu không được sắp xếp.
- Khi dữ liệu được lấy từ các bảng như vậy, buộc phải quét(scan) toàn bộ bảng.
- Điều đó làm cho việc xử lý truy vấn lấy dữ liệu chậm xuống. Để tăng tốc độ truy vấn dữ liệu, các index cần phải được tạo.
- Khi index được tạo trên bảng, index tạo ra một thứ tự các dòng dữ liệu trong bảng như hình sau:

Index

- Index được tạo tự động khi ràng buộc `PRIMARY KEY` và `UNIQUE` được định nghĩa trên bảng.
- Các index làm giảm các thao tác nhập/xuất đĩa và chiếm ít tài nguyên hơn.
- Câu lệnh `CREATE INDEX` được sử dụng để tạo index
- Cú pháp để tạo một index như sau:

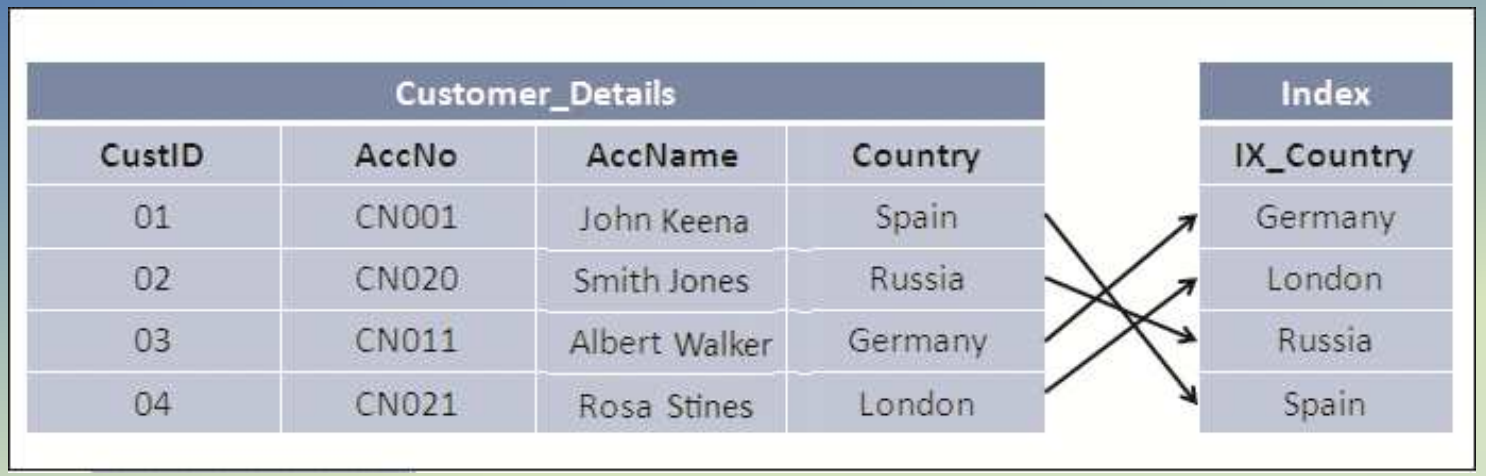
CREATE INDEX <tên_index> **ON** <tên_bảng>(tên_cột)

- Đoạn code sau đây minh họa tạo một index có tên **`IX_Country`** trên cột **`Country`** trong bảng **`Customer_Details`**.

```
USE CUST_DB
CREATE INDEX IX_Country ON Customer_Details(Country);
GO
```

Index

- Hình dưới đây minh họa bảng **`Customer_Details`** được tạo index



Customer_Details			
CustID	AccNo	AccName	Country
01	CN001	John Keena	Spain
02	CN020	Smith Jones	Russia
03	CN011	Albert Walker	Germany
04	CN021	Rosa Stines	London

Index
IX_Country
Germany
London
Russia
Spain

- Các chỉ mục (index) trỏ tới các vị trị của dòng trên trang dữ liệu thay cho việc tìm kiếm thông qua bảng.
- Hãy xem xét sự việc và hướng dẫn sau đây về index
  - Index làm tăng tốc độ các truy vấn ghép nối các bảng hoặc thực hiện các thao tác sắp xếp.
  - Index thực thi tính duy nhất của các dòng nếu nó được định nghĩa.
  - Index được tạo ra và duy trì các dòng theo thứ tự tăng dần hoặc giảm dần.

Scenario

- Trong danh mục điện thoại (telephone directory), ở đó một lượng lớn dữ liệu được lưu trữ theo thứ tự bảng chữ cái và được truy xuất thường xuyên.
- Nếu như dữ liệu đó không được sắp xếp, gần như không thể tìm kiếm một số điện thoại cụ thể.
- Tương tự, trong csdl có một số lượng lớn các bản ghi thường xuyên được truy cập, dữ liệu nên được sắp xếp để việc lấy ra được nhanh chóng.
- Khi một index được tạo trên bảng, index sẽ sắp xếp vật lý hoặc logic các bản ghi.
- Do vậy, việc tìm kiếm các bản ghi trở lên nhanh hơn, giảm sử dụng quá mức tài nguyên hệ thống.

SQL Server 2012

Tạo hay không tạo index ?

- Tạo index
  - Tăng tốc độ truy xuất dữ liệu
  - Đảm bảo tính duy nhất các dòng

Nên tạo index:

Cho khóa chính, khóa ngoại

Thường xuyên tìm kiếm theo một miền

Thường xuyên truy cập theo thứ tự

- Không tạo index
  - Tiêu tốn không gian đĩa
  - Incurs overhead

Không nên tạo index

Trên các cột hiếm khi được tham chiếu đến trong các câu truy vấn

Cột có ít giá trị duy nhất

Các cột có kiểu dữ liệu là bit, text, hoặc images

Kiến trúc Index

- Các bảng của csdl SQL Server 2012 sử dụng một trong hai phương pháp để tổ chức dữ liệu trong các trang dữ liệu trong một phân vùng (partition)
  - Nếu bảng có một clustered index, nó được gọi là bảng clustered. Các dòng dữ liệu trong bảng được lưu trữ sắp xếp theo thứ tự khóa của clustered index.
  - Nếu bảng không có clustered index, được gọi là bảng heap. Các dòng dữ liệu trong bảng không được sắp xếp theo thứ tự cụ thể nào.

Employee_Details		
EmpID	EmpName	DeptID
CN00020	Rosa	BN0001
CN00018	John	BN0020
CN00019	Smith	BN0021
CN00012	Robert	BN0011

Heap Structure

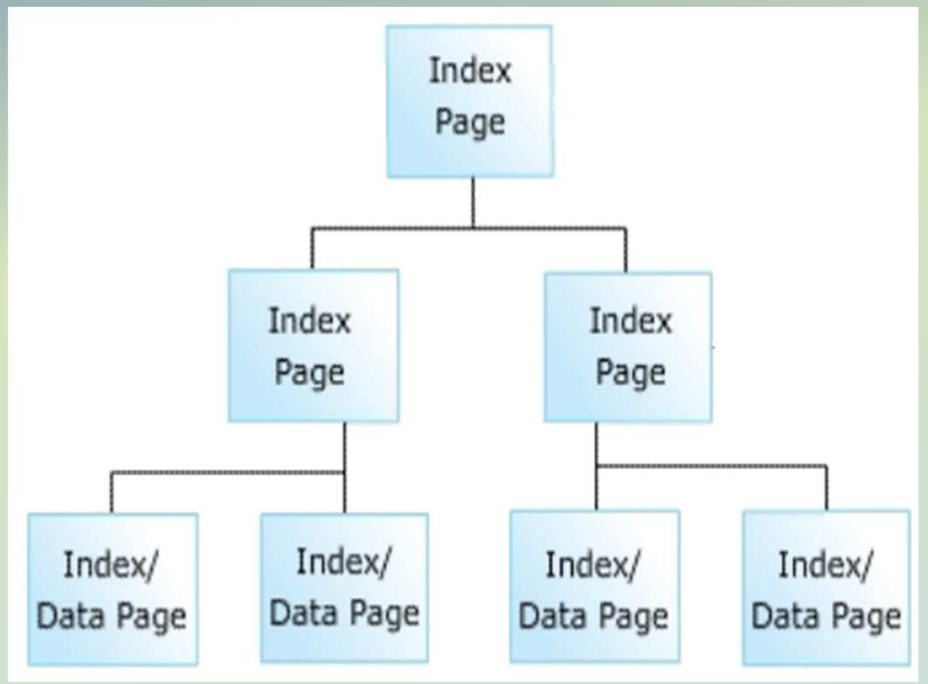
Employee_Details		
EmpID	EmpName	DeptID
CN00012	Robert	BN0011
CN00018	John	BN0020
CN00019	Smith	BN0021
CN00020	Rosa	BN0001

Clustered Structure

Tìm hiểu cấu trúc B-tree (Cây nhị phân)

- Trong SQL Server, các index được tổ chức theo dạng cây nhị phân (B-tree) như hình dưới:

- Trong cấu trúc B-Tree, có duy nhất một nút gốc(root node) nằm tại mức định (top).
- Sau đó nút gốc chia ra nhánh trong mức tiếp theo, được gọi là mức trung gian(first intermediate level) đầu tiên.
- Các nút ở mức trung gian đầu tiên có thể chia tiếp thành các nhánh nữa.
- Việc chia nhánh này, sau đó có thể tiếp tục chia thành nhiều mức trung gian, cuối cùng là mức lá(leaf level)
- Các nút ở mức lá được gọi là các nút lá (leaf nodes).



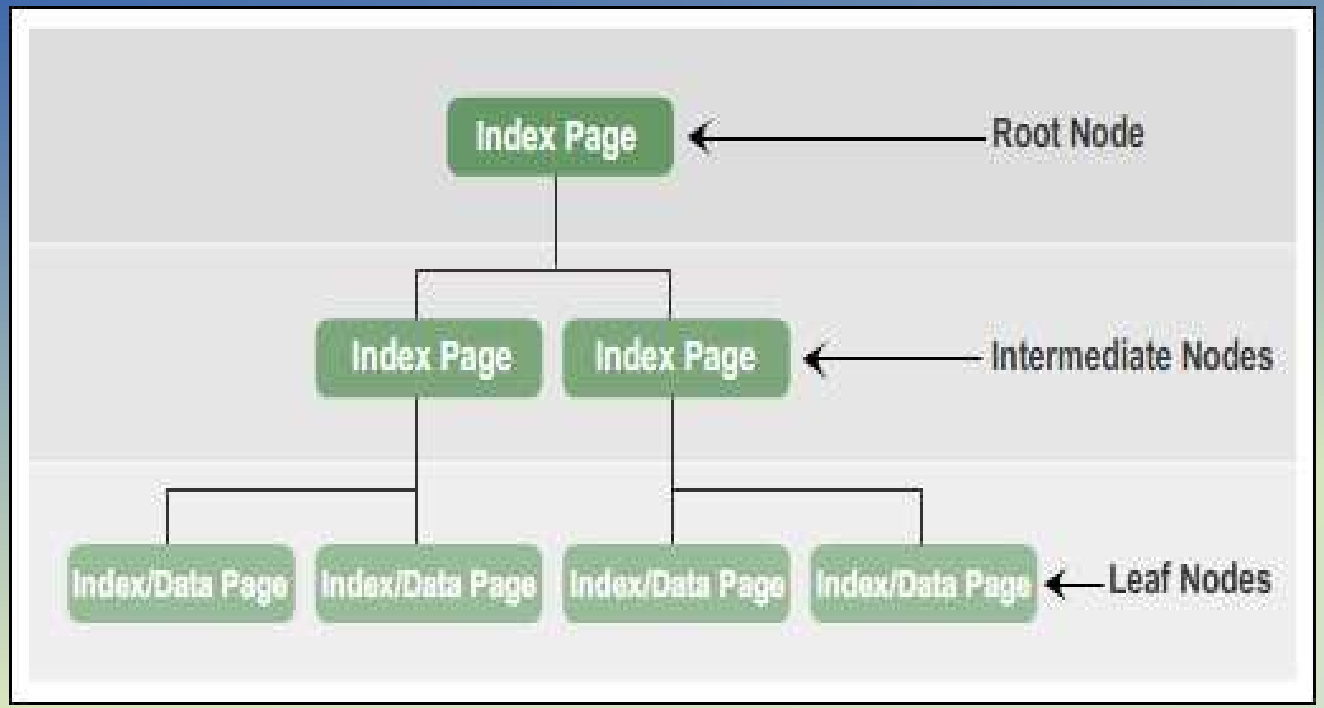
SQL Server 2012

Cấu trúc cây Index B-Tree

- Trong cấu trúc B-Tree của một index, nút gốc là một trang index (index page).
- Trang index ở mức gốc này chứa con trỏ trỏ tới các trang index có ở mức trung gian thứ nhất.
- Có thể có nhiều mức trung gian trong một cây B-Tree index.
- Các nút lá của cây B-Tree index có thể là các data page (trang dữ liệu) chứa các dòng dữ liệu hoặc cũng có thể là các trang index có chứa các dòng index trỏ đến các dòng dữ liệu như trong hình sau đây:

