

Session - 11

Indexes

Welcome to the Session, **Indexes**.

This session explains indexes and their performance. It also explains different types of indexes. Finally, the session identifies and describes the procedure to query performance data using indexes.

In this Session, you will learn to:

- Define and explain indexes
- Describe the performance of indexes
- Explain clustered indexes
- Explain nonclustered indexes
- Explain partitioning of data
- Explain the steps to display query performance data using indexes



11.1 Giới thiệu

SQL Server 2012 sử dụng các chỉ mục để tìm dữ liệu khi truy vấn được xử lý. Công cụ SQL Server sử dụng chỉ mục theo cách tương tự như học viên sử dụng chỉ mục cuốn sách. Ví dụ, xem xét rằng bạn cần phải tìm tất cả các tham khảo đến các câu lệnh INSERT trong một cuốn sách SQL. Cách tiếp cận ngay lập tức đã sử dụng sẽ quét từng trang của cuốn sách bắt đầu từ trang bắt đầu. Bạn đánh dấu mỗi lần từ INSERT được tìm thấy, cho đến tới cuối cuốn sách. Cách tiếp cận này rất tốn thời gian và công sức. Cách thứ hai là sử dụng chỉ mục ở phía sau của cuốn sách để tìm số trang cho mỗi lần xuất hiện của các câu lệnh INSERT. Cách thứ hai tạo ra các kết quả tương tự như cách đầu tiên, nhưng rất tiết kiệm thời gian.

Khi SQL Server đã không xác định bất kỳ chỉ mục để tìm kiếm, khi đó quá trình này tương tự như cách đầu tiên trong ví dụ, công cụ SQL cần phải ghé thăm mỗi hàng trong bảng. Trong thuật ngữ cơ sở dữ liệu, hành vi này được gọi là quét bảng, hoặc chỉ là quét.

Quét bảng không phải luôn phiền hà, nhưng đôi khi không thể tránh khỏi. Tuy nhiên, khi bảng lớn lên đến hàng ngàn và hàng triệu hàng và hơn thế nữa, quét trở nên chậm hơn và tốn kém hơn. Trong các trường hợp này, đặc biệt nên sử dụng chỉ mục.

Việc tạo hoặc loại bỏ các chỉ mục khỏi lược đồ cơ sở dữ liệu sẽ không ảnh hưởng đến mã của một ứng dụng. Chỉ mục hoạt động ở phía sau với sự hỗ trợ của công cụ cơ sở dữ liệu. Hơn nữa, việc tạo ra một chỉ mục thích hợp có thể làm tăng đáng kể hiệu suất của ứng dụng.

11.1.1 Tổng quan về lưu trữ dữ liệu

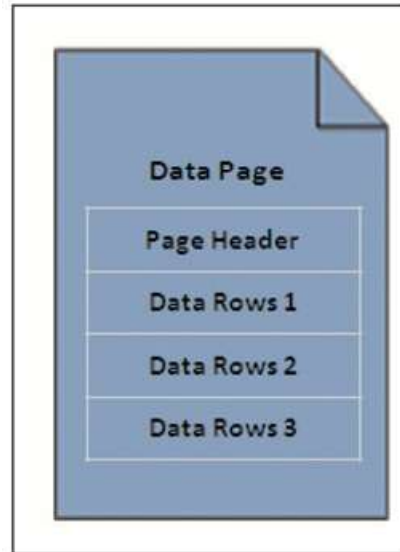
Một cuốn sách có các trang, trong đó có chứa các đoạn văn được tạo ra từ các câu. Tương tự như vậy, SQL Server 2012 lưu trữ dữ liệu trong các đơn vị lưu trữ được gọi là các trang dữ liệu. Những trang này chứa dữ liệu dưới dạng các hàng.

Trong SQL Server 2012, dung lượng của mỗi trang dữ liệu là 8 Kilo Byte (KB). Như vậy, cơ sở dữ liệu SQL Server có 128 trang dữ liệu cho mỗi Mega Byte (MB) không gian lưu trữ.

Trang bắt đầu với phần đầu đề 96 byte, lưu trữ thông tin hệ thống về trang đó. Những thông tin này bao gồm:

- Số của trang
- Loại trang
- Số lượng không gian còn trống trên trang
- ID đơn vị phân bổ của đối tượng để phân bổ cho trang đó

Hình 11.1 trình bày cấu trúc lưu trữ dữ liệu của một trang dữ liệu.



Hình 11.1: Data Storage

Ghi chú - Một trang dữ liệu là đơn vị lưu trữ dữ liệu nhỏ nhất. Đơn vị phân bố là một tập hợp các trang dữ liệu được nhóm lại với nhau dựa trên loại trang. Việc nhóm lại này được thực hiện để quản lý hiệu quả dữ liệu.

11.1.2 Các tập tin dữ liệu

Tất cả các phép tính đầu vào và đầu ra trong cơ sở dữ liệu được thực hiện ở mức trang. Điều này có nghĩa rằng công cụ cơ sở dữ liệu đọc hoặc viết các trang dữ liệu. Một bộ tám trang dữ liệu tiếp giáp được gọi là một mức độ.

SQL Server 2012 lưu trữ dữ liệu trong các tập tin được gọi là các tập tin dữ liệu. Không gian được phân bổ cho một tập tin dữ liệu được chia thành các trang dữ liệu được đánh số thứ tự. Đánh số bắt đầu từ số 0 như được trình bày trong hình 11.2.



Hình 11.2: Data Files

Có ba loại tập tin dữ liệu trong SQL Server 2012. Chúng bao gồm:

➤ **Tập tin dữ liệu chính**

Tập tin dữ liệu chính được tạo ra tự động vào thời điểm tạo ra cơ sở dữ liệu. Tập tin này có các tham khảo tới tất cả các tập tin khác trong cơ sở dữ liệu. Phần mở rộng tập tin được đề nghị cho các tập tin dữ liệu chính là .mdf.

➤ **Tập tin dữ liệu phụ**

Tập tin dữ liệu phụ là tùy chọn trong một cơ sở dữ liệu và có thể được tạo ra để phân biệt các đối tượng cơ sở dữ liệu như là các bảng, khung nhìn, và thủ tục. Phần mở rộng tập tin được đề nghị cho các tập tin dữ liệu phụ là .ndf.

➤ **Tập tin nhật ký**

Tập tin nhật ký chứa thông tin về các thay đổi được thực hiện trong cơ sở dữ liệu. Thông tin này rất hữu ích trong phục hồi dữ liệu trong các sự cố bất ngờ như mất điện đột ngột hoặc cần phải chuyển cơ sở dữ liệu đến một máy chủ khác. Có ít nhất một tập tin nhật ký cho mỗi cơ sở dữ liệu. Phần mở rộng tập tin được đề nghị cho các tập tin nhật ký là .ldf.

11.1.3 Yêu cầu đối với các Indexes

Để tạo điều kiện lấy dữ liệu nhanh chóng từ một cơ sở dữ liệu, SQL Server 2012 cung cấp tính năng lập chỉ mục. Tương tự như chỉ mục trong một cuốn sách, chỉ mục trong cơ sở dữ liệu SQL Server 2012 chứa thông tin cho phép bạn tìm kiếm dữ liệu cụ thể mà không quét qua toàn bộ bảng như được trình bày trong hình 11.3.

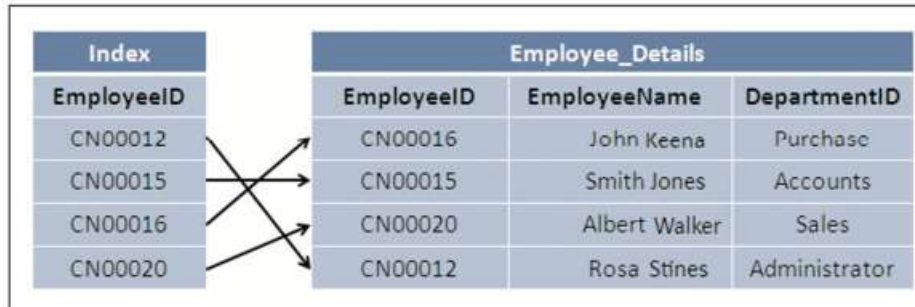
Index			
A			
Adapter	1	Border	19
Aggregate	10	Bullet	58
Analysis	13		
Average	23		
B		C	
		Consistency	20
		Connect	22
Board	17	Communication	24
Brilliant	18	Character	30

Hình 11.3: Requirement for Indexes

11.1.4 Indexes

Trong bảng, các bản ghi được lưu trữ theo thứ tự chúng được nhập vào. Lưu trữ của chúng trong cơ sở dữ liệu không được phân loại. Khi dữ liệu được lấy từ các bảng như vậy, toàn bộ bảng cần phải được quét. Điều này làm chậm quá trình truy xuất truy vấn. Để tăng tốc truy xuất truy vấn, cần phải tạo ra các chỉ mục.

Khi chỉ mục được tạo ra trên một bảng, chỉ mục tạo ra một trật tự cho các hàng dữ liệu hoặc bản ghi trong bảng như được trình bày trong hình 11.4. Điều này hỗ trợ ở vị trí nhanh hơn và truy xuất dữ liệu trong quá trình tìm kiếm.



Hình 11.4: Indexes

Ghi chú - Nhiều chỉ mục có thể được tạo ra trên một bảng

Chỉ mục sẽ được tự động tạo ra khi các ràng buộc PRIMARY KEY và UNIQUE được định nghĩa trên bảng. Chỉ mục giảm các hoạt động nhập/xuất của đĩa và tiêu thụ các tài nguyên hệ thống ít hơn.

Câu lệnh CREATE INDEX được sử dụng để tạo ra một chỉ mục. Sau đây là cú pháp cho câu lệnh này.

Cú pháp:

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

trong đó,

index_name: chỉ ra tên của chỉ mục.

table_name: chỉ ra tên của bảng.

column_name: chỉ ra tên của cột.

Code Snippet 1 tạo ra chỉ mục **IX_Country** trên cột **Country** trong bảng **Customer_Details**.

Code Snippet 1:

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

Hình 11.5 trình bày bảng có chỉ mục của **Customer_Details**.

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

Hình 11.5: Indexed Table of Customer_Details

Chỉ mục trở đến vị trí của một hàng trên một trang dữ liệu thay vì tìm kiếm suốt cả bảng.

Xem xét các sự kiện và hướng dẫn sau đây về chỉ mục:

- Chỉ mục làm tăng tốc độ các truy vấn nối các bảng hoặc thực hiện các hoạt động phân loại.
- Chỉ mục thực hiện sự duy nhất của các hàng nếu được định nghĩa khi bạn tạo chỉ mục.
- Chỉ mục được tạo ra và duy trì theo thứ tự tăng dần hoặc giảm dần.