SQL Server 2012

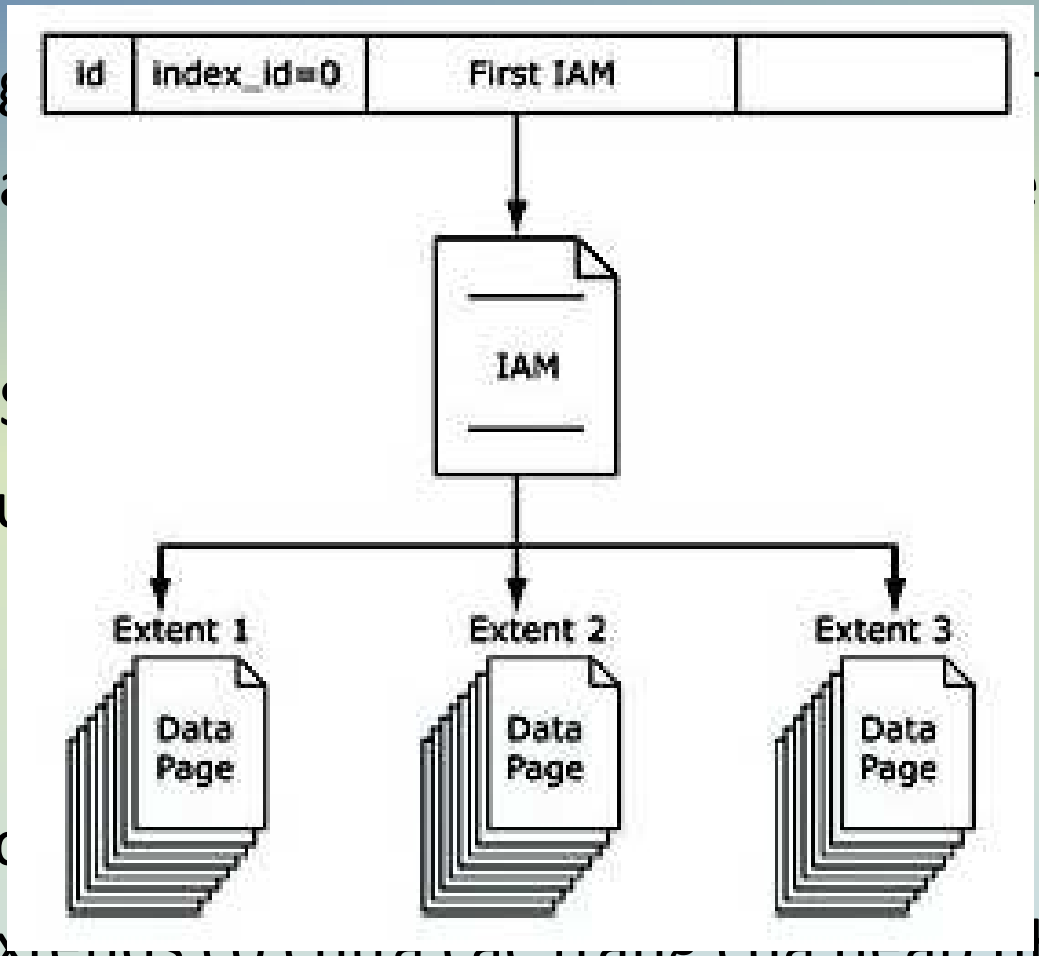
Cấu trúc cây Index B-Tree



- **Nút gốc(root node):**
  - Chứa một trang index với các con trỏ trỏ tới các trang index ở mức trung gian thứ nhất
- **Nút trung gian**
  - Chứa các trang index với các con trỏ trỏ tới các trang index ở mức trung gian tiếp theo hoặc tới các trang index hay trang dữ liệu ở mức lá
- **Nút lá**
  - Có thể là trang dữ liệu hoặc có thể là trang index trỏ tới trang dữ liệu

Cấu trúc Heap

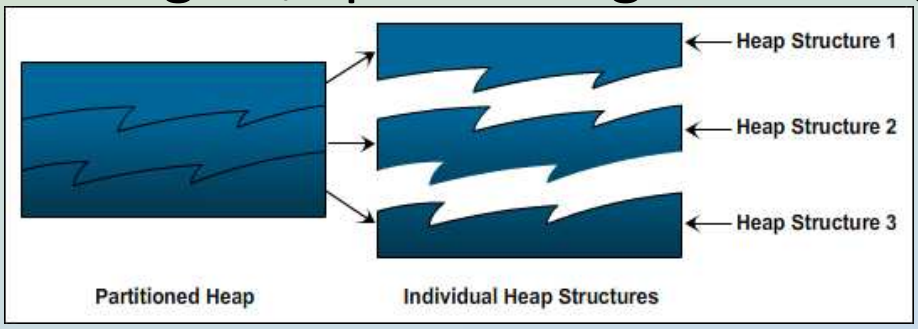
- Trong cấu trúc heap, các trang dữ liệu và các bản ghi không được sắp xếp theo thứ tự.
- Sự nối kết g thông tin được ghi trong ca x Allocation Map (IAM)]
- Trong SQL S dụng để quét qua một cấu tru
- Các extent bởi đơn vị cấp phát trong
- Một heap c trang IAM để tìm kiếm các extends có chứa các trang của heap như hình sau:



SQL Server 2012

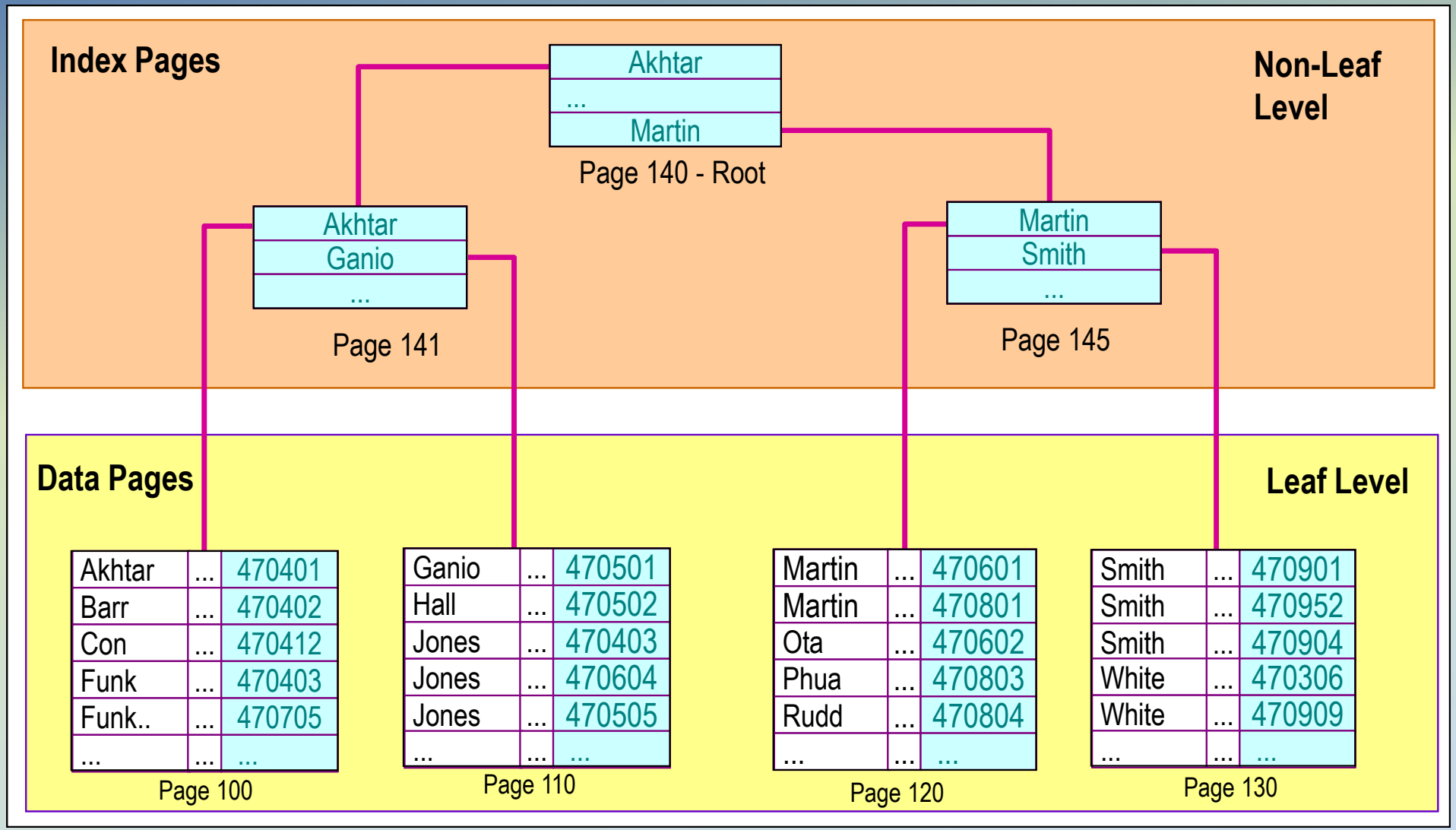
Phân vùng cấu trúc Heap

- Một bảng có thể được phân chia logic thành nhiều nhóm các dòng nhỏ hơn.
- Sự phân chia này được xem như là sự phân vùng.
- Các bảng được phân vùng để thực hiện các thao tác duy trì thêm hiệu quả.
- Mặc định, một bảng là một phân vùng.
- Khi các phân vùng được tạo trên bảng có cấu trúc heap, mỗi phân vùng sẽ chứa dữ liệu trong một cấu trúc heap riêng biệt.
- Ví dụ, nếu một heap có ba phân vùng, thì có ba cấu trúc heap hiện diện, mỗi heap trong một phân vùng như trong hình sau:



Cấu trúc Clustered Index

- Clustered index làm cho các bản ghi được lưu trữ theo thứ tự ở mức vật lý.



Tạo Clustered Index

- Clustered index được tự động tạo ra khi định nghĩa khóa chính cho bảng.
- Clustered index làm cho các bản ghi được sắp xếp ở mức vật lý (sắp xếp trong các trang) theo thứ tự của giá trị khóa.
- Cú pháp:

```
CREATE CLUSTERED INDEX <tên_index>  ON
<tên_bảng>(tên_cột)
```

Tạo Clustered Index

- Ví dụ: Tạo clustered index *IX_CustId* trên cột *CustId* trong bảng *Customer_Details*

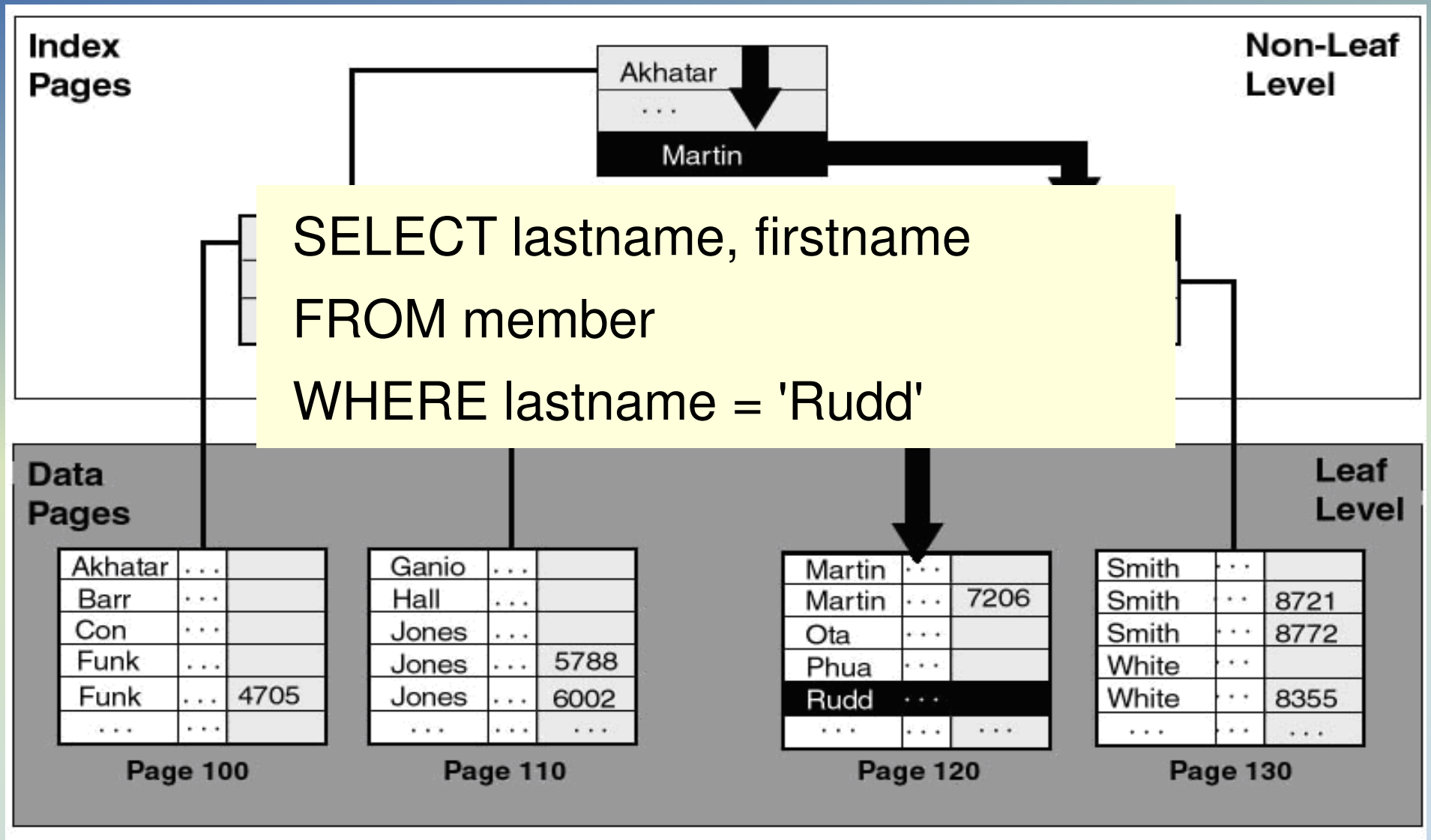
```
CREATE CLUSTERED INDEX IX_CustID ON Customer_Details(CustID)
```

- Cột được chọn để tạo clustered index cho bảng phải là cột không có gía trị trùng nhau.
- Index này sẽ tổ chức các bản ghi theo thứ tự của các giá trị trong cột index.
- Clustered index được dùng để định vị(tìm kiếm) một dòng hoặc một dãy các dòng.

Clustered Index			
CustID	First Name	Last Name	State
1	Kevin	Wilson	Alaska
2	John	Clinton	Colorado
3	Smith	Kidman	Delaware
4	Albert	Johnson	Florida
5	Sam	Williams	Georgia
6	Christopher	Jones	Hawaii

- Bắt đầu từ trang đầu tiên của cây index, giá trị tìm được so sánh với mỗi khóa(key) trên trang.
- Khi một giá khóa trùng khớp, bộ máy csdl(database engine) di chuyển tới trang được chỉ ra bởi giá trị như trong hình sau:

Truy xuất dữ liệu với Clustered Index



Truy xuất dữ liệu với Clustered Index

- Khi tạo clustered index, hãy xem xét các hướng dẫn sau đây:
  - Chỉ có thể tạo một clustered index cho mỗi bảng.
  - Thứ tự vật lý các dòng trong bảng và thứ tự các dòng trong index là như nhau.
  - Các giá trị khóa trong một clustered index phải là duy nhất.
- Trong trường hợp bảng không có khóa chính, Clustered Index nên được xem xét các cột sau:
  - Các cột khóa được tìm kiếm rộng rãi.
  - Các cột được sử dụng trong các truy vấn mà trả lại tập kết quả lớn.
  - Các cột có dữ liệu duy nhất.
  - Các cột được sử dụng việc join các bảng.

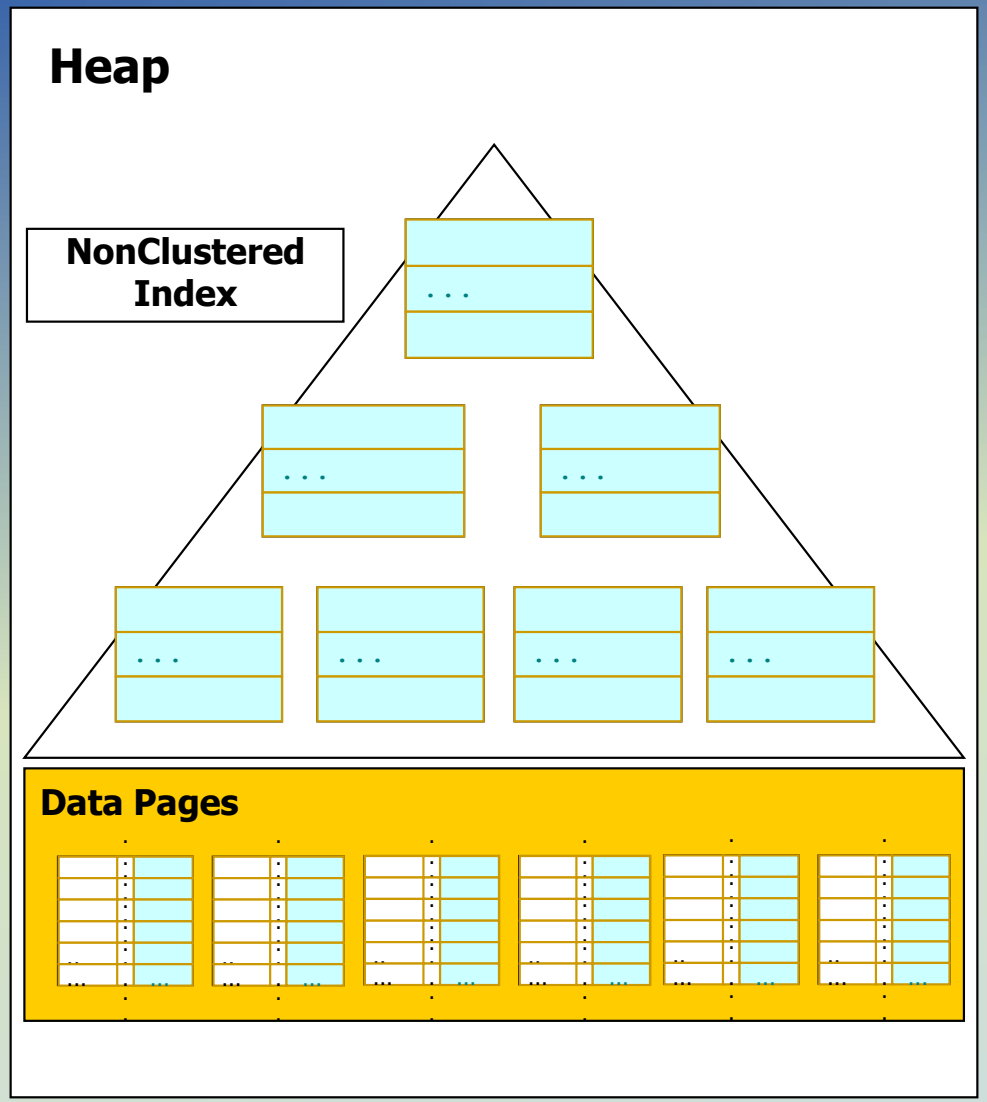
SQL Server 2012

Cấu trúc nonclustered index

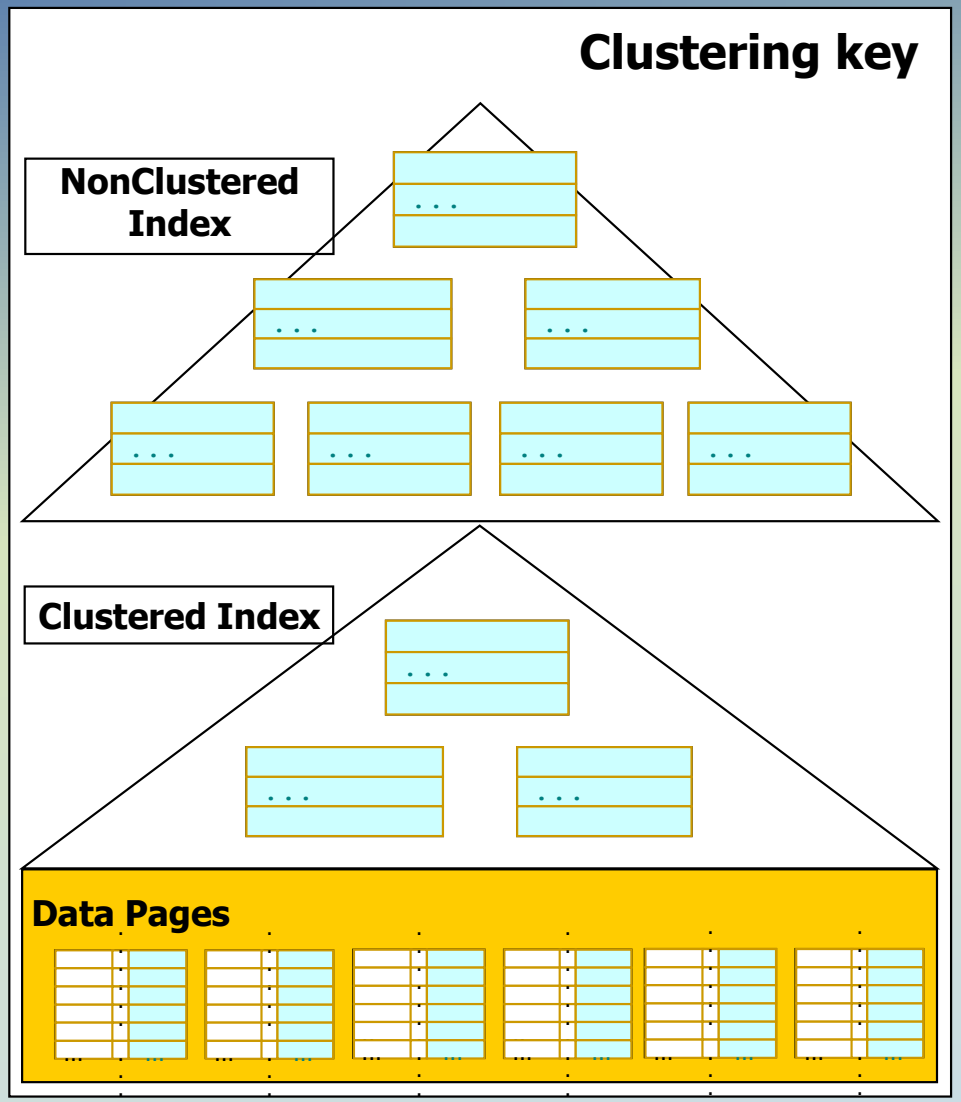
- Nonclustered index được định nghĩa trên bảng có dữ liệu được tổ chức theo cấu trúc clustered hoặc cấu trúc heap.
- Nonclustered index là loại index mặc định nếu không có index được định nghĩa trên bảng.
- Mỗi dòng trong nonclustered index có chứa một giá trị khóa nonclustered và bộ định vị dòng(row locator)
- Bộ định vị dòng trỏ tới dòng dữ liệu trong bảng tương ứng với giá trị khóa.
- Hình sau đây minh họa cấu trúc nonclustered index:

SQL Server 2012

Cấu trúc nonclustered index



Kiến trúc của một Nonclustered Index trên một Heap



Kiến trúc của một Nonclustered Index trên một bảng có clustered Index

SQL Server 2012

Cấu trúc nonclustered index

- Các nonclustered index có cấu trúc B-Tree tương tự như của clustered index nhưng có một số khác biệt sau đây:
  - Việc lưu trữ các dòng dữ liệu của bảng ở mức vật lý không theo đúng thứ tự của khóa nonclustered index được định nghĩa cho bảng.
  - Trong cấu trúc nonclustered index, các nút mức lá có chứa các dòng index (index row) .
- Nonclustered index rất hữu ích khi bạn cần có nhiều cách tìm kiếm dữ liệu.
- Một số thông tin và hướng dẫn cần được xem xét trước khi tạo một nonclustered index như sau:
  - Khi một clustered index được tạo lại hoặc tùy chọn DROP_EXISTING được sử dụng, SQL Server xây dựng lại các nonclustered index hiện có.
  - Một bảng có thể có tới 999 nonclustered index.
  - Tạo clustered index trước khi tạo ra một nonclustered index.

Tạo Nonclustered Index

- Cú pháp được sử dụng để tạo nonclustered index

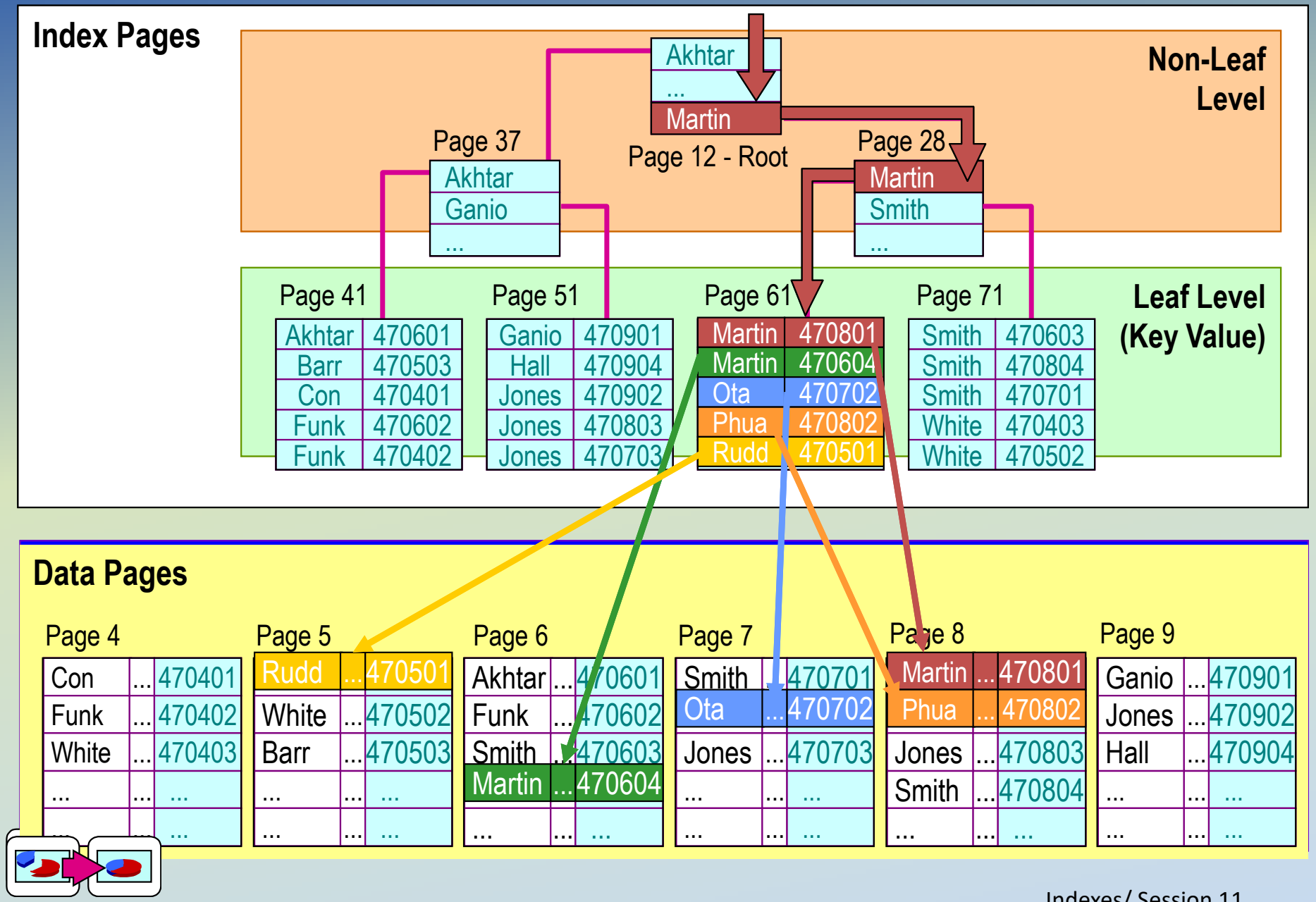
```
CREATE NONCLUSTERED INDEX <tên_index> ON
<tên_bảng> (tên_cột)
```

- Đoạn code dưới đây tạo một noclustered index **IX_State** trên cột **State** trong bảng