11.1.5 Kịch bản

Trong danh bạ điện thoại, nơi một số lượng lớn dữ liệu được lưu trữ và thường xuyên được truy cập, lưu trữ dữ liệu được thực hiện theo thứ tự bảng chữ cái. Nếu dữ liệu đó đã không được phân loại, nó sẽ gần như không thể tìm kiếm một số điện thoại cụ thể.

Tương tự như vậy, trong một bảng cơ sở dữ liệu có một số lượng lớn các bản ghi được truy cập thường xuyên, dữ liệu sẽ được sắp xếp để truy xuất nhanh chóng. Khi chỉ mục được tạo ra trên bảng, chỉ mục phân loại một cách thực tế hoặc logic các bản ghi. Vì vậy, tìm kiếm một bản ghi cụ thể trở nên nhanh hơn và có ít căng thẳng lên tài nguyên hệ thống.

11.1.6 Truy cập theo nhóm dữ liệu

Chỉ mục rất hữu ích khi dữ liệu cần phải được truy cập theo nhóm. Ví dụ, bạn muốn thực hiện các sửa đổi cho trợ cấp đi lại cho tất cả các nhân viên dựa trên bộ phận nơi họ làm việc. Ở đây, bạn muốn thực hiện những thay đổi cho tất cả các nhân viên trong một bộ phận trước khi chuyển sang các nhân viên trong bộ phận khác. Trong trường hợp này, chỉ mục có thể được tạo ra như được trình bày trong hình 11.6 trên cột **Department** trước khi truy cập các bản ghi.

Chỉ mục này sẽ tạo ra các khối logic của các hàng dữ liệu dựa trên bộ phận. Điều này một lần nữa sẽ hạn chế số lượng dữ liệu thực sự đã quét trong thời gian truy xuất truy vấn.

Do đó, truy xuất sẽ nhanh hơn và sẽ có ít căng thẳng hơn lên tài nguyên hệ thống

Department Name	Employee Name
Marketing	Jenny Woods
Marketing	Merry Thomas
Marketing	John Updeeke
Marketing	Robert Williamson
Sales	Smith Gordon
Sales	Albert Wang

Hình 11.6: Accessing Data Group-wise

11.2 Kiến trúc chỉ mục

Trong SQL Server 2012, có thể lưu trữ dữ liệu trong cơ sở dữ liệu theo cách có sắp xếp hoặc ngẫu nhiên. Nếu dữ liệu được lưu trữ theo một cách có sắp xếp, dữ liệu được cho là có mặt trong một cấu trúc ghép cụm. Nếu được lưu trữ một cách ngẫu nhiên, nó được cho là có mặt trong một cấu trúc heap.

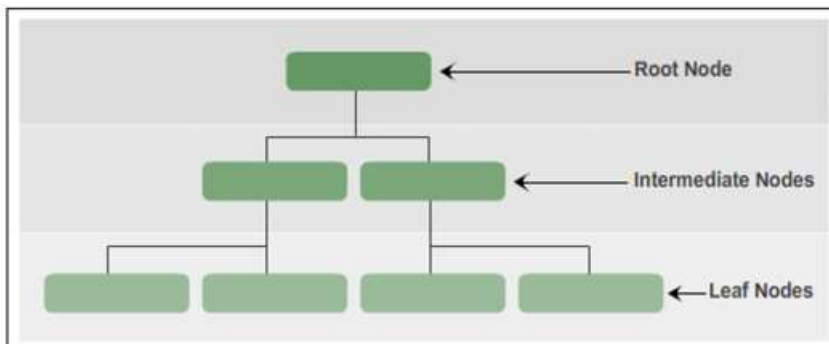
Hình 11.7 trình bày ví dụ minh họa kiến trúc chỉ mục.

Employee_Details		
EmpID	EmpName	DeptID
CN00020	Rosa Stevens	BN0001
CN00018	John Updeeke	BN0020
CN00019	Smith Gordon	BN0021
CN00012	Robert Tyson	BN0011
Heap Structure		
Employee_Details		
EmpID	EmpName	DeptID
CN00012	Robert Tyson	BN0011
CN00018	John Updeeke	BN0020
CN00019	Smith Gordon	BN0021
CN00020	Rosa Stevens	BN0001
Clustered Structure		

Hình 11.7: Index Architecture

11.2.1 B-Tree

Trong SQL Server, tất cả các chỉ mục được cấu trúc ở dạng B-Tree. Cấu trúc B-Tree có thể được hình dung như một cây ngược với gốc ngay ở trên cùng, chia thành các nhánh và sau đó, vào lá ngay ở dưới cùng như trong hình 11.8.



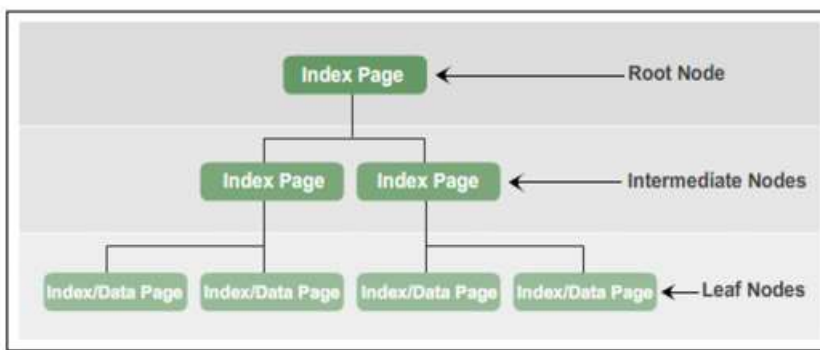
Hình 11.8: B-Tree

Trong cấu trúc B-Tree, có một nút gốc duy nhất ở trên cùng. Sau đó nút này phân nhánh vào mức tiếp theo, được gọi là mức ở giữa đầu tiên. Những nút ở mức ở giữa đầu tiên có thể phân nhánh ra thêm. Việc phân nhánh này có thể tiếp tục thành nhiều mức ở giữa và sau đó, cuối cùng là mức lá. Những nút ở mức lá được gọi là các nút lá.

Ghi chú - Chỉ mục B-Tree đi qua từ trên xuống dưới bằng cách sử dụng các con trỏ

11.2.2 Cấu trúc B-Tree của chỉ mục

Trong cấu trúc B-Tree của một chỉ mục, nút gốc bao gồm một trang chỉ mục. Trang chỉ mục này có chứa các con trỏ tới các trang chỉ mục có trong mức ở giữa đầu tiên. Những trang chỉ mục này đến lượt mình trỏ tới các trang chỉ mục có trong mức ở giữa tiếp theo. Có thể có nhiều mức ở giữa trong một B-Tree của chỉ mục. Các nút lá của B-Tree của chỉ mục có các trang dữ liệu chứa các hàng dữ liệu hoặc các trang chỉ mục có chứa các hàng chỉ mục trỏ đến các hàng dữ liệu như được trình bày trong hình 11.9.



Hình 11.9: Index B-Tree Structure

Các loại nút khác nhau như sau:

- **Nút gốc** - Chứa một trang chỉ mục với các con trỏ trỏ đến các trang chỉ mục ở mức ở giữa đầu tiên.
- **Các nút trung gian** - Chứa các trang chỉ mục với các con trỏ trỏ các trang chỉ mục ở mức trung gian tiếp theo hoặc tới các trang chỉ mục hoặc dữ liệu ở mức lá.
- **Các nút lá** - Chứa các trang dữ liệu hoặc các trang chỉ mục trỏ đến các trang dữ liệu.

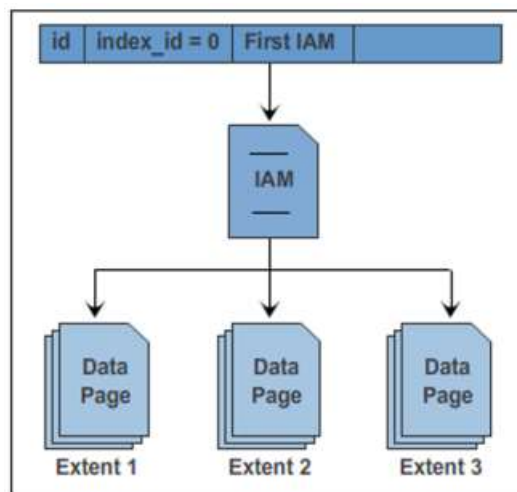
Ghi chú - Trang dữ liệu có chứa các mục nhập chỉ mục được gọi là trang chỉ mục

11.2.3 Các cấu trúc heap

Trong cấu trúc heap, các trang dữ liệu và bản ghi không được sắp xếp theo thứ tự được sắp xếp. Kết nối duy nhất giữa các trang dữ liệu là thông tin được ghi trong các trang Bản đồ phân bổ chỉ mục (IAM).

Trong SQL Server 2012, các trang IAM được sử dụng để quét qua một cấu trúc heap. Các trang IAM ánh xạ các mức độ được sử dụng bởi một đơn vị phân bổ trong một phần của tập tin cơ sở dữ liệu.

Heap có thể được đọc bằng cách quét các trang IAM để tìm những mức độ có chứa những trang cho heap đó như được trình bày trong hình 11.10.



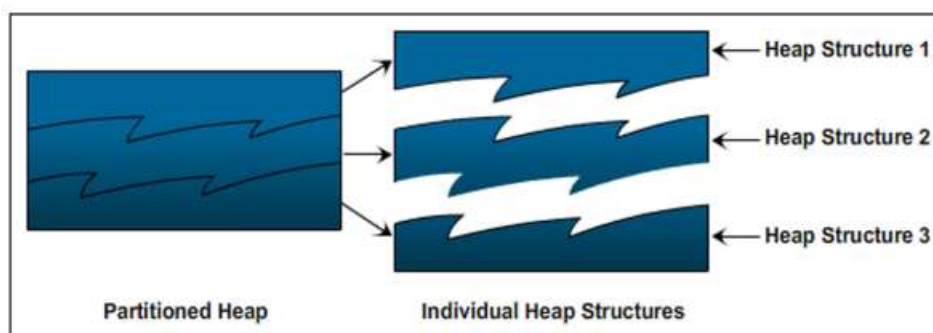
Hình 11.10: Heap Structures

Ghi chú - Nếu một đơn vị phân bổ có chứa các mức độ từ nhiều hơn một tập tin, sẽ có nhiều trang IAM được liên kết với nhau trong một chuỗi IAM để ánh xạ những mức độ này.

11.2.4 Phân vùng các cấu trúc heap

Bảng có thể được chia một cách logic thành các nhóm hàng nhỏ hơn. Sự phân chia này được gọi là phân vùng. Bảng được phân vùng để thực hiện các hoạt động bảo trì hiệu quả hơn. Theo mặc định, bảng có một phân vùng duy nhất.

Khi các phân vùng được tạo ra trong bảng có cấu trúc heap, mỗi phân vùng sẽ chứa dữ liệu trong cấu trúc heap riêng lẻ. Ví dụ, nếu heap có ba phân vùng, khi đó có ba cấu trúc heap hiện diện, một trong mỗi phân vùng như được trình bày trong hình 11.11.