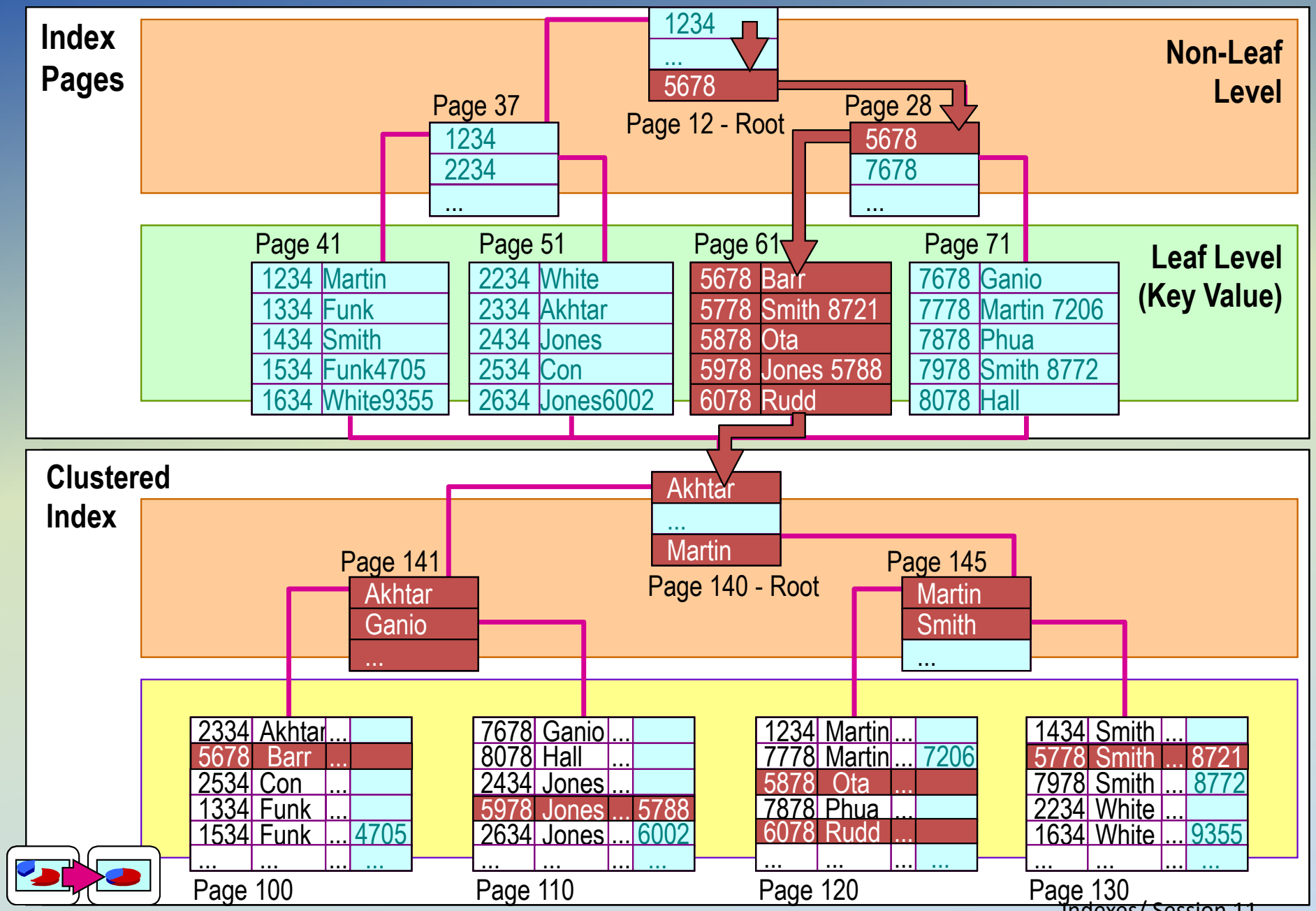
```
USE CUST_DB
CREATE NONCLUSTERED INDEX IX_State ON
Customer_Details(State)
GO
```

SQL Server 2012

Truy xuất dãy dữ liệu trong một Heap



Truy xuất dãy dữ liệu trong Clustering Key



ColumnStore Index 1-5

- ColumnStore … 2012.
- Nó cải thiện k… vào kho dữ liệu (d…
- Các index hoà… erver trước đó, dữ … dạng dòng (row-wise), nhưng columnstore index trong SQL Server được lưu trữ theo dạng cột(column-wise)
- Do đó tốc độ … tro… , vì vậy cột colum… sử du… việc nhập/xuất đĩa … cho …
- B-Tree và hea… theo dạng dòng(row-wise), có nghĩa là dữ liệu từ t… của một dòng được lưu trữ nối tiếp nhau trên cùng trang.

EmployeeID	Name	City	State
1	Ross	San Francisco	CA
2	Sherry	New York	NY
3	Gus	Seattle	WA
4	Stan	San Jose	CA
5	Lijon	Sacramento	CA

Row Store

1 Ross San Francisco CA

2 Sherry New York NY

3 Gus Seattle WA

4 Stan San Jose CA

5 Lijon Sacramento CA

Columnstore

1 2 3 4 5

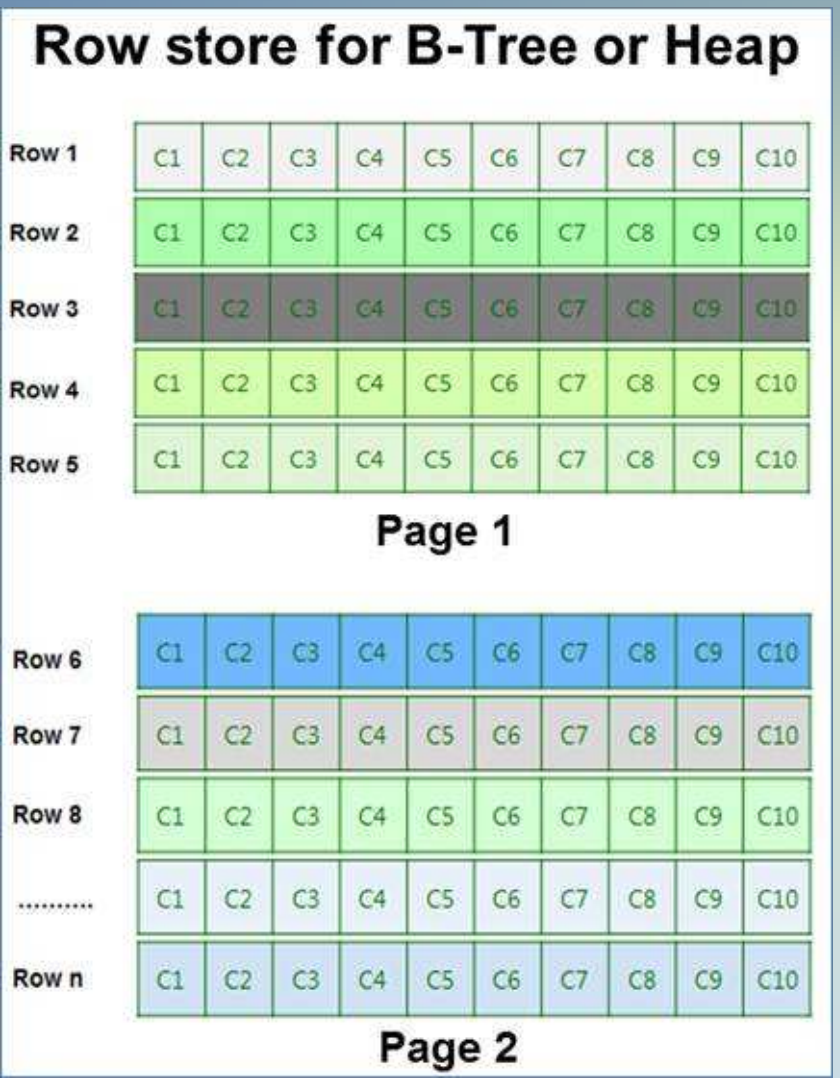
Ross Sherry Gus Stan Lijon

San Francisco New York Seattle San Jose Sacramento

CA NY WA CA CA

ColumnStore Index 2-5

- Ví dụ, Nếu một bảng với 10 cột (c1 tatới C10), dữ liệu của tất cả 10 cột từ mỗi dòng được lưu trữ liên tiếp nhau trên cùng trang như hình sau đây:

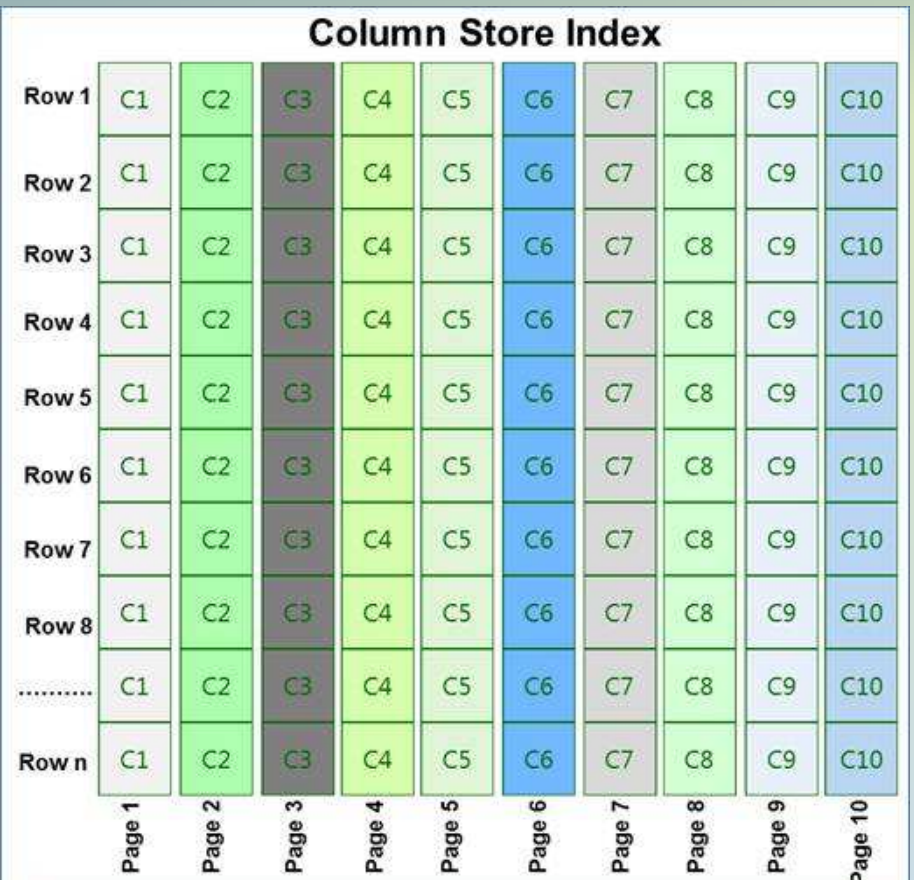


StudentID	Name	Gender	Birthday	City	Postcode	CourseCode	DateEnrolled
1	Ross Taylor	Male	17/06/1991	London	SW17 2NA	CS-110	15/02/2010
2	Jim Stott	Male	17/06/1993	Leeds	LS1 2AZ	BN-120	15/07/2011
3	Ian Hall	Male	21/10/1994	Bradford	BD1 2BL	BN-120	15/07/2011
4	Stacy Jones	Female	05/11/1992	Glasgow	G1 2AB	GS-130	15/07/2012
5	Clair Millard	Female	30/11/1991	Manchester	M1 9DB	CS-110	15/02/2012

Data Page
1 Ross Taylor Male 17/06/1991 London SW17 9NA CS-110 15/02/2010
2 Jim Stott Male 17/06/1993 Leeds LS1 2AZ BN-120 15/07/2011
3 Ian Hall Male 21/10/1994 Bradford BD1 2BL BN-120 15/07/2011
4 Stacy Jones Female 05/11/1992 Glasgow G1 2AB GS-130 15/07/2012
5 Clair Millard Female 30/11/1991 Manchester M1 9DB CS-110 15/02/2012

ColumnStore Index 3-5

- Khi cột columnstore index được tạo, các dữ liệu được lưu trữ dưới dạng cột, nghĩa là dữ liệu trong mỗi cột riêng biệt của tất cả các hàng được lưu trữ với nhau trên cùng một trang.
- Ví dụ, dữ liệu trong cột C1 của tất cả các hàng được lưu trữ cùng nhau trên một trang và dữ liệu trong cột C2 của tất cả các hàng được lưu trữ trên một trang khác và như thể hiện trong hình sau đây:



StudentID	Name	Gender	Birthday	City	Postcode	CourseCode	DateEnrolled
1	Ross Taylor	Male	17/06/1991	London	SW17 2NA	CS-110	15/02/2010
2	Jim Stott	Male	17/06/1993	Leeds	LS1 2AZ	BN-120	15/07/2011
3	Ian Hall	Male	21/10/1994	Bradford	BD1 2BL	BN-120	15/07/2011
4	Stacy Jones	Female	05/11/1992	Glasgow	G1 2AB	GS-130	15/07/2012
5	Clair Millard	Female	30/11/1991	Manchester	M1 9DB	CS-110	15/02/2012

Data Page StudentID
1
2
3
4
5

Data Page Name
Clair Millard
Ian Hall
Jim Stott
Ross Taylor
Stacy Jones

Data Page Gender
Female
Female
Male
Male
Male

Data Page Birthday
17/06/1991
30/11/1991
05/11/1992
17/06/1993
21/10/1994

Data Page City
Bradford
Glasgow
Manchester
Leeds
London

Data Page Postcode
BD1 2BL
G1 2AB
M1 9DB
LS1 2AZ
SW17 2NA

Data Page CourseCode
BN-120
BN-120
CS-110
CS-110
GS-130

Data Page DateEnrolled
15/02/2010
15/07/2011
15/07/2011
15/02/2012
15/07/2012

SQL Server 2012

Column Store Index 4-5

- Cú pháp được sử dụng để tạo columnstore index như sau:

```
CREATE [ NONCLUSTERED ] COLUMNSTORE INDEX index_name
 ON <object> ( column [ ,...n ] ) [ WITH ( <column_index_option> [ ,...n ] ) ]  ON
```

- Giả sử bảng **ResellerSalesPtnd** đã được tạo trong csdl AdventureWorks2012
- Đoạn code dưới đây minh họa cách tạo columnstore index trên bảng đó

```
USE CUST_DB
CREATE NONCLUSTERED COLUMNSTORE INDEX [csindx_ResellerSalesPtnd]
ON [ResellerSalesPtnd]
(
[ProductKey],
[OrderDateKey],
[DueDateKey],
[ShipDateKey],
[CustomerKey],
[EmployeeKey],
```

Column Store Index 5-5

```
[PromotionKey],
[CurrencyKey],
[SalesTerritoryKey],
[SalesOrderNumber],
[SalesOrderLineNumber],
[RevisionNumber],
[OrderQuantity],
[UnitPrice],
[ExtendedAmount],
[UnitPriceDiscountPct],
[DiscountAmount],
[ProductStandardCost],
[TotalProductCost],
[SalesAmount],
[TaxAmt],
[Freight],
[CarrierTrackingNumber],
[CustomerPONumber],
[OrderDate],
[DueDate],
[ShipDate]
);
```

Xóa Index 1-3

- Trong khi xóa tất cả index trên bảng, trước tiên bạn phải xóa nonclustered index, sau đó là clustered indexes.
- SQL Server 2012 có thể xóa clustered index và di chuyển heap (bảng không được sắp xếp) thành filegroup hoặc lược đồ phân vùng khác bằng tùy chọn MOVE TO.
- Tùy chọn này không hợp lệ cho nonclustered index
- Lược đồ phân vùng hoặc filegroup được chỉ ra trong mệnh đề MOVE TO phải tồn tại rồi.
- Bảng sẽ được định vị trong cùng lược đồ phân vùng hoặc filegroup của clustered index được xóa.

Xóa Index 2-3

- Cú pháp xóa clustered index:

```
DROP INDEX <index_name> ON <table_name>
[ WITH ( MOVE TO { <partition_scheme_name> (
<column_name> ) | <filegroup_name> | 'default' }) ]
```

- Đoạn code dưới đây xóa index **IX_SuppID** được tạo trên cột **SuppID** của bảng **Supplier_Details**:

```
DROP INDEX IX_SuppID ON Supplier_Details
WITH (MOVE TO 'default')
```

- Dữ liệu trong bảng **Supplier_Details** được di chuyển đến vị trí mặc định

Xóa Index 3-3

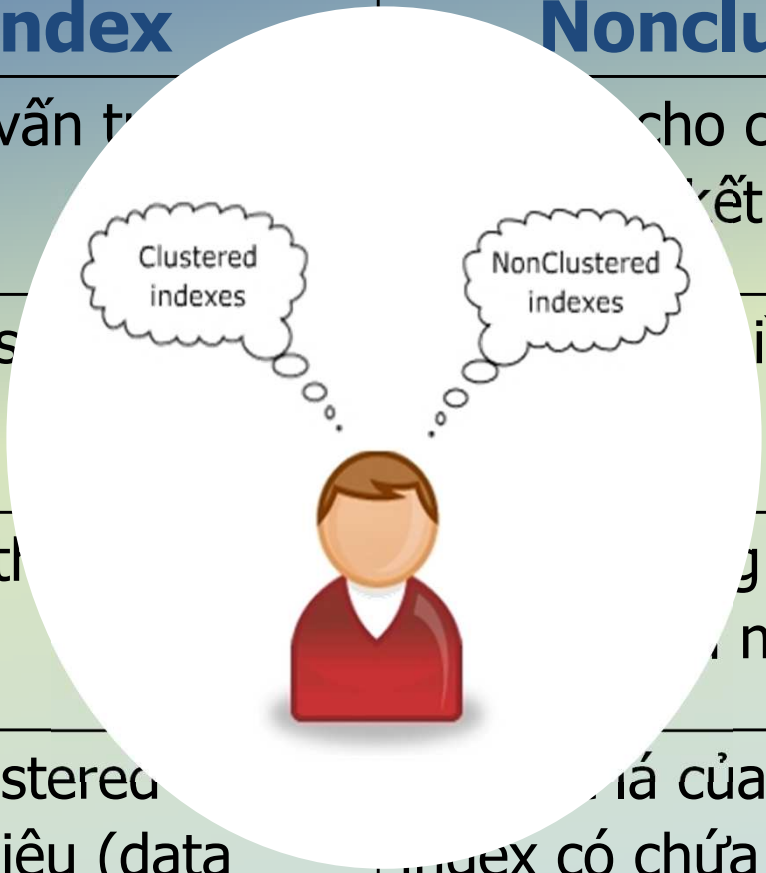
- Đoạn code dưới đây xóa index **IX_SuppID** được tạo trên cột **SuppID** của bảng **Supplier_Details**:

```
DROP INDEX IX_SuppID ON Supplier_Details
WITH (MOVE TO FGCountry)
```

- Dữ liệu trong bảng **Supplier_Details** được di chuyển đến filegroup FGCountry

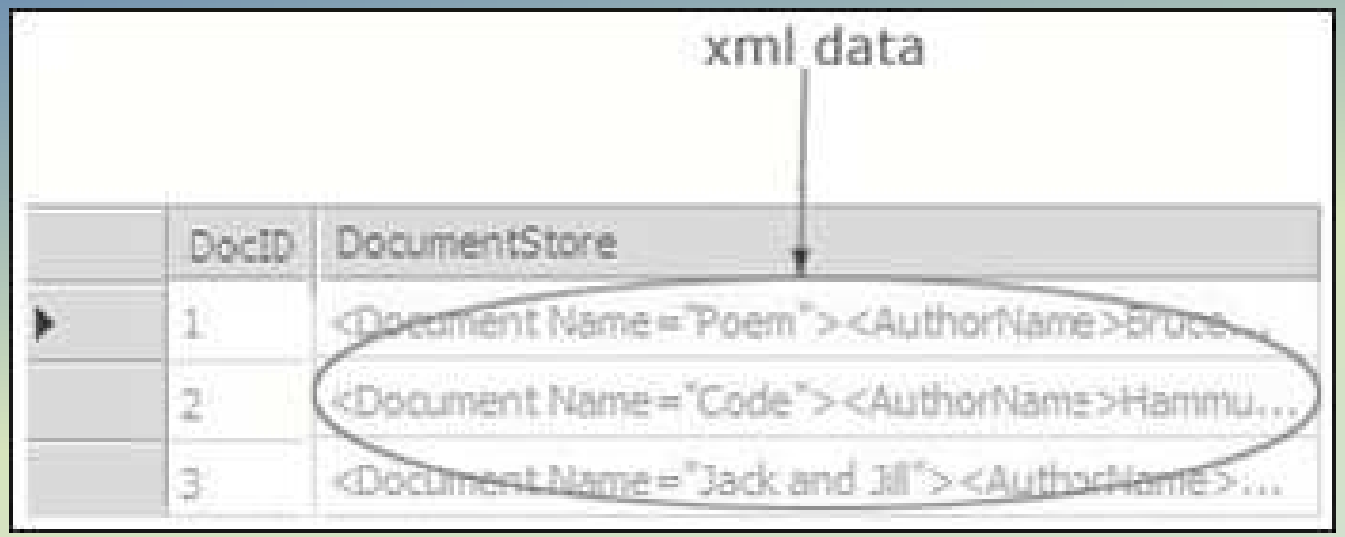
Sự khác biệt giữa Clustered Index và Nonclustered Index

Clustered Index	Nonclustered Index
Sử dụng cho các truy vấn t tập kết quả lớn	cho các truy vấn không trả ết quả lớn
Chỉ có thể tạo một clus cho một bảng	iều nonclustered index
Dữ liệu được sắp xếp t trên khóa clustered	g được sắp xếp theo thứ nonclustered
Các nút lá của một clustered có chứa các trang dữ liệu (data page).	á của một nonclustered index có chứa các trang chỉ mục(index page).



XML Index

- Kiểu dữ liệu XML được sử dụng để lưu trữ các tài liệu và các đoạn XML. Một đoạn XML là thể hiện của một XML thiếu phần tử mức đỉnh.



- Do các cột XML có kích thước lớn, truy vấn tìm kiếm dữ liệu trong các cột này có thể sẽ chậm. Bạn có thể tăng tốc độ truy vấn trên các cột này bằng việc tạo một XML index cho mỗi cột.
- Một XML index có thể là clustered index hoặc nonclustered index. Mỗi bảng có thể có tới 249 index.

Các loại XML Index 1-5

- Các index XML có thể được tạo ra trên một bảng chỉ khi có một clustered index dựa trên khóa chính của bảng.
- Khóa chính này không thể vượt quá 15 cột
- Các loại XML index khác nhau
  - XML Index sơ cấp (primary XML Index).
  - XML index thứ cấp (secondary XML Index)
  - XML index có lựa chọn (selective XML Index)

Các loại XML Index 2-5

- **XML Index sơ cấp** (primary XML Index):
  - Quá trình thực hiện các truy vấn bên trong cột XML đôi khi có thể chậm.
  - Một XML index sơ cấp được tạo ra trên mỗi cột XML để làm tăng tốc độ các truy vấn này.
  - Nó là một index đặc biệt mà cắt nhỏ(shreds) dữ liệu XML để lưu trữ thông tin.
  - Cú pháp được sử dụng để tạo ra một chỉ mục XML chính như sau:

```
CREATE PRIMARY XML INDEX index_name ON
<table_name> (column_name)
```

Các loại XML Index 3-5

- Đoạn code sau đây minh họa tạo một XML index sơ cấp trên cột **CatalogDescription** trong bảng **Production.ProductModel**

```
USE AdventureWorks2012;
CREATE PRIMARY XML INDEX
PXML_ProductModel_CatalogDescription
ON Production.ProductModel (CatalogDescription);
GO
```

Các loại XML Index 4-5

- **XML Index thứ cấp** (secondary XML Index):
  - XML index thứ cấp là các index XML chuyên dụng giúp với các truy vấn XML cụ thể.
  - Các tính năng của XML index thứ cấp như sau
    - Tìm kiếm các giá trị ở bất kỳ vị trí nào trong tài liệu XML.
    - Lấy các thuộc tính của đối tượng cụ thể từ bên trong tài liệu XML.
  - **XML index** thứ cấp chỉ có thể tạo trên các cột đã có XML index sơ cấp.

Các loại XML Index 5-5

- Đoạn code sau minh họa cách tạo XML index trên cột của **CatalogDescription** bảng **Production.ProductModel**

```
USE AdventureWorks2012;
CREATE XML INDEX
IXML_ProductModel_CatalogDescription_Path
ON Production.ProductModel (CatalogDescription)
USING XML INDEX
PXML_ProductModel_CatalogDescription FOR PATH ;
GO
```

Các loại XML Index 6-5

- XML index có tuyển chọn (**Selective XML Indexes (SXI)**)
  - Đây là kiểu XML index mới được giới thiệu trong SQL Server 2012.
  - Các tính năng của index mới này là để cải thiện hiệu suất truy vấn trên các dữ liệu được lưu trữ như XML trong SQL Server, cho phép việc lập index trên khối lượng công việc dữ liệu XML lớn nhanh hơn, và cải thiện khả năng mở rộng bằng cách giảm chi phí lưu trữ của index.
  - Cú pháp:

```
CREATE SELECTIVE XML INDEX index_name ON
<table_name> (column_name)
```

Các loại XML Index 7-5

- Dưới đây là một tài liệu XML trong bảng có khoảng 500.000 dòng

```
<book>
  <created>2004-03-01</created>
  <authors>Various</authors>
  <subjects>
     <subject>English wit and humor Periodicals</subject>
     <subject>AP</subject>
  </subjects>
  <title>Punch, or the London Charivari, Volume 156, April 2,
  1919</title>
  <id>etext11617</id>
</book>
```

Các loại XML Index 8-5

- Đoạn code sau minh họa cách tạo XML index trên cột của **BookDetails** bảng **BooksBilling**

```
USE CUST_DB
CREATE SELECTIVE XML INDEX SXI_index
ON BooksBilling(BookDetails)
FOR
(
pathTitle = '/book/title/text()' AS XQUERY
'xs:string',
pathAuthors = '/book/authors' AS XQUERY 'node()',
pathId = '/book/id' AS SQL NVARCHAR(100)
)
GO
```

SQL Server 2012

Chỉnh sửa XML index

- Một XML index sơ cấp hay thứ cấp có thể được chỉnh sửa bằng câu lệnh `ALTER INDEX`
- Cú pháp chỉnh sửa XML index

```
ALTER INDEX <tên_xml_index> ON <tên_bảng> REBUILD
```

- Đoạn code dưới đây dựng lại XML index sơ cấp `PXML_DocumentStore` được tạo trên bảng `XMLDocument`

```
ALTER INDEX PXML_DocumentStore ON XMLDocument REBUILD
```

Xóa XML index

- Cú pháp để xóa một XML index bằng câu lệnh DROP INDEX

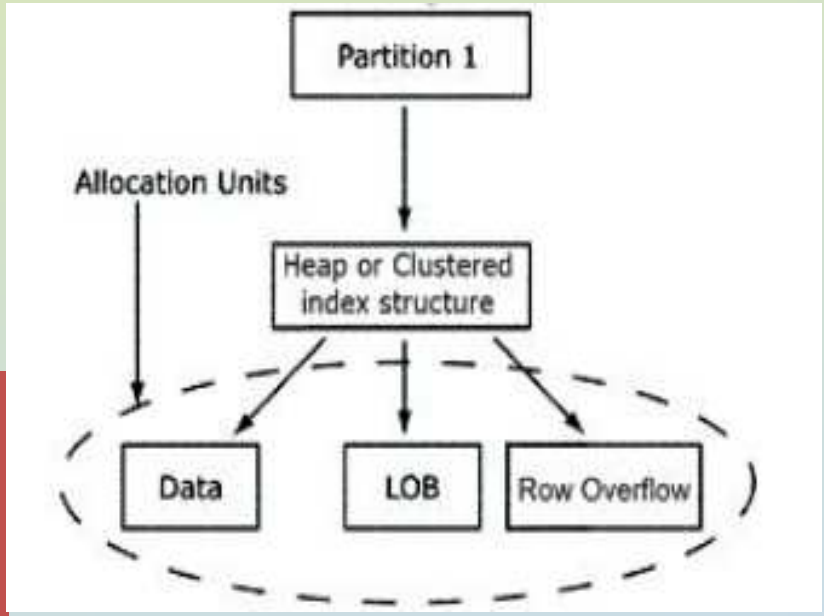
```
DROP INDEX <xml_index_name> ON <table_name>
```

- Đoạn code sau xóa bỏ một XML index sơ cấp `PXML_DocumentStore` được tạo trên bảng `XMLDocument`

```
DROP INDEX PXML_DocumentStore ON XMLDocument
```

Đơn vị cấp phát (Allocation Unit)

- Một cấu trúc clustered index hay heap có chứa các trang dữ liệu trong một hoặc nhiều đơn vị cấp phát.
- Một đơn vị cấp phát là một tập hợp các trang được sử dụng để quản lý dữ liệu dựa trên kiểu trang của chúng.
- Sau đây là các kiểu đơn vị cấp phát được sử dụng để quản lý dữ liệu trong các bảng và index.
  - IN_ROW_DATA.
  - LOB_DATA.
  - ROW_OVERFLOW_DATA.



Được dùng để quản lý dữ liệu hoặc các dòng chỉ mục mà có chứa tất cả các loại dữ liệu ngoại trừ đối tượng dữ liệu lớn (LOB).