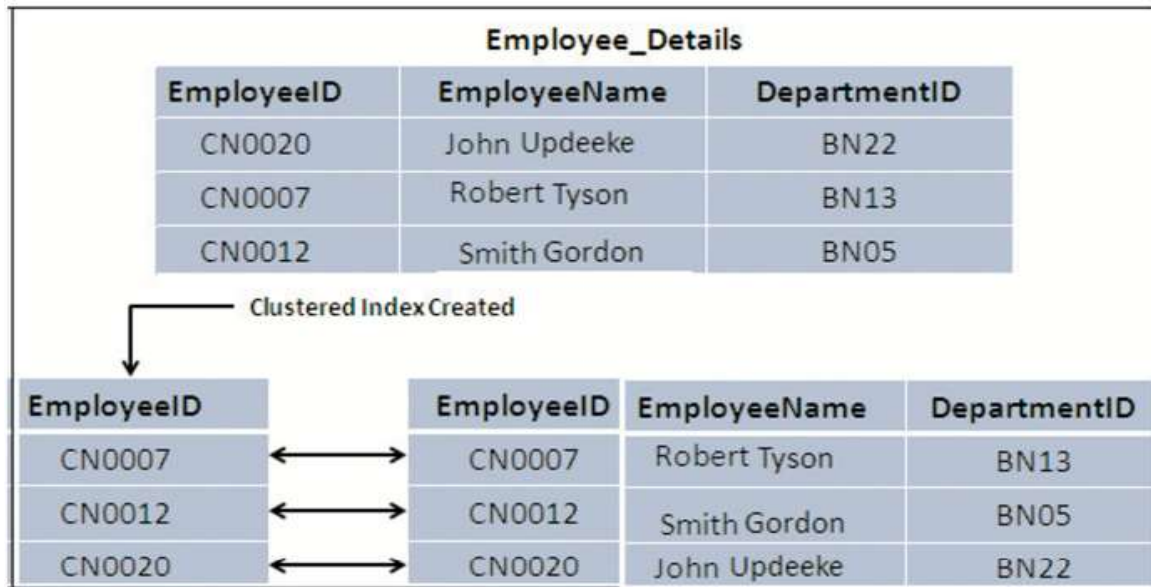


Hình 11.11: Partitioning of Heap Structure

11.2.5 Các cấu trúc chỉ mục ghép cụm

Chỉ mục ghép cụm làm cho các bản ghi được lưu trữ vật lý theo thứ tự sắp xếp hoặc tuần tự. Chỉ mục ghép cụm xác định thứ tự thực tế theo đó dữ liệu được lưu trữ trong cơ sở dữ liệu. Do đó, bạn có thể tạo ra chỉ một chỉ mục ghép cụm trong một bảng.

Tính duy nhất của một giá trị trong một chỉ mục ghép cụm được duy trì một cách rõ ràng sử dụng từ khóa UNIQUE hoặc một cách ngầm hiểu sử dụng một mã định danh duy nhất nội bộ như được trình bày trong hình 11.12.



Hình 11.12: Clustered Indexes

11.2.6 Tạo chỉ mục ghép cụm

Chỉ mục ghép cụm làm cho các bản ghi được lưu trữ vật lý theo thứ tự sắp xếp hoặc tuần tự. Do đó, chỉ mục ghép cụm xác định thứ tự thực tế theo đó dữ liệu được lưu trữ trong cơ sở dữ liệu. Vì vậy, bạn có thể tạo ra chỉ một chỉ mục ghép cụm trong một bảng.

Chỉ mục ghép cụm được tạo ra bằng cách sử dụng câu lệnh **CREATE INDEX** với từ khóa **CLUSTERED**. Cú pháp sau đây tạo ra một chỉ mục ghép cụm trên một bảng đã chỉ định.

Cú pháp:

```
CREATE CLUSTERED INDEX index_name ON <table name> (column_name)
```

trong đó,

CLUSTERED: Chỉ ra rằng một chỉ mục ghép cụm được tạo ra.

Code Snippet 2 tạo ra chỉ mục ghép cụm **IX_CustID** trên cột **CustID** trong bảng **Customer_Details**.

Code Snippet 2:

```
USE CUST_DB
CREATE CLUSTERED INDEX IX_CustID ON Customer_Details (CustID)
GO
```

Ghi chú - Trước khi bạn tạo ra một chỉ mục ghép cụm, bạn cần phải chắc chắn rằng không gian trống trong hệ thống của bạn có ít nhất 1,2 lần số lượng dữ liệu trong bảng.

11.2.7 Truy cập dữ liệu với một chỉ mục ghép cụm

Chỉ mục ghép cụm có thể được tạo ra trên một bảng sử dụng một cột không có các giá trị trùng lặp. Chỉ mục này sẽ tổ chức lại các bản ghi theo thứ tự tuần tự của những giá trị trong cột chỉ mục.

Chỉ mục ghép cụm được sử dụng để định vị một hàng đơn lẻ hoặc một loạt các hàng. Bắt đầu từ trang đầu tiên của chỉ mục, giá trị tìm kiếm được kiểm tra đối với mỗi giá trị khóa trên trang này. Khi giá trị khóa so khớp được tìm thấy, công cụ cơ sở dữ liệu chuyển đến trang được chỉ ra bằng giá trị đó như được trình bày trong hình 11.13. Hàng mong muốn hoặc một loạt các hàng sau đó được truy cập.

Chỉ mục ghép cụm rất hữu ích cho các cột được tìm kiếm thường xuyên cho các giá trị khóa hoặc được truy cập theo thứ tự sắp xếp.

Clustered Index

CustomerID	CustomerName
CN0001	John Updeeke
CN0002	Smith Gordon
CN0003	Albert Wang
CN0004	Rosa Stines
CN0005	Jenny Woods

Data is physically stored in a sorted manner

Hình 11.13: Accessing Data with a Clustered Index

Chỉ mục ghép cụm được tự động tạo ra trên một bảng khi khóa chính được định nghĩa trên bảng đó. Trong bảng không có cột khóa chính, chỉ mục ghép cụm lý tưởng nên được định nghĩa trên:

- Cột khóa được tìm kiếm trên diện rộng.
- Cột được sử dụng trong các truy vấn trả lại các tập kết quả lớn.
- Cột có dữ liệu duy nhất.
- Cột được sử dụng trong phép nối bảng.

Ghi chú - Hai hoặc nhiều bảng có thể được nối theo logic qua các cột là phổ biến cho những bảng này. Sau đó dữ liệu có thể được lấy từ những bảng này như thể chúng là một bảng đơn lẻ.

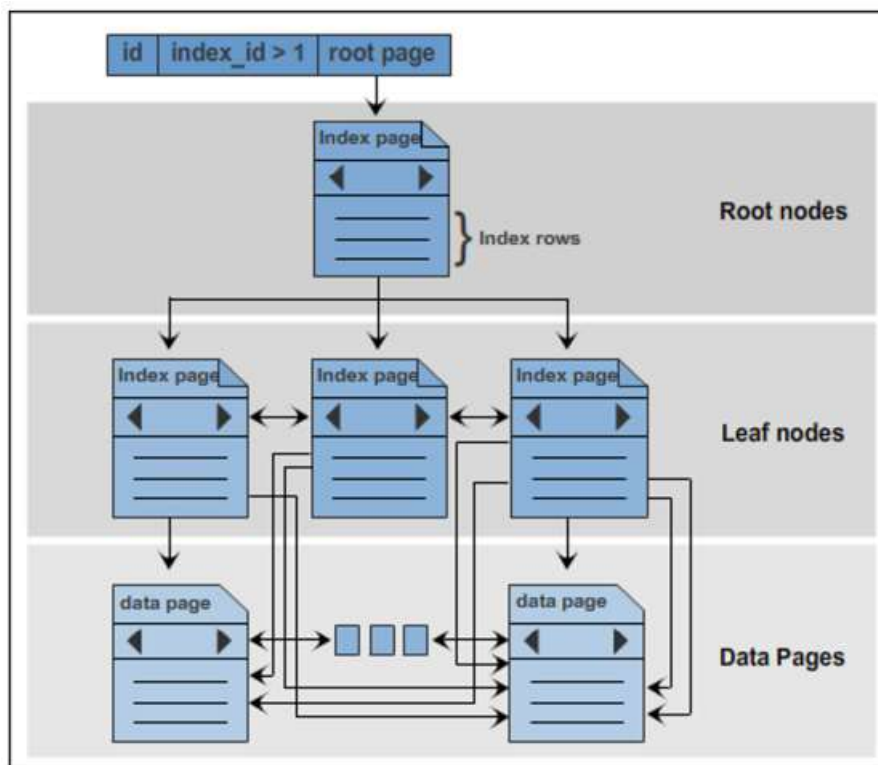
11.2.8 Các cấu trúc chỉ mục không ghép cụm

Chỉ mục không ghép cụm được định nghĩa trên bảng có dữ liệu trong cấu trúc hoặc heap ghép cụm. Chỉ mục không ghép cụm sẽ là loại mặc định nếu chỉ mục không được định nghĩa trên bảng. Mỗi hàng chỉ mục trong chỉ mục không ghép cụm chứa một giá trị khóa không ghép cụm và bộ định vị hàng. Bộ định vị hàng này trỏ tới hàng dữ liệu tương ứng với giá trị khóa trong bảng.

Các chỉ mục không ghép cụm có một cấu trúc B-Tree tương tự như các chỉ mục ghép cụm nhưng với những khác biệt sau :

- Những hàng dữ liệu của bảng không được lưu trữ vật lý theo thứ tự được định nghĩa bằng các khóa không ghép cụm của chúng.
- Trong cấu trúc chỉ mục không ghép cụm, mức lá có chứa các hàng chỉ mục.

Hình 11.14 trình bày cấu trúc chỉ mục không ghép cụm.



Hình 11.14: Nonclustered Index Structure

Các chỉ mục không ghép cụm rất hữu ích khi bạn cần nhiều cách để tìm kiếm dữ liệu. Một số thực tế và hướng dẫn phải được xem xét trước khi tạo ra một chỉ mục không ghép cụm như sau :

- Khi một chỉ mục ghép cụm được tái tạo hoặc tùy chọn DROP _ EXISTING được sử dụng, SQL Server xây dựng lại các chỉ mục không ghép cụm hiện có.
- Một bảng có thể có đến 999 chỉ mục không ghép cụm.
- Tạo chỉ mục ghép cụm trước khi tạo ra một chỉ mục không ghép cụm.

Cú pháp sau đây tạo ra một chỉ mục không ghép cụm.

Cú pháp:

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

trong đó,

NONCLUSTERED: chỉ ra rằng một chỉ mục không ghép cụm được tạo ra.

Code Snippet 3 tạo ra chỉ mục không ghép cụm **IX_State** trên cột **State** trong bảng **Customer_Details**.