SQL Server 2012

Phân vùng

- Phân vùng một bảng hay index là chia dữ liệu thành nhóm các dòng nhỏ hơn.
- Thực hiện thao tác quản lý trên các nhóm dòng sẽ hiệu quả hơn, vì chỉ thực hiện trên một khối nhỏ dữ liệu thay vì trên toàn bộ bảng hay index.
- Mặc định một index hoặc một bảng chỉ có một phân vùng có chứa toàn bộ dữ liệu hoặc các trang index
- Khi một bảng hoặc index sử dụng nhiều phân vùng, dữ liệu được phân chia theo chiều ngang thành các nhóm dòng như thể hiện trong hình sau đây:

Phân vùng

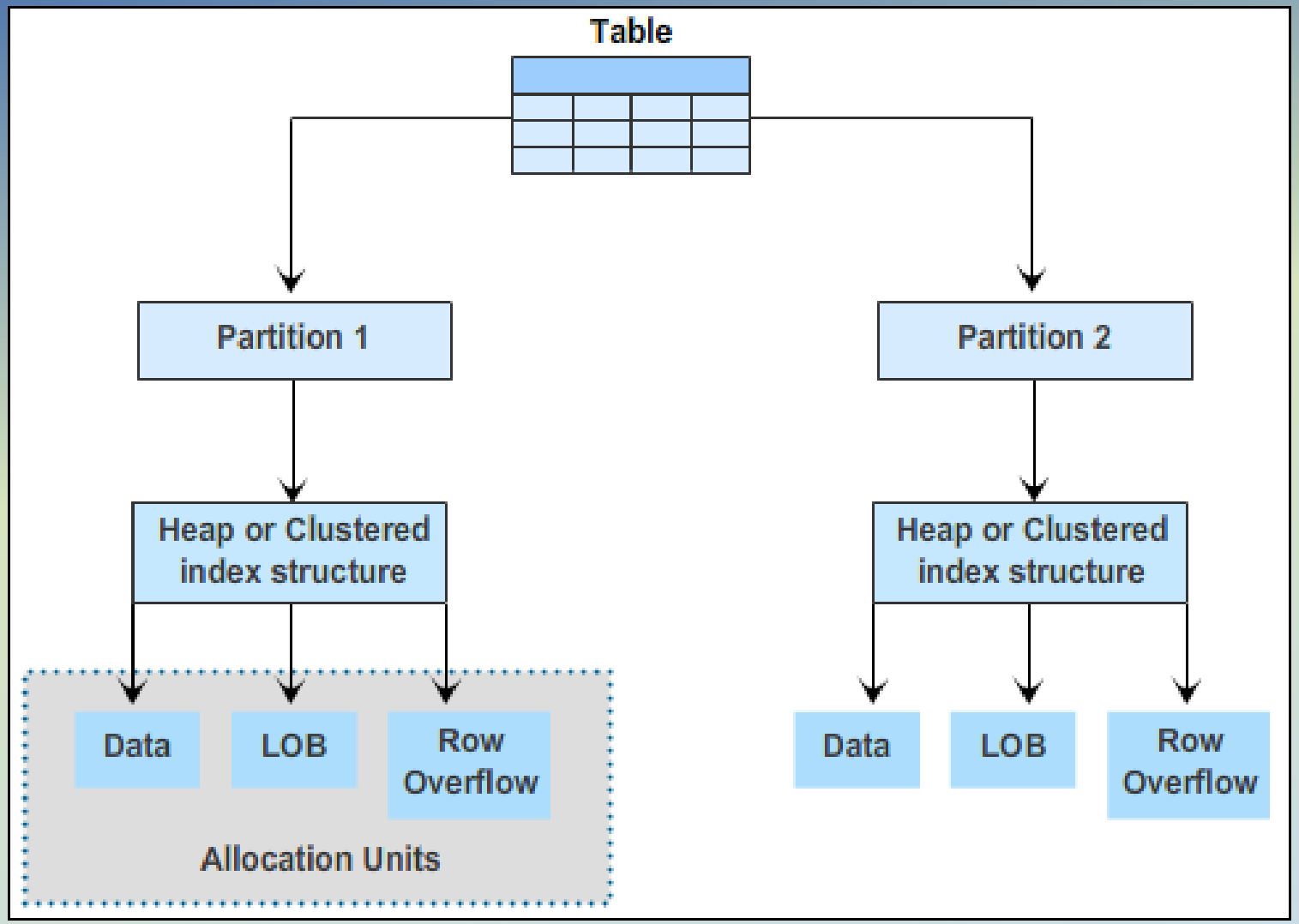


Table
Partition 1
Partition 2
Heap or Clustered index structure
Heap or Clustered index structure
Data
LOB
Row Overflow
Data
LOB
Row Overflow
Allocation Units

Phân vùng index (partitioned index)

- Các bước tạo phân vùng index
  - Bước 1: Tạo một hàm phân vùng (Partition Function)

```
CREATE PARTITION FUNCTION Tên_hàm_phân_vùng(kiểudl_của_tham_số)
AS RANGE [LEFT | RIGHT] FOR VALUES (Giá_trị_1, …,Giá_trị_n)
```

  - Bước 2: Tạo một lược đồ phân vùng (Partition Scheme)

```
CREATE PARTITION SCHEME Tên_lược_đồ_phân_vùng
AS PARTITION Tên_hàm_phân_vùng
ALL TO (tên_file_group | [PRIMARY] , …)
```

  - Bước 3: Tạo bảng mới có phân vùng hoặc index phân vùng

```
CREATE TABLE Tên_bảng (tên_cột kiểudl, …)
ON Tên_lược_đồ_phân_vùng (tên_cột_phân_vùng)
```

Phân vùng index (partitioned index)

- Ví dụ

```
CREATE PARTITION FUNCTION PF_Employee_Details (int)
AS RANGE LEFT FOR VALUES (5,10)
```

  - Tạo lược đồ phân vùng *PF_Employee_Details*

```
CREATE PARTITION SCHEME PS_Employee_Details
AS PARTITION PF_Employee_Details
ALL TO ([PRIMARY])
```

  - Tạo bảng Employee_Details được phân vùng với *PF_Employee_Details*

```
CREATE TABLE Employee_Details
( EmpID int NOT NULL, FirstName varchar(30) NOT NULL,
  LastName varchar(30) NOT NULL, DateofBirth datetime
NOT NULL,
  Gender varchar(6) NOT NULL, City varchar(40) NOT
NULL,
) ON  PS_Employee_Details (EmpID)
```

Phân vùng index(partitioned index)

- Cú pháp tạo index phân vùng

```
CREATE [UNIQUE | CLUSTERED | NONCLUSTERED] INDEX
Tên_index
ON Tên_bảng(tên_cột)
[ON Tên_lược_đồ_phân_vùng(Tên_cột_phân_vùng)]
```

- Ví dụ

```
CREATE NONCLUSTERED INDEX Ix_City
ON Employee_Details(City)
ON PS_Employee_Details(EmpId)
```

SQL Server 2012

"sys.partitions" view

- "sys.partitions" view là một view hệ thống chứa toàn bộ thông tin về các partition của tất cả các bảng và index khác nhau trong csdl.
- Một số index đặc biệt như Fulltext, Spatial, và XML không có trong view này.
- Bảng mô tả các cột khác nhau của view "sys.partitions"

Tên cột	Kiểu dữ liệu	Mô tả
partition_id	bigint	Chứa ID của phân vùng và là duy nhất trong csdl
object_id	int	Chứa ID của đối tượng mà phân vùng này thuộc về
index_id	int	Có chứa ID của index mà phân đoạn này thuộc về.
partition_number	int	Có chứa số lượng phân vùng bên trong index hoặc heap
hobt_id	bigint	Chứa ID của heap dữ liệu hoặc B-tree mà có chứa các dòng cho phân vùng này.
rows	bigint	Cho biết gần đúng số dòng có trong phân vùng

SQL Server 2012

"sys.partitions" view

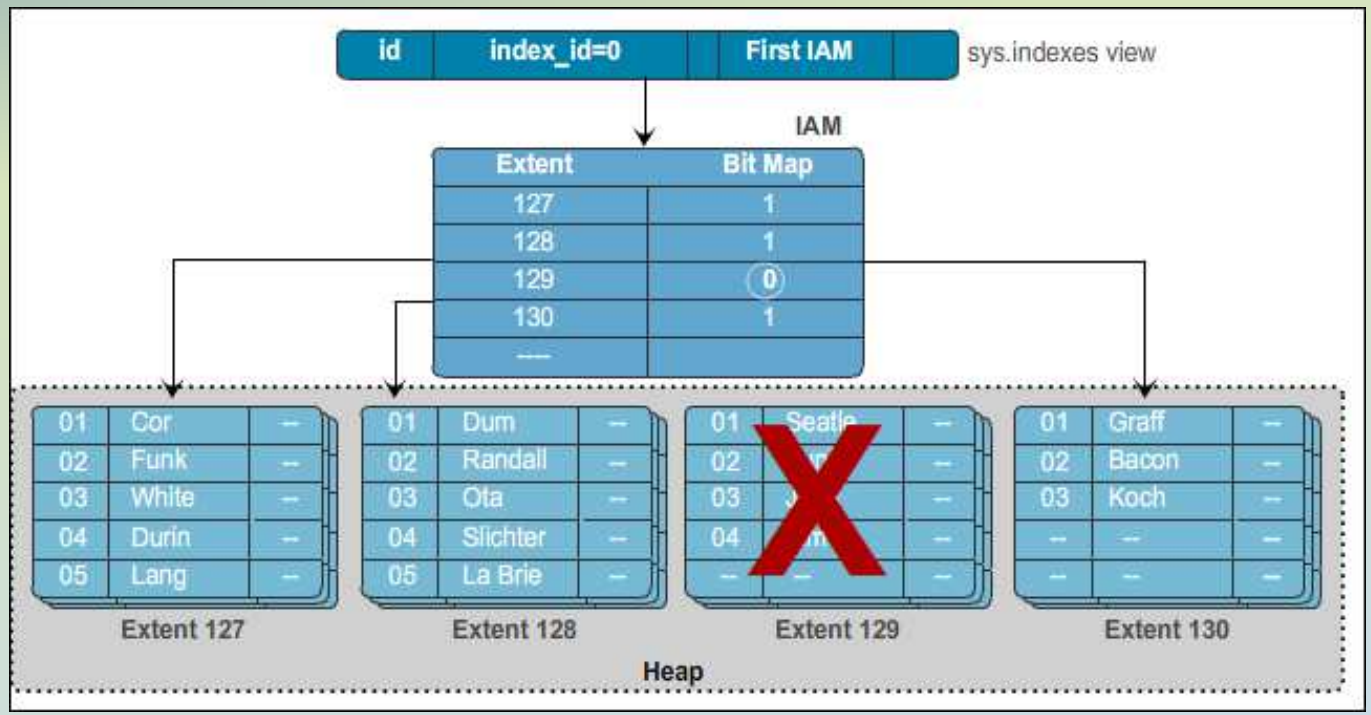
- Giá trị của cột index_id là duy nhất bên trong bảng mà phân vùng được tạo trong đó:
  - Giá trị `index_id` cho một heap là 0.
  - Giá trị `index_id` cho một cluestered index là 1.
  - Giá trị `index_id` cho một nonclustered index là lớn hơn 1.
  - Giá trị `index_id` cho các đối tượng lớn(LOB) là lớn hơn 250.

	partition_id	object_id	index_id	partition_number	hobt_id	rows
1	196608	3	1	1	196608	1779
2	327680	5	1	1	327680	332
3	458752	7	1	1	458752	379
4	524288	8	0	1	524288	2
5	281474977103872	6	1	1	281474977103872	0
6	281474977300480	9	1	1	281474977300480	0
7	281474977824768	17	1	1	281474977824768	0
8	281474977890304	18	1	1	281474977890304	0
9	281474977955840	19	1	1	281474977955840	0
10	281474978021376	20	1	1	281474978021376	2

SQL Server 2012

Tìm kiếm dòng 1-2

- Khi không có index nào được tạo trên bảng, SQL Server sử dụng các view danh mục(catalog views) để tìm dòng.
- Nó sử dụng view `sys.indexes` để tìm trang IAM.
- Trang IAM này có chứa danh sách tất cả các trang của bảng được chỉ ra, thông qua đó SQL Servercó thể đọc tất cả các trang dữ liệu.
- Khi view `sys.indexes` được sử dụng , trình tối ưu hóa truy vấn(query optimizer) kiểm tra tất cả các dòng trong bảng và chỉ rút ra các dòng được tham chiếu bởi truy vấn như hình minh họa sau:



- This scan generates many I/O operations and utilizes many resources.

Tìm kiếm dòng 2-2

- Đoạn code sau minh họa tạo bảng **Employee_Details** không có index:

```
USE CUST_DB
CREATE TABLE Employee_Details
(
      EmpID int not null,
      FirstName varchar(20) not null,
      LastName varchar(20) not null,
      DateofBirth datetime not null,
      Gender varchar(6) not null,
      City varchar(30) not null,
)
GO
```

- Giả sử đã chèn nhiều bản ghi vào bảng **Employee_Details**.
- Câu lệnh SELECT được sử dụng để tìm kiếm bản ghi có **FirstName** là John.
- Do không có index gắn với cột **FirstName**, SQL Server sẽ thực hiện quét toàn bộ bảng (full table scan).

SQL Server 2012

Tìm kiếm dòng với Nonclustered Index 1-3

Một nonclustered index được ví tương tự như mục lục quyển sách (book index); dữ liệu và mục lục(index) được lưu ở hai nơi khác nhau.

Các con trỏ (pointers) trong mức lá (leaf level) của index trỏ tới vị trí lưu trữ dữ liệu trong bảng phía dưới.

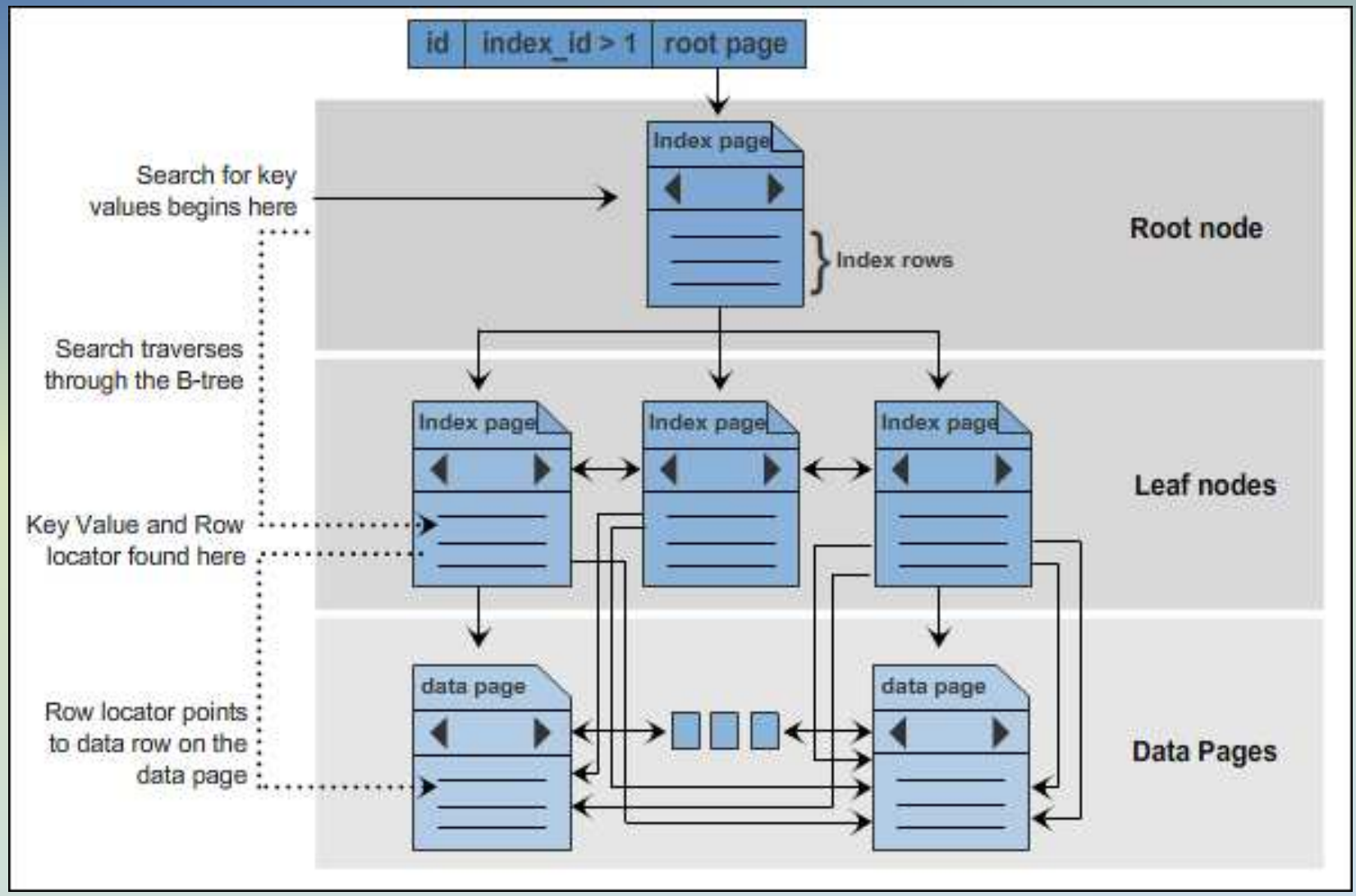
Nonclustered index được sử dụng để tìm kiếm cho các truy vấn so khớp chính xác.

Điều này là bởi vì index có chứa các mục mô tả vị trí chính xác của dữ liệu trong bảng.

Để việc tìm kiếm dòng sử dụng các nonclustered index, một câu lệnh SELECT được sử dụng với cột nonclustered index được chỉ ra trong mệnh đề WHERE.

Tìm kiếm dòng với Nonclustered Index 2-3

- Hình minh họa sau cho thấy quá trình tìm kiếm dòng với nonclustered index:



Tìm kiếm dòng với Nonclustered Index 3-3

- Đoạn code dưới đây minh họa tạo nonclustered index **IX_EmployeeCity** trên cột **City** của bảng **Employee_Details**:

```
USE CUST_DB
CREATE NONCLUSTERED INDEX IX_EmployeeCity ON Employee_Details(City);
GO
```

- Giả sử đã chèn nhiều bản ghi vào bảng **Employee_Details**
- Câu lệnh SELECT được sử dụng để tìm kiếm bản ghi của nhân viên đến từ thành phố Boston như trong đoạn code sau đây:

```
USE CUST_DB
SELECT EmpID,FirstName,LastName,City FROM Employee_Details WHERE
City='Boston'
GO
```

- Do có một nonclustered index gắn với cột **City**, SQL Server sẽ sử dụng index **IX_EmployeeCity** để rút các bản ghi như hình sau đây:

Results | Messages

	EmpID	FirstName	LastName	City
1	101	Andrew	Waller	Boston
2	103	Sophia	broderich	Boston

SQL Server 2012

Tìm kiếm dòng với Clustered Index 1-3

Clustered indexe lưu trữ các dòng dữ liệu trong bảng dựa trên các giá trị khóa của chúng.

Dữ liệu được lưu trữ theo thứ tự sắp xếp.

Nếu giá trị khóa clustered key là nhỏ, thêm số lượng dòng index có thể được đặt trên trang index.

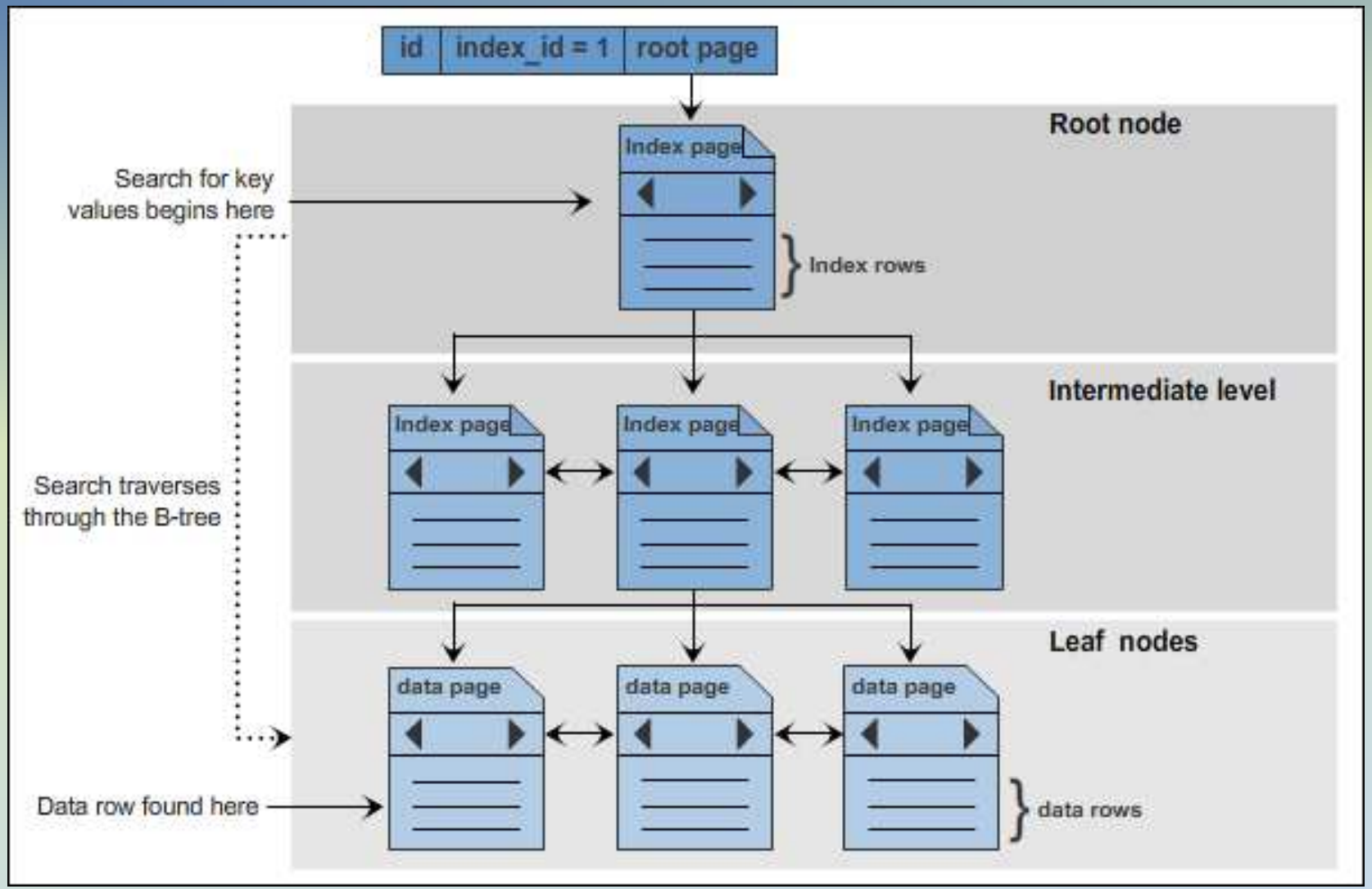
Điều này làm giảm bớt số lượng các mức trong index B-Tree mà phải duyệt để đạt được các dòng dữ liệu sản xuất kết quả truy vấn nhanh hơn. Điều này giảm thiểu nhập / xuất.

Để sử dụng clustered indexe cho việc tìm dòng, câu lệnh `SELECT` được sử dụng với cột clustered index được chỉ ra trong mệnh đề `WHERE`.

SQL Server 2012

Tìm kiếm dòng với Clustered Index 2-3

- Hình minh họa sau cho thấy quá trình tìm kiếm dòng với clustered index:



Tìm kiếm dòng với Clustered Index 3-3

- Đoạn code sau minh họa tạo một clustered index có tên **IX_EmployeeID** trên cột **EmpID** của bảng **Employee_Details**:

```
USE CUST_DB
CREATE UNIQUE CLUSTERED INDEX IX_EmployeeID ON
Employee_Details(EmpID);
GO
```

- Giả sử đã chèn nhiều bản ghi vào bảng **Employee_Details**.
- Câu lệnh SELECT được sử dụng để tìm kiếm bản ghi của nhân viên có **EmpID** trong khoảng 102 và 105 được trình bay trong ví dụ sau:

```
USE CUST_DB
SELECT EmpID,FirstName,LastName,City FROM Employee_Details WHERE EmpID
>= 102 AND EmpID <= 105;
GO
```

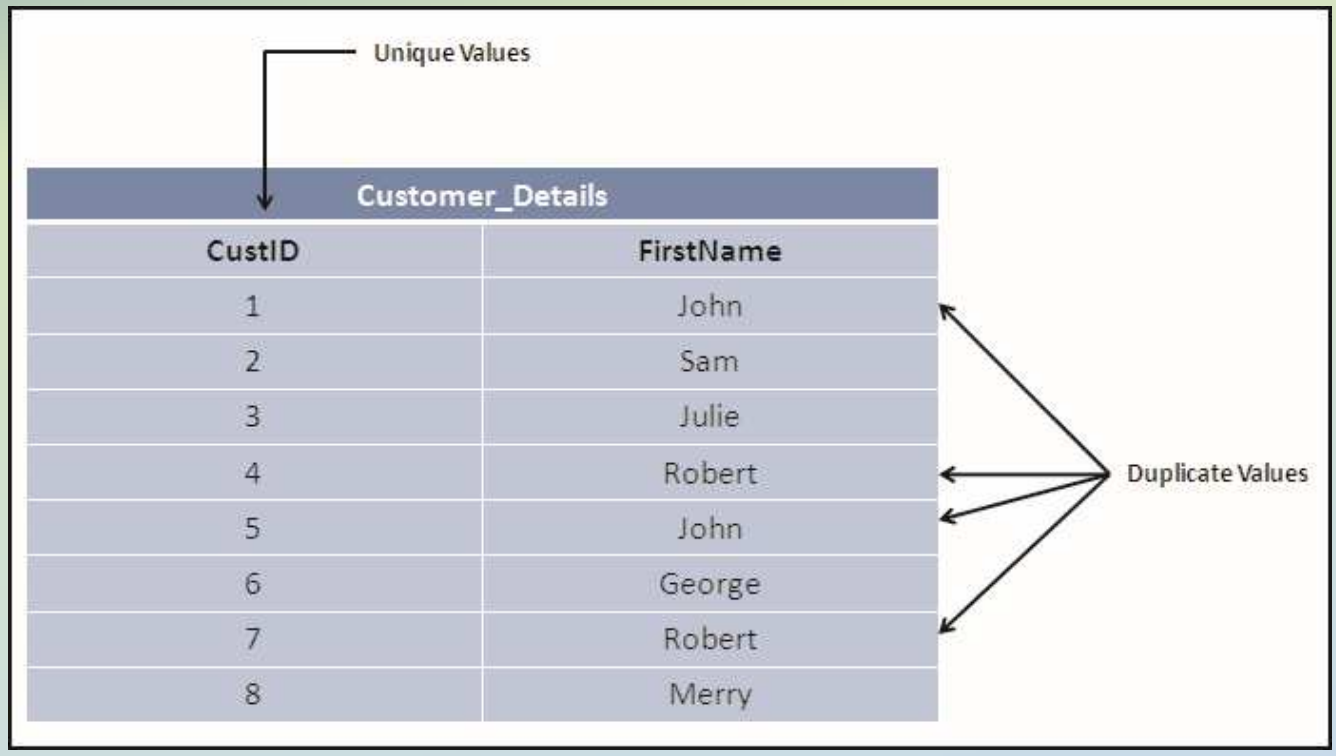
- Do có một clustered index gắn trên cột **EmpID**, SQL Server sẽ sử dụng index **IX_EmployeeID** để lấy các bản ghi như trong hình sau:

Results | Messages

	EmpID	FirstName	LastName	City
1	102	AJ	Stiles	Liverpool
2	103	Sophia	broderich	Boston
3	104	Shawn	roderichs	Texas

Tạo Unique Index 1-2

- Có thể tạo một unique index trên cột không có bất kỳ giá trị trùng nhau.
- Khi unique index được tạo cho cột, cột đó sẽ không chấp nhận các giá trị trùng nhau.
- Do vậy, unique index chỉ nên được tạo trên các cột mà ở đó tính duy nhất của giá trị là đặc điểm làm khóa. Một unique index đảm bảo tính toàn vẹn thực thể trong một bảng.
- Nếu định nghĩa một bảng có khóa chính hoặc một cột với ràng buộc UNIQUE, SQL Server tự động tạo ra một bảng khi bạn thực thi câu lệnh CREATE TABLE như thể hiện trong hình sau đây:



Tạo Unique Index 2-2

- Có thể tạo unique index bằng câu lệnh `CREATE UNIQUE INDEX` hoặc bằng SSMS.
- Cú pháp tạo unique index như sau:

```
CREATE UNIQUE INDEX <index_name> ON
<table_name>(<column_name>)
```

Trong đó,

column_name: chỉ ra tên của cột mà index được tạo trên đó.

UNIQUE: chỉ ra rằng không cho phép có gía trị trùng nhau trong cột.