Code Snippet 3:

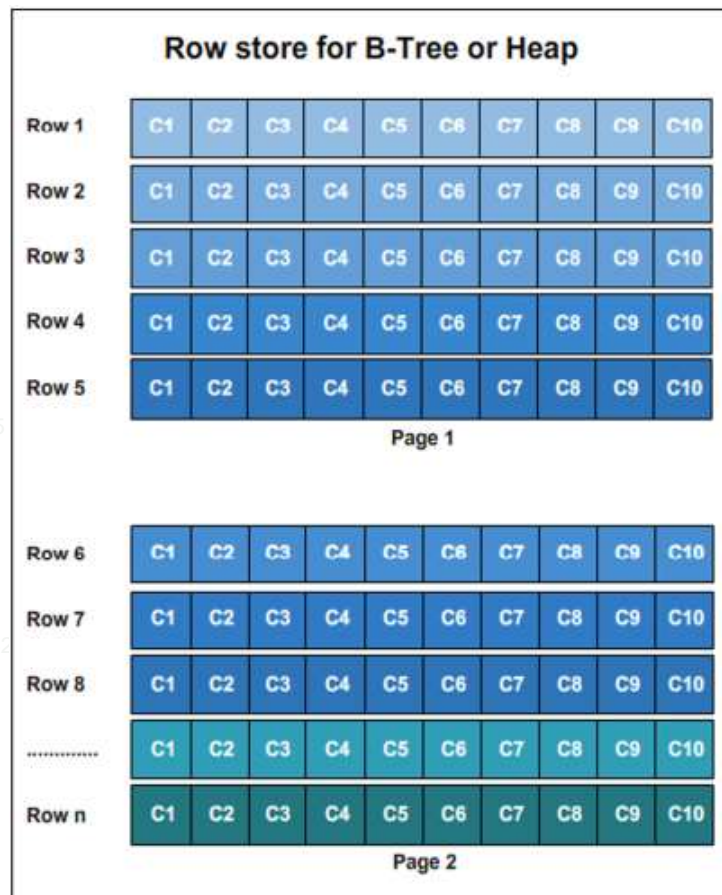
```
USE CUST_DB  
CREATE NONCLUSTERED INDEX IX_State ON Customer_Details (State)  
GO
```

11.2.8 Chỉ mục lưu trữ cột

Chỉ mục lưu trữ cột là một tính năng mới trong SQL Server 2012. Nó giúp tăng cường hiệu suất của các truy vấn kho dữ liệu một cách bao quát. Những chỉ mục thông thường hoặc các heap của các máy chủ SQL cũ lưu trữ dữ liệu trong cấu trúc B-Tree theo hàng, nhưng chỉ mục lưu trữ cột trong SQL Server 2012 lưu trữ dữ liệu theo cột. Do tốc độ truyền dữ liệu chậm trong các máy chủ cơ sở dữ liệu, do đó chỉ mục lưu trữ cột sử dụng phép nén mạnh mẽ để giảm nhập/xuất đĩa cần thiết để phục vụ yêu cầu truy vấn.

B-Tree và heap lưu trữ dữ liệu theo hàng, có nghĩa là dữ liệu từ tất cả các cột của một hàng được lưu trữ cùng nhau liên tục kế nhau trên cùng một trang.

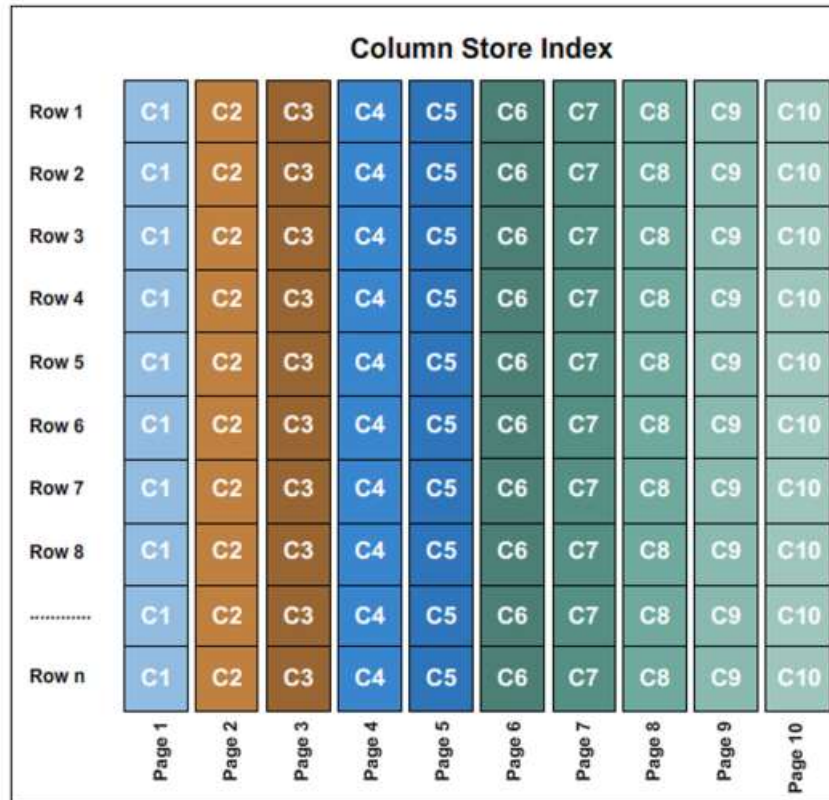
Ví dụ, nếu có một bảng với mười cột (C1 đến C10), dữ liệu của tất cả mười cột từ mỗi hàng được lưu trữ cùng nhau liên tục kế nhau trên cùng một trang như được trình bày trong hình 11.15.



Hình 11.15: A B-Tree Index

Khi chỉ mục lưu trữ cột được tạo ra, dữ liệu được lưu trữ theo cột, có nghĩa là dữ liệu của mỗi cột riêng lẻ từ mỗi hàng được lưu trữ với nhau trên cùng một trang.

Ví dụ, dữ liệu của cột C1 của tất cả các hàng được lưu trữ cùng nhau trên một trang và dữ liệu cho cột C2 của tất cả các hàng được lưu trữ trên một trang khác và vân vân như được trình bày trong hình 11.16.



Hình 11.16: A Column Store Index

Sau đây là cú pháp để tạo ra một chỉ mục lưu trữ cột:

Cú pháp:

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

Giả sử rằng một bảng có tên là **ResellerSalesPtnd** đã được tạo ra trong cơ sở dữ liệu AdventureWorks2012. Code Snippet 4 trình bày cách tạo ra một chỉ mục lưu trữ cột trên bảng này.

Code Snippet 4:

```
CREATE NONCLUSTERED COLUMNSTORE INDEX [csindx_ResellerSalesPtnd]
ON [ResellerSalesPtnd]
```

```
{  
    [ProductKey],  
    [OrderDateKey],  
    [DueDateKey],  
    [ShipDateKey],  
    [CustomerKey],  
    [EmployeeKey],  
    [PromotionKey],  
    [CurrencyKey],  
    [SalesTerritoryKey],  
    [SalesOrderNumber],  
    [SalesOrderLineNumber],  
    [RevisionNumber],  
    [OrderQuantity],  
    [UnitPrice],  
    [ExtendedAmount],  
    [UnitPriceDiscountPct],  
    [DiscountAmount],  
    [ProductStandardCost],  
    [TotalProductCost],  
    [SalesAmount],  
    [TaxAmt],  
    [Freight],  
    [CarrierTrackingNumber],  
    [CustomerPONumber],  
    [OrderDate],  
    [DueDate],  
    [ShipDate]  
};
```

Ghi chú - COLUMNSTORE INDEX chỉ hoạt động trên phiên bản doanh nghiệp của SQL Server 2012.

11.2.10 Thả bỏ một chỉ mục

Khi thả bỏ một chỉ mục ghép cụm, những hàng ở cấp độ lá của chỉ mục ghép cụm được sao chép vào heap. Tất cả các chỉ mục không ghép cụm trên bảng này sau đó sẽ trở đến heap trong đó dữ liệu được lưu trữ. Điều này được thực hiện bằng cách xây dựng lại các chỉ mục không ghép cụm khi chỉ mục ghép cụm được thả bỏ. Do đó, việc thả bỏ chỉ mục ghép cụm là một quá trình tốn nhiều thời gian. Vì vậy, trong khi thả bỏ tất cả các chỉ mục trên bảng, trước tiên bạn phải thả bỏ các chỉ mục không ghép cụm đầu tiên và sau đó, các chỉ mục ghép cụm.

SQL Server 2012 có thể thả bỏ chỉ mục ghép cụm và di chuyển heap (bảng không có thứ tự) vào một nhóm tập tin khác hoặc một lược đồ phân vùng sử dụng tùy chọn MOVE TO.

- Tùy chọn này là không hợp lệ cho các chỉ mục không ghép cụm.
- Lược đồ phân vùng hoặc nhóm tập tin được chỉ ra trong mệnh đề MOVE TO phải tồn tại.
- Bảng sẽ được đặt trong cùng một lược đồ phân vùng hoặc nhóm tập tin của chỉ mục ghép cụm đã thả bỏ.

Sau đây là cú pháp để thả bỏ một chỉ mục ghép cụm.

Cú pháp:

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

trong đó,

index_name: chỉ ra tên của chỉ mục.

partition_scheme_name: chỉ ra tên của lược đồ phân vùng.

filegroup_name: chỉ ra tên của nhóm tập tin để lưu trữ các phân vùng.

default: chỉ ra vị trí mặc định để lưu trữ bảng kết quả.

Code Snippet 5 thả bỏ chỉ mục **IX_SuppID** được tạo ra trên cột **SuppID** của bảng **Supplier_Details**.

Code Snippet 5:

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

Dữ liệu trong bảng **Supplier_Details** kết quả được chuyển đến vị trí mặc định.

Code Snippet 6 thả bỏ chỉ mục **IX_SupplID** được tạo ra trên cột **SupplID** của bảng **Supplier_Details**.

Code Snippet 6:

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

Dữ liệu trong bảng **Supplier_Details** kết quả được chuyển đến nhóm tập tin **FGCountry**.

11.2.11 Sự khác nhau giữa chỉ mục ghép cụm và không ghép cụm

Các chỉ mục ghép cụm và không ghép cụm là khác nhau về kiến trúc của chúng và tính hữu dụng của chúng trong các lần thực thi truy vấn. Bảng 11.1 liệt kê ra những khác biệt giữa các chỉ mục ghép cụm và không ghép cụm.

Các chỉ mục ghép cụm	Các chỉ mục không ghép cụm
Được sử dụng trong các truy vấn trả lại các tập kết quả lớn	Được sử dụng trong các truy vấn không trả lại các tập kết quả lớn
Chỉ một chỉ mục ghép cụm có thể được tạo ra trên một bảng	Nhiều chỉ mục không ghép cụm có thể được tạo ra trên một bảng
Dữ liệu được lưu trữ theo một cách có sắp xếp trên khóa ghép cụm	Dữ liệu không được lưu trữ theo một cách có sắp xếp trên khóa không ghép cụm
Các nút lá của một chỉ mục ghép cụm chứa các trang dữ liệu	Những nút lá của một chỉ mục không ghép cụm chứa các trang chỉ mục

Table 11.1: Differences Between Clustered and Nonclustered Indexes

11.2.12 XML Indexes

Kiểu dữ liệu xml được sử dụng để lưu trữ các tài liệu và phân đoạn XML như được trình bày trong hình 11.17. Một đoạn XML là một thể hiện XML có một phần tử mức cao nhất bị mất tích.

Do kích thước lớn của các cột XML, các truy vấn tìm kiếm trong những cột này có thể sẽ chậm. Bạn có thể tăng tốc những truy vấn này bằng cách tạo ra một chỉ mục XML trên mỗi cột. Chỉ mục XML có thể là một chỉ mục ghép cụm hoặc không ghép cụm. Mỗi bảng có thể có đến 249 chỉ mục XML.

	DocID	DocumentStore
▶	1	<Document Name="Poem"><AuthorName>Bruce...
	2	<Document Name="Code"><AuthorName>Hammu...
	3	<Document Name="Jack and Jill"><AuthorName>...

Hình 11.17: XML Data Type

Ghi chú - XML là một văn bản đơn giản, ngôn ngữ dựa trên Unicode cung cấp cơ chế để mô tả cấu trúc tài liệu sử dụng các thẻ đánh dấu. Ví dụ, hãy xem xét một tổ chức có các chi tiết của nhân viên được lưu trữ trong tài liệu XML. Thông tin cho mỗi nhân viên được lưu trữ ở định dạng sau đây:

```
<Employees>
<Name>John</Name>
<Age>34</Age>
<Salary>500000</Salary>
</Employees>
```

Ở đây, **<Employees>** là nút gốc và **<Name>**, **<Age>**, và **<Salary>** là những nút con.

11.2.13 Các loại XML Indexes

Các chỉ mục XML có thể được tạo ra trên một bảng chỉ khi có một chỉ mục ghép cụm dựa trên khóa chính của bảng. Khóa chính này không thể vượt quá 15 cột.

Những loại chỉ mục XML khác nhau như sau:

- **Chỉ mục XML chính** - Quá trình thực hiện các truy vấn trong một cột XML đôi khi có thể rất chậm. Chỉ mục XML chính được tạo ra trên mỗi cột XML để tăng tốc những truy vấn này. Nó là một chỉ mục đặc biệt cất nhỏ dữ liệu XML để lưu trữ thông tin. Sau đây là cú pháp để tạo ra một chỉ mục XML chính.

Cú pháp:

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

Code Snippet 7 tạo ra một chỉ mục XML chính trên cột **CatalogDescription** trong bảng **Production.ProductModel**

Code Snippet 7:

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

- **Chỉ mục XML phụ** - Chỉ mục XML phụ là chỉ mục XML chuyên biệt trợ giúp với các truy vấn XML cụ thể. Những tính năng của các chỉ mục XML phụ như sau:
 - Tìm kiếm các giá trị bất cứ nơi nào trong tài liệu XML.
 - Lấy các thuộc tính đối tượng đặc biệt từ bên trong một tài liệu XML.

Các chỉ mục XML phụ chỉ có thể được tạo ra trên các cột đã có một chỉ mục XML chính.
Code Snippet 8 trình bày cách tạo ra một chỉ mục XML phụ trên cột **CatalogDescription** trong bảng **Production.ProductModel**.

Code Snippet 8:

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

- **Chỉ mục XML có lựa chọn (SXI)** – Đây là một loại chỉ mục XML mới đã được giới thiệu trong SQL Server 2012. Những tính năng của chỉ mục mới này là để cải thiện hiệu suất truy vấn trên dữ liệu được lưu trữ là XML trong SQL Server, cho phép lập chỉ mục nhanh hơn của khối lượng công việc dữ liệu XML lớn, và cải thiện khả năng mở rộng bằng cách giảm chi phí lưu trữ của chỉ mục. Sau đây là cú pháp để tạo ra một chỉ mục XML có lựa chọn:

Cú pháp:

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

Sau đây là một tài liệu XML trong một bảng gồm khoảng 500.000 hàng.

```
<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>
```

Code Snippet 9 trình bày cách tạo ra một chỉ mục XML có lựa chọn trên cột **BookDetails** trong bảng **BooksBilling**.

Code Snippet 9:

```
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 SQLNVARCHAR(100)
)
GO
```

Ghi chú - SELECTIVE XML INDEX sẽ chỉ làm việc trong phiên bản doanh nghiệp của SQL Server 2012.

11.2.14 Sửa đổi chỉ mục XML

Chỉ mục XML, chính hay phụ, có thể được sửa đổi bằng cách sử dụng câu lệnh ALTER INDEX.

Cú pháp:

```
ALTER INDEX <xml_index_name> ON <table_name> REBUILD
```

trong đó,

xml_index_name : chỉ ra tên của chỉ mục XML.

Code Snippet 10 xây dựng lại chỉ mục XML chính **PXML_DocumentStore** được tạo ra trên bảng **XMLDocument**.

Code Snippet 10:

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

11.2.15 Loại bỏ chỉ mục XML

Sau đây là cú pháp để loại bỏ chỉ mục XML sử dụng câu lệnh DROP INDEX.

Cú pháp :

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

Code Snippet 11 loại bỏ chỉ mục XML chính **PXML_DocumentStore** được tạo ra trên bảng **XMLDocument**.

Code Snippet 11:

```
DROP INDEX PXML_DocumentStore ON XMLDocument
```

11.3 Các đơn vị phân bổ

Heap hoặc cấu trúc chỉ mục ghép cụm có chứa các trang dữ liệu trong một hoặc nhiều đơn vị phân bổ. Đơn vị phân bổ là một tập hợp các trang và được sử dụng để quản lý dữ liệu dựa trên loại trang của chúng. Những loại đơn vị phân bổ được sử dụng để quản lý dữ liệu trong các bảng và chỉ mục như sau:

➤ **IN_ROW_DATA**

Nó được sử dụng để quản lý các hàng dữ liệu hoặc chỉ mục có chứa tất cả các loại dữ liệu ngoại trừ dữ liệu đối tượng lớn (LOB).

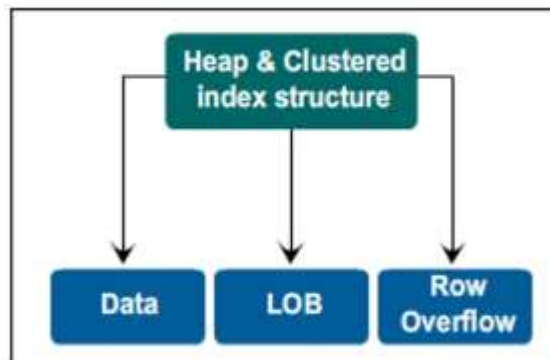
➤ **LOB_DATA**

Nó được sử dụng để quản lý dữ liệu đối tượng lớn, được lưu trữ trong một hoặc nhiều kiểu dữ liệu sau: varbinary(max), varchar(max), và xml.

➤ **ROW_OVERFLOW_DATA**

Nó được sử dụng để quản lý dữ liệu có độ dài biến đổi, được lưu trữ trong các cột varchar, nvarchar, varbinary, hoặc sql_variant.

Hình 11.18 trình bày các đơn vị phân bổ.



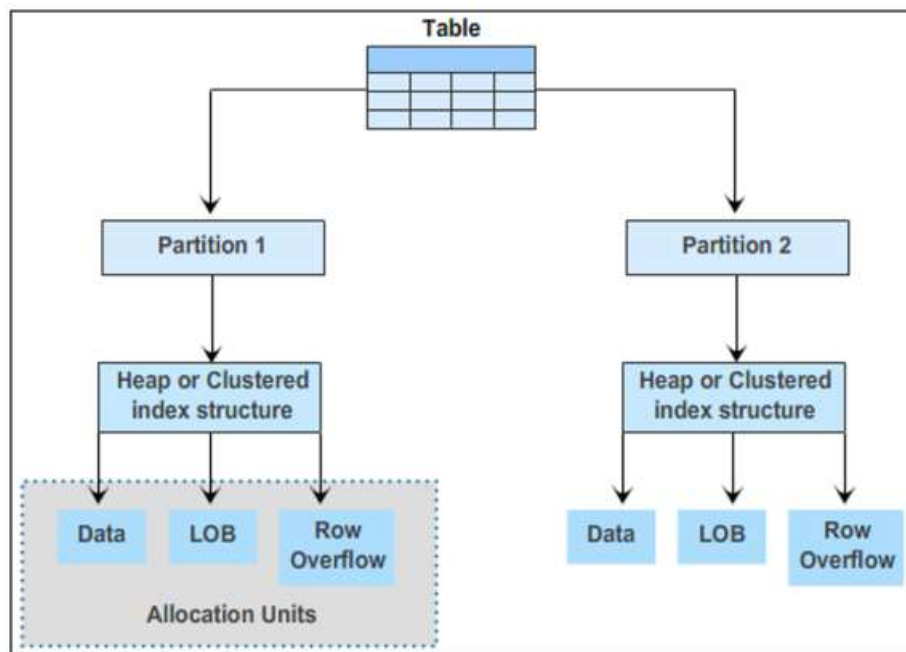
Hình 11.18: Allocation Units

Ghi chú - Heap có thể chỉ có một đơn vị phân bổ thuộc từng loại trong một phân vùng cụ thể của một bảng.

11.4 Phân vùng

Phân vùng chia dữ liệu thành các tập con. Điều này làm cho các bảng lớn hoặc chỉ mục dễ quản lý hơn. Phân vùng cho phép bạn truy cập dữ liệu một cách nhanh chóng và hiệu quả. Các hoạt động bảo trì trên một phần nhỏ của dữ liệu được thực hiện hiệu quả hơn bởi vì chúng chỉ nhắm tới phần nhỏ của dữ liệu yêu cầu thay vì toàn bộ bảng.

Theo mặc định, bảng hoặc chỉ mục chỉ có một phân vùng chứa tất cả các dữ liệu hoặc các trang chỉ mục. Khi một bảng hoặc chỉ mục sử dụng nhiều phân vùng, dữ liệu được phân chia theo chiều ngang thành các nhóm hàng như được trình bày trong hình 11.19.



Hình 11.19: Partitioning

11.4.1 Khung nhìn sys.partitions

Khung nhìn sys.partitions là một khung nhìn hệ thống có chứa thông tin đầy đủ về những phân vùng khác nhau của tất cả các bảng và chỉ mục trong cơ sở dữ liệu.

Bảng 11.2 trình bày các cột khác nhau của khung nhìn sys.partitions cùng với các kiểu dữ liệu và mô tả của chúng:

Tên Cột	Loại dữ liệu	Mô tả
partition_id	bigint	Chứa id của phân vùng và là duy nhất trong một cơ sở dữ liệu.
Object_id	int	Chứa id của đối tượng có chứa phân vùng đó.

Tên cột	Data Type	Mô tả
Index_id	int	Chứa id của chỉ mục nơi phân vùng thuộc đó
Partition_number	int	Chứa số phân vùng trong chỉ mục hoặc heap.
hobt_id	bigint	Chứa id của heap dữ liệu hoặc B-Tree có chứa các hàng cho phân vùng đó.
rows	bigint	Chỉ rõ số lượng gần đúng của các hàng trong phân vùng.

Table 11.2: Columns of the sys.partitions View and Data Types

11.4.2 Cột index_id

Cột **index_id** chứa id của chỉ mục nơi phân vùng thuộc vào đó. Khung nhìn danh mục **sys.partitions** trả về một hàng cho mỗi phân vùng trong một bảng hoặc chỉ mục. Các giá trị của cột **index_id** là duy nhất trong bảng, trong đó phân vùng được tạo ra. Kiểu dữ liệu của cột **index_id** là **int**.

Sau đây là các giá trị khác nhau của cột **index_id** :

- Giá trị **index_id** cho một heap là 0.
- Giá trị **index_id** cho một chỉ mục ghép cụm là 1.
- Giá trị **index_id** cho một chỉ mục không ghép cụm lớn hơn 1.
- Giá trị **index_id** cho các đối tượng lớn sẽ lớn hơn 250

Hình 11.20 trình bày cột **index_id** trong khung nhìn **sys.partitions**.

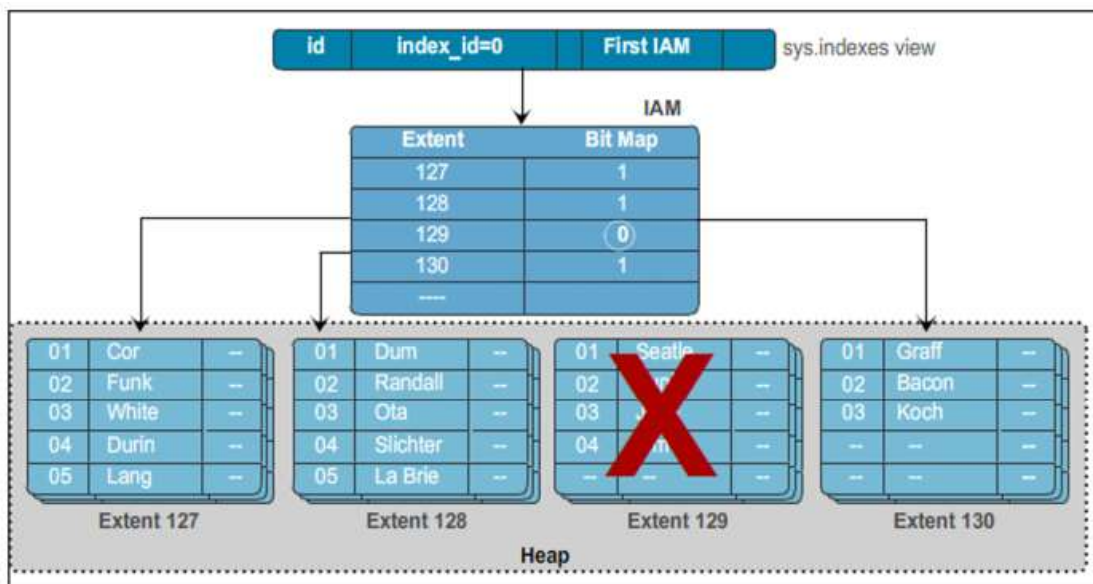
	partition_id	object_id	index_id	partition_number	hobt_id	rows
1	196608	3	1	1	196608	1779
2	327680	5	1	1	327680	332
3	458752	7	1	1	458752	379
4	524288	8	0	1	524288	2
5	281474977103872	6	1	1	281474977103872	0
6	281474977300480	9	1	1	281474977300480	0
7	281474977824768	17	1	1	281474977824768	0
8	281474977890304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

Hình 11.20: The Index_id Column

11.5 Tìm các hàng

SQL Server sử dụng các khung nhìn danh mục để tìm các hàng khi chỉ mục không được tạo ra trên một bảng. Nó sử dụng khung nhìn sys.indexes để tìm trang IAM. Trang IAM này chứa một danh sách tất cả các trang của một bảng cụ thể thông qua đó SQL Server có thể đọc tất cả các trang dữ liệu.

Khi khung nhìn sys.indexes được sử dụng, trình tối ưu truy vấn kiểm tra tất cả các hàng trong một bảng và trích xuất chỉ những hàng được tham chiếu trong truy vấn như được trình bày trong hình 11.21. Việc quét này tạo ra nhiều hoạt động nhập/xuất và tận dụng nhiều tài nguyên.



Hình 11.21: Finding Rows without Indexes

Code Snippet 12 trình bày cách tạo ra bảng **Employee_Details** không có chỉ mục.

Code Snippet 12:

```
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 rằng nhiều bản ghi được chèn vào trong bảng **Employee_Details**. Câu lệnh SELECT được sử dụng để tìm kiếm các bản ghi có **FirstName** là John.

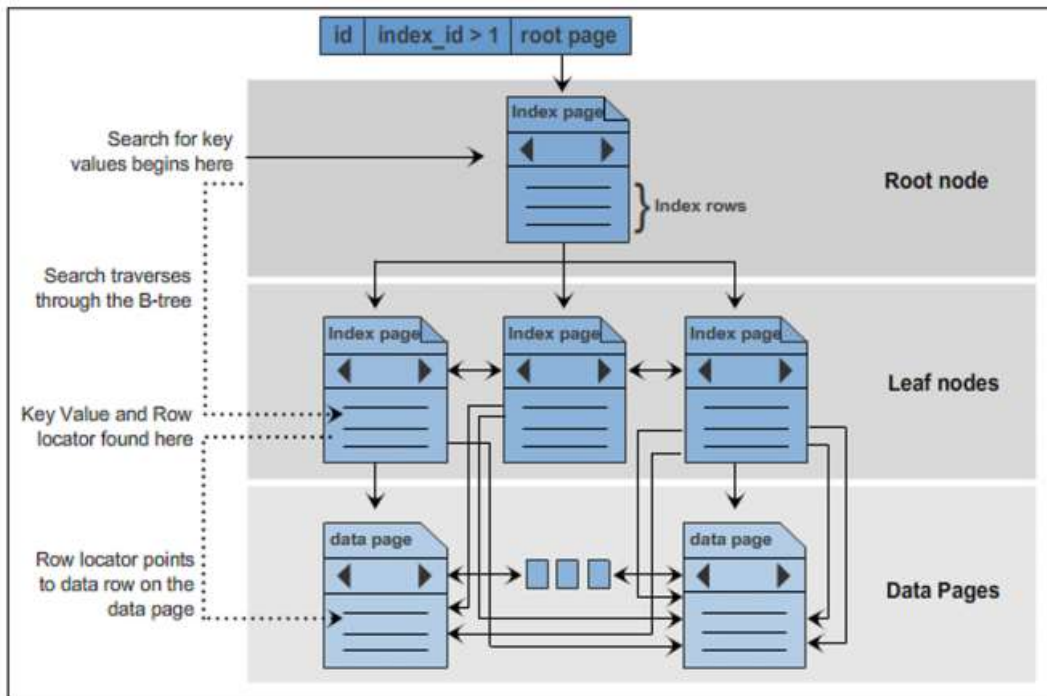
Vì không có chỉ mục liên quan đến cột **FirstName**, SQL Server sẽ thực hiện một lần quét bảng đầy đủ.

11.5.1 Tìm các dòng có chỉ mục không ghép cụm

Chỉ mục không ghép cụm tương tự như chỉ mục cuốn sách, dữ liệu và chỉ mục được lưu trữ ở những nơi khác nhau. Những con trỏ ở cấp độ lá của chỉ mục trỏ đến vị trí lưu trữ của dữ liệu trong bảng bên dưới. Chỉ mục không ghép cụm được sử dụng để tìm kiếm các truy vấn khớp chính xác. Điều này là do chỉ mục chứa các mục nhập mô tả vị trí chính xác của dữ liệu trong bảng.

Để tìm kiếm các hàng sử dụng các chỉ mục không ghép cụm, câu lệnh SELECT được sử dụng với cột chỉ mục không ghép cụm đã chỉ định trong mệnh đề WHERE.

Hình 11.22 trình bày quá trình tìm các hàng với chỉ mục không ghép cụm.



Hình 11.22: Finding Rows with Nonclustered Index

Code Snippet 13 trình bày cách tạo ra một chỉ mục không ghép cụm **IX_EmployeeCity** trên cột **City** của bảng **Employee_Details**.

Code Snippet 13:

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

Giả sử rằng nhiều bản ghi được chèn vào trong bảng **Employee_Details**. Câu lệnh **SELECT** được sử dụng để tìm kiếm các hồ sơ của người lao động từ thành phố Boston như được trình bày trong Code Snippet 14.

Code Snippet 14:

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

Do có một chỉ mục không ghép cụm liên quan đến cột **City**, SQL Server sẽ sử dụng chỉ mục **IX_EmployeeCity** để trích xuất các bản ghi như được trình bày trong hình 11.23.

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

Hình 11.23: The IX_EmployeeCity Index

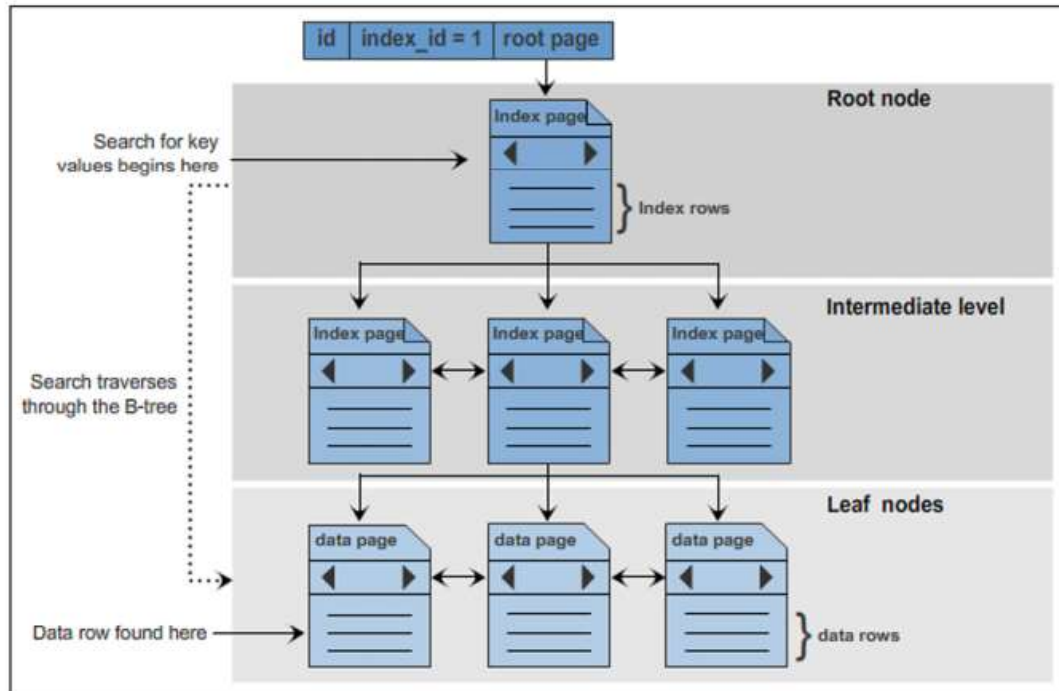
Ghi chú - Trong bảng có một chỉ mục không ghép cụm, không có thứ tự lưu trữ cụ thể của dữ liệu. Dữ liệu được lấy trực tiếp từ vị trí vật lý của nó.

11.5.2 Tìm các hàng trong một chỉ mục ghép cụm

Các chỉ mục ghép cụm lưu trữ các hàng dữ liệu trong bảng dựa trên các giá trị khóa của chúng. Dữ liệu này được lưu trữ theo một cách có sắp xếp. Nếu giá trị khóa ghép cụm nhỏ, có thể đặt thêm số lượng các hàng chỉ mục trên trang chỉ mục. Điều này làm giảm số lượng các mức trong B-Tree chỉ mục phải đi qua để đạt tới các hàng dữ liệu tạo ra các kết quả truy vấn nhanh hơn. Điều này làm giảm thiểu tổng lượng nhập/xuất.

Để tìm kiếm các hàng sử dụng các chỉ mục ghép cụm, câu lệnh **SELECT** được sử dụng với cột chỉ mục ghép cụm đã chỉ ra trong mệnh đề **WHERE**.

Hình 11.24 trình bày quá trình tìm các hàng với chỉ mục ghép cụm.



Hình 11.24: Finding Rows with Clustered Index

Code Snippet 15 trình bày cách tạo ra một chỉ mục ghép cụm **IX_EmployeeID** trên cột **EmpID** của bảng **Employee_Details**.

Code Snippet 15:

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

Giả sử rằng nhiều bản ghi được chèn vào trong bảng **Employee_Details**. Câu lệnh SELECT được sử dụng để tìm kiếm các hồ sơ của người lao động có **EmpID** từ 102 đến 105 như được trình bày trong Code Snippet 16.

Code Snippet 16:

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

Do có một chỉ mục ghép cụm liên quan đến cột **EmpID**, SQL Server sẽ sử dụng chỉ mục **IX_EmployeeID** để trích xuất các bản ghi như được trình bày trong hình 11.25.

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

Hình 11.25: The IX_EmployeeID Index

11.5.3 Tạo các chỉ mục duy nhất

Chỉ mục duy nhất có thể được tạo ra trên một cột không có bất kỳ giá trị trùng lặp nào. Ngoài ra, một khi chỉ mục duy nhất được tạo ra, giá trị trùng lặp sẽ không được nhận trong cột này. Do đó, các chỉ mục duy nhất nên được tạo ra chỉ trên các cột nơi tính duy nhất của các giá trị là một đặc điểm quan trọng. Chỉ mục duy nhất đảm bảo tính toàn vẹn thực thể trong một bảng.

Nếu định nghĩa bảng có cột **PRIMARY KEY** hoặc một cột với ràng buộc **UNIQUE**, SQL Server tự động tạo ra một chỉ mục duy nhất khi bạn thực thi câu lệnh **CREATE TABLE** như được trình bày trong hình 11.26.

Chỉ mục duy nhất có thể được tạo ra bằng cách sử dụng câu lệnh **CREATE UNIQUE INDEX** hoặc sử dụng **SSMS**.

Customer_Details	
CustID	FirstName
1	John
2	Sam
3	Julie
4	Robert
5	John
6	George
7	Robert
8	Merry

Hình 11.26: Creating Unique Indexes

Cú pháp sau đây được sử dụng để tạo ra chỉ mục duy nhất.

Cú pháp :

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

trong đó,

column_name: chỉ ra tên của cột trên đó chỉ mục sẽ được tạo ra.

UNIQUE: chỉ ra rằng không có các giá trị trùng lặp nào được cho phép trong cột này.

Code Snippet 17 tạo ra chỉ mục duy nhất trên cột **CustID** trong bảng **Customer_Details**.

Code Snippet 17:

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

11.5.4 Tạo các cột tính

Cột tính là một cột ảo trong một bảng có giá trị được tính vào thời gian chạy. Những giá trị trong cột này không được lưu trữ trong bảng nhưng được tính toán dựa trên biểu thức định nghĩa cột đó. Biểu thức này bao gồm tên cột không tính toán kết hợp với hằng số, hàm, hoặc biến sử dụng các toán tử số học hoặc logic.

Một cột tính có thể được tạo ra sử dụng các câu lệnh CREATE TABLE hoặc ALTER TABLE như được trình bày trong hình 11.27.

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

Computed column
(A=L*B)

Hình 11.27: Creating Computed Columns

Cú pháp sau đây được sử dụng để tạo ra một cột tính.

Cú pháp:

```
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 của cột tính cũng như biểu thức định nghĩa những giá trị trong cột này.

Code Snippet 18 tạo ra cột tính **Area** với giá trị của nó có được tính toán từ các giá trị nhập vào các trường **Length** và **Breadth**.

Code Snippet 18:

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

Ghi chú - Không thể sử dụng một cột tính làm một phần của bất kỳ định nghĩa ràng buộc PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, hay CHECK

11.5.5 Tạo chỉ mục trên các cột tính

Có thể tạo ra chỉ mục trên một cột tính nếu cột đó được đánh dấu **PERSISTED**. Điều này đảm bảo rằng Công cụ cơ sở dữ liệu lưu trữ các giá trị tính toán trong bảng. Những giá trị này được cập nhật khi có bất kỳ cột nào khác nơi cột tính phụ thuộc vào đó sẽ được cập nhật.

Công cụ cơ sở dữ liệu sử dụng giá trị tiếp tục này khi nó tạo ra một chỉ mục trên cột.

Chỉ mục được tạo ra trên cột tính bằng cách sử dụng câu lệnh **CREATE INDEX** như được trình bày trong hình 11.28.

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

Hình 11.28: Creating Index on Computed Columns

Cú pháp sau đây tạo ra một chỉ mục trên cột tính.

Cú pháp :

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

trong đó,

computed_column_name : chỉ ra tên của cột tính.

Code Snippet 19 tạo ra một chỉ mục **IX_Area** trên cột tính **Area**.

Code Snippet 19:

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

11.6 Cursors - Các con trỏ

Một đối tượng cơ sở dữ liệu được sử dụng để lấy dữ liệu mỗi lần một hàng, từ một tập kết quả được gọi là con trỏ. Con trỏ được sử dụng thay vì các lệnh Transact-SQL hoạt động trên tất cả các hàng cùng một lúc trong tập kết quả. Con trỏ được sử dụng khi các bản ghi trong bảng cơ sở dữ liệu cần phải được cập nhật một hàng mỗi lần.

Các loại con trỏ

- **Con trỏ tĩnh** – Những con trỏ này giúp chuyển đến tập kết quả khi con trỏ được tạo ra và kết quả truy vấn được lưu trữ. Đây là cái chậm nhất trong tất cả các con trỏ. Con trỏ này có thể di chuyển/ cuộn theo cả hai hướng tiến lùi. Ngoài ra, con trỏ tĩnh không thể được cập nhật hoặc xóa.
- **Con trỏ động** – Những con trỏ này cho phép bạn xem các thủ tục chèn, cập nhật và xóa khi con trỏ được mở ra. Đây là một trong những con trỏ nhạy cảm nhất và có thể cuộn.
- **Con trỏ chỉ chuyển tiếp** – Những con trỏ này còn hỗ trợ cập nhật và xóa dữ liệu. Đây là con trỏ nhanh nhất mặc dù nó không hỗ trợ cuộn về phía sau. The three types of forward cursors are FORWARD _ ONLY KEYSET, FORWARD _ ONLY STATIC, và FAST _ FORWARD.
- **Con trỏ hướng tập khóa** – Những con trỏ này tạo ra một tập hợp các mã định danh duy nhất làm các khóa trong một tập khóa được sử dụng để điều khiển con trỏ. Đây còn là một con trỏ nhạy cảm có thể cập nhật và xóa dữ liệu. Tập khóa phụ thuộc vào tất cả các hàng xác định câu lệnh SELECT tại thời điểm mở con trỏ.

Sau đây là cú pháp để khai báo con trỏ.

Cú pháp:

```
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 con trỏ đã định nghĩa, cursor_name phải tuân thủ những quy tắc cho mã định danh.

LOCAL: chỉ ra rằng con trỏ có thể được sử dụng cục bộ cho khối lệnh, thủ tục lưu trữ, hoặc trigger trong đó con trỏ được tạo ra.

GLOBAL: chỉ ra rằng con trỏ có thể được sử dụng trên toàn cầu cho kết nối đó. **FORWARD_ONLY**: chỉ ra rằng con trỏ chỉ có thể được cuộn từ hàng đầu tiên đến hàng cuối cùng.

STATIC: định nghĩa một con trỏ tạo ra một bản sao tạm thời của dữ liệu để con trỏ sử dụng.

KEYSET: chỉ ra rằng tư cách thành viên và thứ tự của các hàng trong con trỏ là cố định khi con trỏ được mở ra.

DYNAMIC: định nghĩa một con trỏ phản ánh tất cả thay đổi dữ liệu được thực hiện cho những hàng trong tập kết quả của nó khi bạn cuộn xung quanh con trỏ.

FAST_FORWARD: chỉ ra **FORWARD_ONLY**, con trỏ **READ_ONLY** với tối ưu hóa hiệu suất đã cho phép.

READ_ONLY: ngăn chặn các lần cập nhật được thực hiện thông qua con trỏ này.

SCROLL_LOCKS: chỉ ra rằng các lần cập nhật hoặc xóa đã định vị đã thực hiện thông qua con trỏ được đảm bảo sẽ thành công.

Code Snippet 20 tạo ra bảng **Employee** trong cơ sở dữ liệu **SampleDB**.

Code Snippet 20:

```
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, 'Houston')
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
```

Hình 11.29 shows the output of the 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

Hình 11.29: Employee Table Created in SampleDB Database

Code Snippet 21 trình bày cách khai báo con trỏ trên bảng **Employee**.

Code Snippet 21:

```
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
```

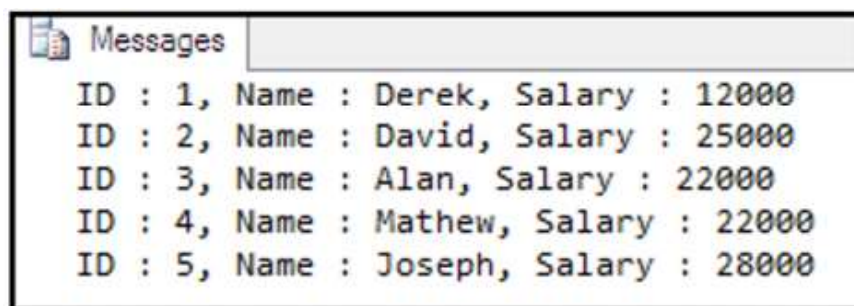
```

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

```

Hình 11.30 trình bày kết quả của đoạn mã.



```

Messages
ID : 1, Name : Derek, Salary : 12000
ID : 2, Name : David, Salary : 25000
ID : 3, Name : Alan, Salary : 22000
ID : 4, Name : Mathew, Salary : 22000
ID : 5, Name : Joseph, Salary : 28000

```

Hình 11.30: Cursor Created on Employee Table

Trong đoạn mã này, những chi tiết được lấy mỗi lần một hàng. Thủ tục này sẽ giúp lấy các cơ sở dữ liệu lớn theo tuần tự.

Đầu tiên, con trỏ được khai báo bằng cách định nghĩa câu lệnh SQL trả về một tập kết quả. Sau đó, nó được mở ra và đặt bằng cách thực hiện câu lệnh SQL được định nghĩa bằng con trỏ này. Sau đó các hàng được lấy từ con trỏ từng cái một hoặc trong một khối để thực hiện thao tác dữ liệu.

Con trỏ sau đó được đóng và cuối cùng, định nghĩa con trỏ sẽ bị xóa và tất cả các tài nguyên hệ thống liên quan đến con trỏ này được giải phóng.

11.7 Kiểm tra tiến độ

1. Điều nào sau đây là mã đúng để định nghĩa một chỉ mục?

(A)	USE CUST_DB CREATE INDEX IX_CountryCustomer Details (Country); GO	(C)	USE CUST_DB CREATE INDEX IX_CountryWith Customer_Details (Country); GO
(B)	USE CUST_DB CREATE INDEX IX_CountryFROMCustomer_ Details (Country); GO	(D)	USE CUST_DB CREATE INDEX IX_CountryON Customer_Details (Country); GO

2. SQL Server 2012 lưu trữ dữ liệu trong các đơn vị lưu trữ được gọi là _____

(A)	các trang dữ liệu	(C)	các bản ghi
(B)	các biểu mẫu	(D)	các cột

3. Đoạn mã nào sau đây di chuyển dữ liệu trong bảng **Supplier _ Details** kết quả tới vị trí mặc định?

(A)	DROP INDEX IX_SuppIDON Supplier_Details	(C)	DROP INDEX IX_SuppIDON Supplier_ Details WITH (MOVE TO 'default')
(B)	MOVE INDEX IX_SuppIDON Supplier_Details WITH ('default')	(D)	DELETE INDEX IX_SuppIDON Supplier_Details WITH ('default')

4. Mã nào sau đây được sử dụng để tạo ra INDEX ghép cụm trên **CustID** trong bảng **Customer _ Details**?

(A)	USE CUST_DB CREATE CLUSTERED INDEX Customer_Details (CustID) GO	(C)	USE CUST_DB CREATE INDEX Customer_Details ON IX_CustID GO
(B)	USE CUST_DB CREATE INDEX IX_CustIDON Customer_Details GO	(D)	USE CUST_DB CREATE CLUSTERED INDEX IX_CustID ON Customer_Details (CustID) GO

5. Chỉ mục ghép cụm có thể được tạo ra trên một bảng sử dụng một cột không có các giá trị _____.

(A)	tương tự	(C)	trong suốt
(B)	trùng lặp	(D)	bất kỳ

11.7.1 Đáp án

1.	D
2.	A
3.	C
4.	D
5.	B

Tóm tắt

- Các chỉ mục tăng tốc độ của quá trình truy vấn bằng cách cung cấp khả năng truy cập nhanh tới các hàng hoặc cột trong một bảng dữ liệu.
- SQL Server 2012 lưu trữ dữ liệu trong các đơn vị lưu trữ được gọi là các trang dữ liệu.
- Tất cả các phép tính đầu vào và đầu ra trong cơ sở dữ liệu này được thực hiện ở mức trang.
- SQL Server sử dụng các khung nhìn danh mục để tìm các hàng khi chỉ mục không được tạo ra trên một bảng.
- Chỉ mục ghép cụm làm cho các bản ghi được lưu trữ vật lý theo thứ tự sắp xếp hoặc tuần tự.
- Chỉ mục không ghép cụm được định nghĩa trên bảng có dữ liệu trong cấu trúc ghép cụm hoặc heap.
- Các chỉ mục XML có thể tăng tốc các truy vấn trên các bảng có dữ liệu XML.
- Chỉ mục lưu trữ cột giúp tăng cường hiệu suất của các truy vấn kho dữ liệu một cách bao quát.

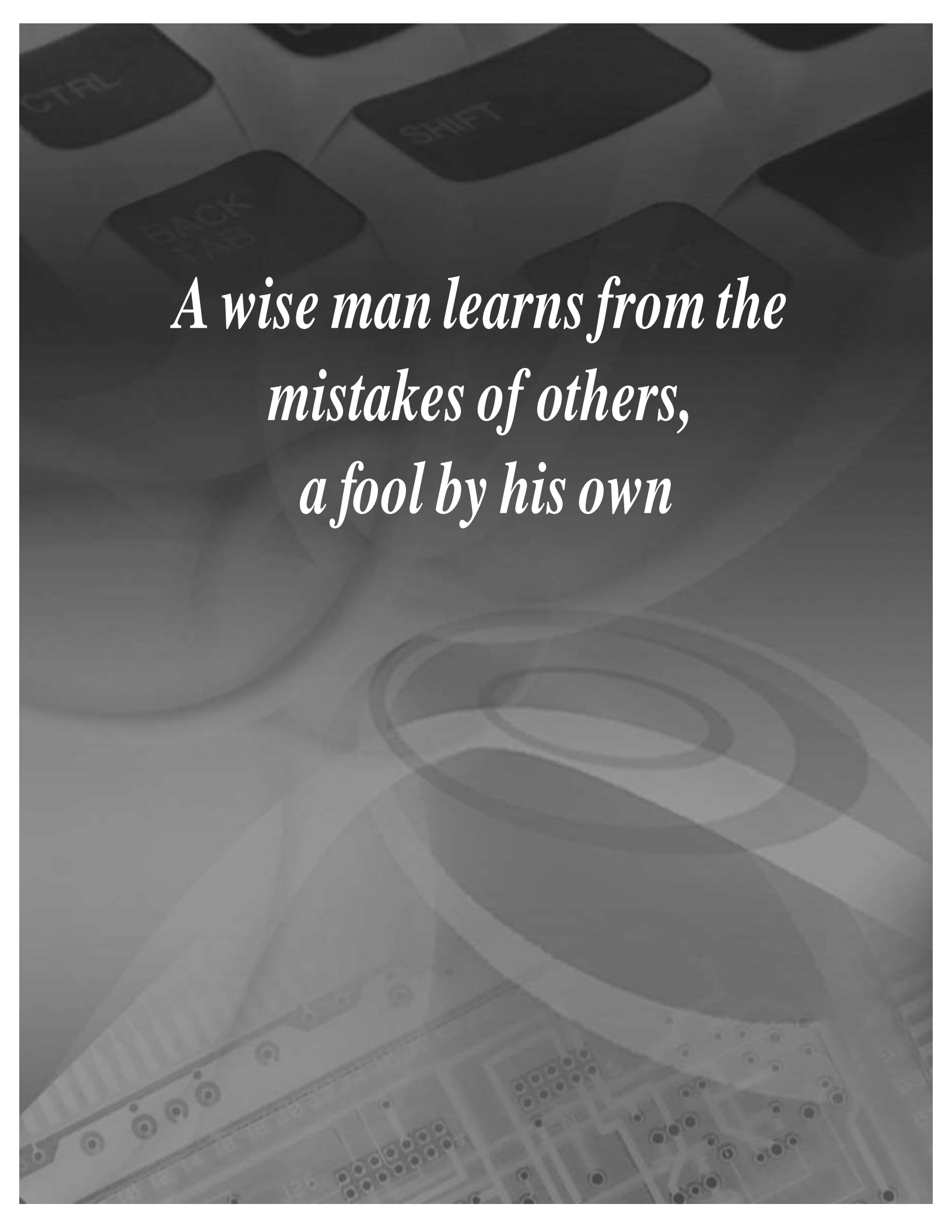


1. Thư viện Tiểu bang Houston là một trong những thư viện nổi tiếng tại Houston, Texas. Thư viện có một lượng lưu kho khoảng 1000000 cuốn sách thuộc các thể loại khác nhau. Thư viện phát hành sách cho học viên của các trường đại học gần đó. Với luồng vào của các học viên đến thư viện ngày càng tăng theo cấp số nhân, Thư viện Tiểu bang Houston đã quyết định để tự động hóa toàn bộ quá trình phát hành sách cho học viên. Thư viện đã tăng số lượng từng đầu sách đến 10 bản, tùy thuộc vào nhu cầu từ các học viên.
 - Hãy tạo ra một cơ sở dữ liệu có tên là **HoustonStateLibrary** để lưu trữ các chi tiết về sách của Thư viện.
 - Tạo ra một bảng có tên là **BooksMaster** để lưu trữ các chi tiết về sách trong thư viện như được trình bày trong bảng 11.3.

Field Name	Data Type	Key Field	Description
BookCode	varchar(50)	Primary Key	Stores book code of the book
Title	varchar(max)		Stores the book title
ISBN	varchar(50)		Stores the ISBN of the book
Author	char(30)		Stores author name of the book
Price	money		Stores price of the book
Publisher	char(30)		Stores publisher name of the book
NumPages	numeric(10,0)		Stores number of pages in the book

Table 11.3: BooksMaster Table

- Tạo ra một chỉ mục ghép cụm có tên là **IX _ Title** trên cột **Title** trong bảng **BooksMaster**.
- Tạo bảng **BooksMaster1** có các tên trường **BookCode**, **Title**, và **Book Details**. Chỉ ra kiểu dữ liệu cho **BookDetails** là xml. Tạo ra một tài liệu XML với các chi tiết về **ISBN**, **Author**, **Price**, **Publisher**, và **NumPages**.
- Thư viện muốn lấy tên nhà xuất bản của công ty in một cuốn sách của tác giả cụ thể. Tạo ra chỉ mục XML chính **PXML _ Books** trên cột **BookCode** của bảng **BooksMaster**.



*A wise man learns from the
mistakes of others,
a fool by his own*