- Đoạn code dưới đây monh họa tạo unique index trên cột **CustID** trong bảng **Customer_Details**:

```
CREATE UNIQUE INDEX IX_CustID ON
Customer_Details(CustID)
```

Tạo cột tính toán (Computed Column) 1-2

Cột tính toán (computed column) là cột ảo trong một bảng mà gía trị của nó được tính toán lúc thực thi.

Các giá trị trong cột không được sắp xếp trong bảng nhưng được tính toán dựa dựa trên biểu thức được định nghĩa cho cột.

Biểu thức gồm có cột non-computed kết hợp với hằng, hàm hoặc biến bằng các toán tử toán học hoặc toán tử logic.

- Có thể sử dụng câu lệnh CREATE TABLE hoặc ALTER TABLE để tạo cột tính toán như trong hình minh họa sau:

Length	Breadth	Area
34	10	340
20	20	400
33.4	12	400.8
12	7	84

Computed column (A=L*B)

Tạo cột tính toán (Computed Column) 2-2

- Cú pháp dùng để: tạo cột tính toán như sau:

```
CREATE TABLE <table_name> ([<column_name> AS
<computed_column_expression>])
```

trong đó,

table_name: chỉ ra tên của bảng.

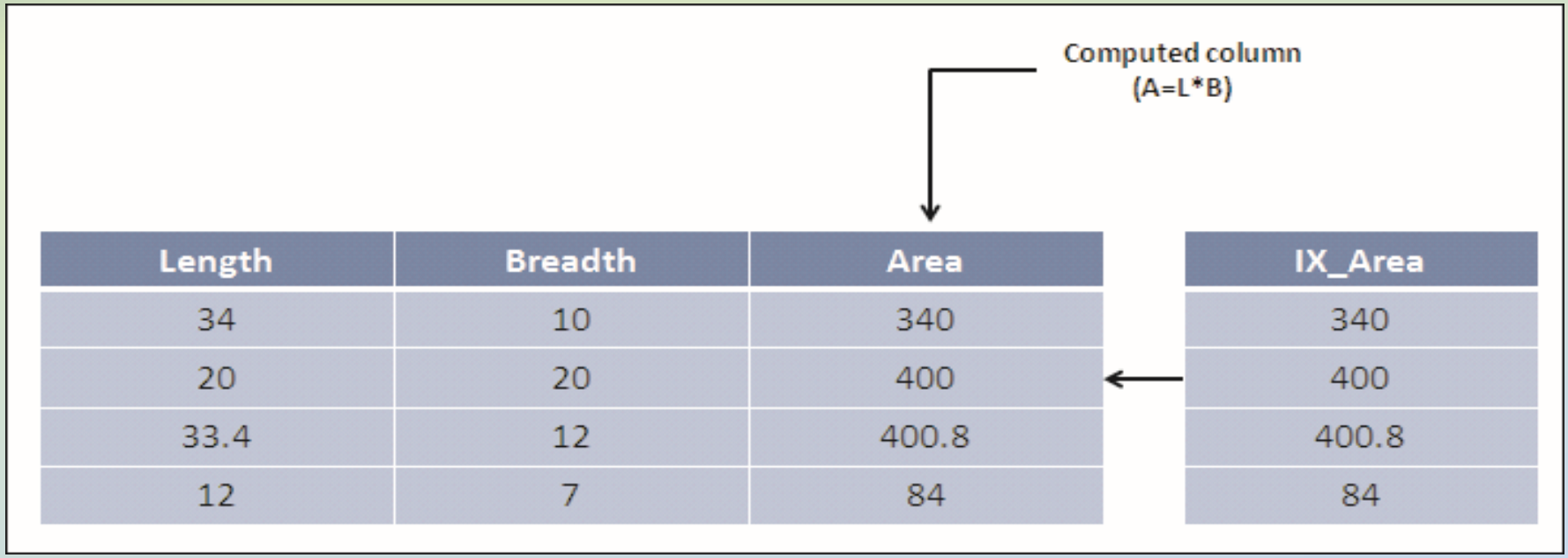
column_name AS computed_column_expression: chỉ ra tên cho cột tính toán cùng với biểu thức xác định giá trị trong cột.

- Đoạn code sau minh họa tạo cột tính toán Area, các giá trị của nó được tính toán từ các giá trị nhập vào các cột **Length** và **Breadth** :

```
USE SampleDB
CREATE TABLE Calc_Area(Length int, Breadth int,
Area AS Length*Breadth)
GO
```

Tạo Index trên cột tính toán 1-2

- Có thể tạo một index trên cột tính toán nếu cột được đánh dấu là PERSISTED.
- Điều này đảm rằng Database Engine lưu trữ cá giá trị tính toán trong bảng.
- Những giá trị này được cập nhật khi bất kỳ cột nào mà cột tính toán phụ thuộc vào được cập nhật.
- Bộ máy csdl (database engine) sử dụng giá trị persisted khi nó tạo một index trên cột.
- Một index được tạo trên cột tính toán bằng câu lệnh CREATE INDEX như hình minh họa dưới đây:



Length	Breadth	Area
34	10	340
20	20	400
33.4	12	400.8
12	7	84

IX_Area
340
400
400.8
84

Tạo Index trên cột tính toán 2-2

- Cú pháp tạo index trên cột tính toán:

```
CREATE INDEX <index_name> ON <table_name>
(<computed_column_name>)
```

Trong đó,

computed_column_name: chỉ ra tên cột tính toán.

- Đoạn code sau tạo index có tên **IX_Area** trên cột tính toán **Area**:

```
USE SampleDB
CREATE INDEX IX_Area ON Calc_Area(Area);
GO
```

SQL Server 2012

Cursors 1-7

- Một đối tượng cơ sở dữ liệu được sử dụng để duyệt lấy từng dòng dữ liệu từ tập kết quả (resultset), được gọi là cursor(con trỏ)..
- Cursor được sử dụng khi các bản ghi trong một bảng của cơ sở dữ liệu cần được cập nhật từng dòng ở một thời điểm.
- Các loại cursor:

Static Cursors

Không thể thực hiện cập nhật hay xóa dữ liệu trong cursor tĩnh (static cursor) .
Khi static cursor được tạo , nó lưu trữ kết quả truy vấn trong cached(trong temdb). Do đó nếu những người dùng khác có các thay đổi dữ liệu ở bảng gốc(base table) thì các thay đổi đó sẽ không được cập nhật tự động đến dữ liệu trong cursor. Đây là cursor chậm nhất trong tất cả cursor.
Cursor này có thể di chuyển/cuộn theo cả hai hướng quay lui(backward) và tiến tới(forward).

Dynamic Cursors

- Dùng chỉ định dữ liệu bên trong cursor là động. Khi đó việc cập nhật dữ liệu trong bảng cơ sở (base table) bởi những người dùng khác sẽ được cập nhật tự động tới dữ liệu trong cursor có kiểu là DYNAMIC
- Cho phép thực hiện các thao tác thêm, cập nhật, xóa khi cursor được mở.

Forward Only Cursors

- Là loại cursors mặc định, cũng hỗ trợ việc cập nhật và xóa dữ liệu.
- Đây là loại cursor nhanh nhất, thế nhưng nó không hỗ trợ cuộn lùi lại.
- Có ba kiểu cursor chỉ di chuyển tiến là `FORWARD_ONLY KEYSET`, `FORWARD_ONLY STATIC`, và `FAST_FORWARD`.

SQL Server 2012

Cursors 2-7

Keyset Driven Cursors

- Các cursor này tạo một tập bộ định danh duy nhất như là các khóa trong một keyset được sử dụng để điều khiển cursor.
- có hoạt động gần giống với kiểu DYNAMIC, các thay đổi dữ liệu trên cáccột không là khóa chính trong bảng cơ sở bởi những người dùng khác sẽ được cập nhật trong dữ liệu cursor.
- Tuy nhiên đối với các mẫu tin vừa thêm mới hoặc các mẫu tin đã bị hủy
- bỏ bởi những người dùng khác sẽ không được hiển thị trong dữ liệu cursor có kiểu là KEYSET.

➢ Cú pháp được sử dụng để khai báo một cursor như sau:

```
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
FOR select_statement
[ FOR UPDATE [ OF column_name [ ,...n ] ] ]
[;]
```

Trong đó,

cursor_name: là tên của cursor được định nghĩa, cursor_name phải tuân theo quy tắc của bộ định danh.

SQL Server 2012

Cursors 3-7

LOCAL: chỉ ra rằng cursor được sử dụng cục bộ trong một lô(batch), stored procedure, hoặc trigger mà cusor được tạo trong đó.

GLOBAL: chỉ ra rằng cursor có thể được sử dụng toàn cục trong một kết nối.

FORWARD_ONLY: chỉ ra rằng cursor chỉ có thẻ cuộn theo chiều tiến từ dòng đầu tiên đến dòng cuối cùng.

STATIC: định nghĩa một cursor, tạo một bản sao tạm thời của dữ liệu được sử dụng bởi cursor.

KEYSET: chỉ ra rằng tính liên đới và thứ tự các dòng trong cursor được cố định khi cursor được mở.

DYNAMIC: định nghĩa cursor mà phản ánh tất cả sự thay đổi dữ liệu tới các dòng trong tập kết quả của nó khi bạn cuộn quanh cursor.

FAST_FORWARD: chỉ ra một cursor FORWARD_ONLY, READ_ONLY cho phép tối ưu hiệu suất.

READ_ONLY: ngăn mọi thao tác thực hiện cập nhật thông qua cursor này.

SCROLL_LOCKS: chỉ ra việc cập nhật, xóa được đánh dấu vị trí thông qua cursor được đảm thành công.

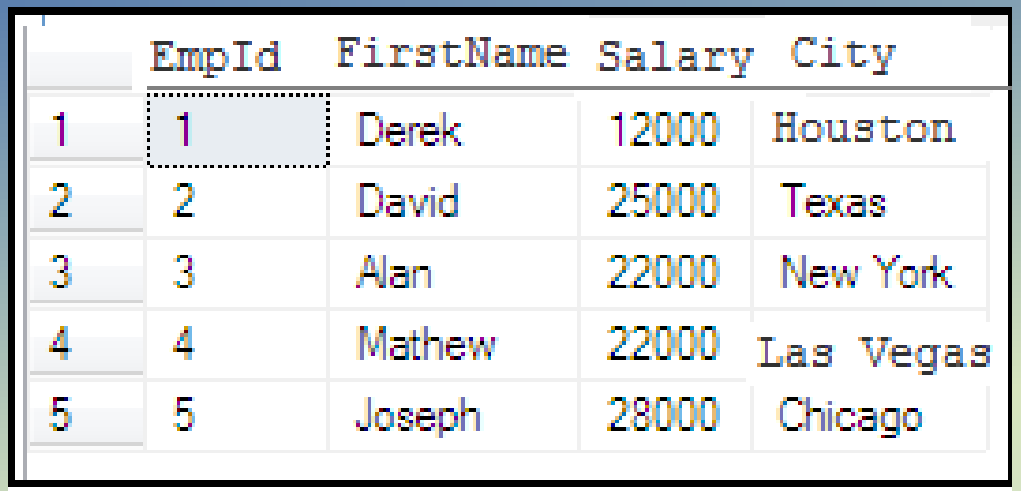
Cursors 4-7

- Đoạn code sau tạo bảng **Employee** trong csdl **SampleDB** :

```
USE SampleDB
CREATE TABLE Employee
(
EmpID int PRIMARY KEY,
EmpName varchar (50) NOT NULL,
Salary int NOT NULL,
Address varchar (200) NOT NULL,
)
GO
INSERT INTO Employee(EmpID,EmpName,Salary,Address)
VALUES(1,'Derek',12000,'Housten')
INSERT INTO Employee(EmpID,EmpName,Salary,Address)
VALUES(2,'David',25000,'Texas')
INSERT INTO Employee(EmpID,EmpName,Salary,Address)
VALUES(3,'Alan',22000,'New York')
INSERT INTO Employee(EmpID,EmpName,Salary,Address)
VALUES(4,'Mathew',22000,'las Vegas')
INSERT INTO Employee(EmpID,EmpName,Salary,Address)
VALUES(5,'Joseph',28000,'Chicago')
GO
SELECT * FROM Employee
```

Cursors 5-7

- Hình dưới đây cho thấy kết quả của đoạn code:



	EmpId	FirstName	Salary	City
1	1	Derek	12000	Houston
2	2	David	25000	Texas
3	3	Alan	22000	New York
4	4	Mathew	22000	Las Vegas
5	5	Joseph	28000	Chicago

- Đoạn code sau minh họa cách khai báo một cursor trên bảng **Employee** :

```
USE SampleDB
SET NOCOUNT ON
DECLARE @Id int
DECLARE @name varchar(50)
DECLARE @salary int
/* A cursor is declared by defining the SQL statement that returns a
resultset.*/
DECLARE cur_emp CURSOR
```

Cursors 6-7

```
STATIC FOR
SELECT EmpID,EmpName,Salary from Employee
/*A Cursor is opened and populated by executing the SQL statement defined by the
cursor.*/
OPEN cur_emp
IF @@CURSOR_ROWS > 0
BEGIN
/*Rows are fetched from the cursor one by one or in a block to do data
manipulation*/
FETCH NEXT FROM cur_emp INTO @Id,@name,@salary
WHILE @@Fetch_status = 0
BEGIN
        PRINT 'ID : '+ convert(varchar(20),@Id)+', Name : '+@name+ ', Salary :
        '+convert(varchar(20),@salary)
        FETCH NEXT FROM cur_emp INTO @Id,@name,@salary
END
END
--close the cursor explicitly
CLOSE cur_emp
/*Delete the cursor definition and release all the system resources associated
with the cursor*/
DEALLOCATE cur_emp
SET NOCOUNT OFF
```

Cursors 7-7

- Trong đoạn code, mỗi lần lấy được một dòng chứa các chi tiết của một Employee.
- Đầu tiên, cursor được khai báo bằng việc định nghĩa câu lệnh SQL trả về một tập kết quả.
- Sau đó, nó được mở và được thao tác bằng việc thực thi câu lệnh SQL được định nghĩa bởi một cursor.
- Các dòng trong một khối từ cursor sau đó được duyệt từng dòng để thực hiện thao tác dữ liệu.
- Cuối cùng, cursor được đóng, cursor được xóa và giải phóng toàn bộ tài nguyên hệ thống đã cấp cho nó.
- Hình dưới đây cho thấy kết quả của đoạn code: