

Bài 5

Index, Stored Procedure, View

Module: JWBD

Thảo luận bài cũ

- Hỏi và trao đổi về các khó khăn gặp phải trong bài “Các hàm thông dụng trong SQL”
- Tóm tắt lại các phần đã học từ bài “Các hàm thông dụng trong SQL”

Mục tiêu

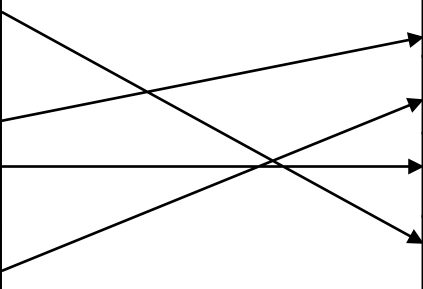
- Trình bày được khái niệm chỉ mục (index)
- Tạo mới, sửa và xóa chỉ mục
- Sử dụng chỉ mục trong truy vấn SQL
- Trình bày được khái niệm khung nhìn (view)
- Tạo mới, sửa và xóa khung nhìn
- Trình bày được khái niệm thủ tục lưu (stored procedure)
- Tạo mới, sửa và xóa thủ tục lưu

Khái niệm chỉ mục - index

- Các bản ghi được lưu trữ vào trong bảng theo đúng thứ tự như khi nhập vào. Do đó có thể dữ liệu này không được sắp xếp.
- Khó khăn: Khi muốn tìm kiếm dữ liệu trên bảng cần phải quét toàn bộ bảng. Điều này làm chậm tốc độ thực thi truy vấn.
- Việc tìm kiếm có thể được trợ giúp nhiều nếu dữ liệu được chứa trong các khối được đánh chỉ mục (index).
- Khi một chỉ mục được tạo ra trên bảng, chỉ mục sẽ tạo nên thứ tự cho các hàng dữ liệu hay các bản ghi trong bảng đó.

Khái niệm chỉ mục – Ví dụ

Emp_No	Emp_No	Emp_Name	Emp_DOB	Emp_DOJ
305	345	James	24-Sep-1968	30-May-1990
345	873	Pamela	27-Jul-1970	19-Nov-1993
693	693	Allan	10-Sep-1970	01-Jul-1992
873	305	Geoff	12-Feb-1973	29-Oct-1996



Chỉ mục – Ưu và nhược điểm

- Ưu điểm:
 - Một chỉ mục cho phép chương trình tìm dữ liệu trong bảng mà không cần duyệt qua toàn bộ bảng.
 - Chỉ mục giúp làm tăng tốc độ thực thi các truy vấn cần nối nhiều bảng hay cần sắp xếp dữ liệu.
 - Một chỉ mục hợp lý có thể cải thiện sự thực thi cơ sở dữ liệu bởi giảm bớt thời gian truy cập.
- Hạn chế:
 - Các bảng có các chỉ mục đòi hỏi nhiều dung lượng bộ nhớ hơn trong CSDL.
 - Các lệnh thao tác dữ liệu đòi hỏi nhiều thời gian xử lý hơn, vì chúng cần cập nhật sự thay đổi tới các chỉ mục.

Hướng dẫn khi tạo chỉ mục

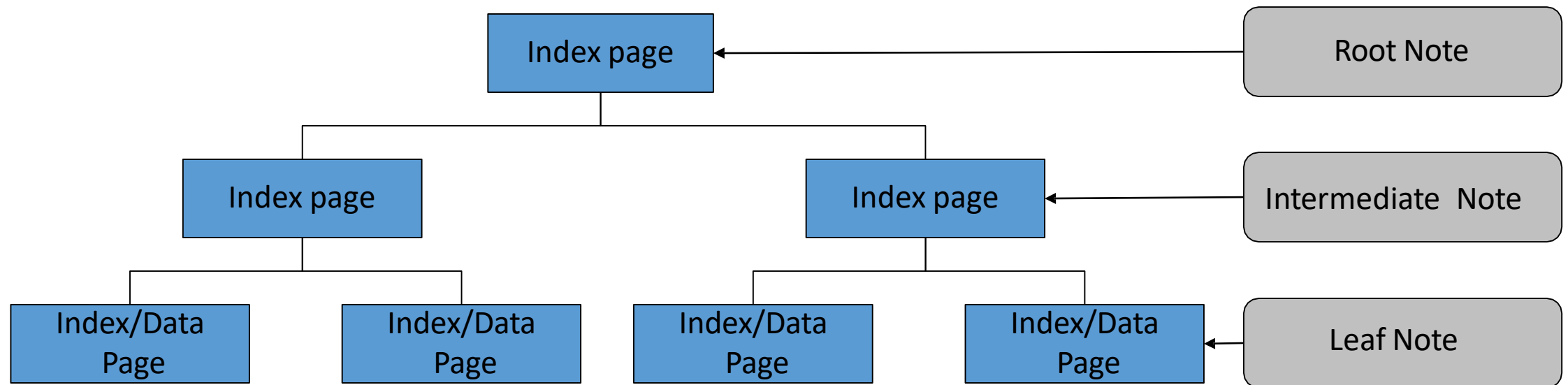
- Cần duy nhất các hàng khi định nghĩa chỉ mục.
- Chỉ mục được tạo và duy trì theo thứ tự sắp xếp tăng hay giảm dần của dữ liệu .
- Một chỉ mục có thể được tạo lập trên một trường hoặc trên nhiều trường của bảng hoặc khung nhìn.
- MySQL tự động tạo lập các chỉ mục cho các kiểu ràng buộc PRIMARY KEY và UNIQUE.

Hướng dẫn khi tạo chỉ mục

- Nên tạo chỉ mục trên các cột khi:
 - Bảng chứa dữ liệu lớn
 - Cột được sử dụng để tìm kiếm thường xuyên
 - Cột được sử dụng để sắp xếp dữ liệu
 - Dữ liệu trong cột có sự phân biệt cao
- Không nên tạo chỉ mục trên các cột khi:
 - Bảng chỉ chứa một vài hàng
 - Cột thường xuyên có các thao tác thêm sửa xóa
 - Cột có sự trùng lặp dữ liệu cao

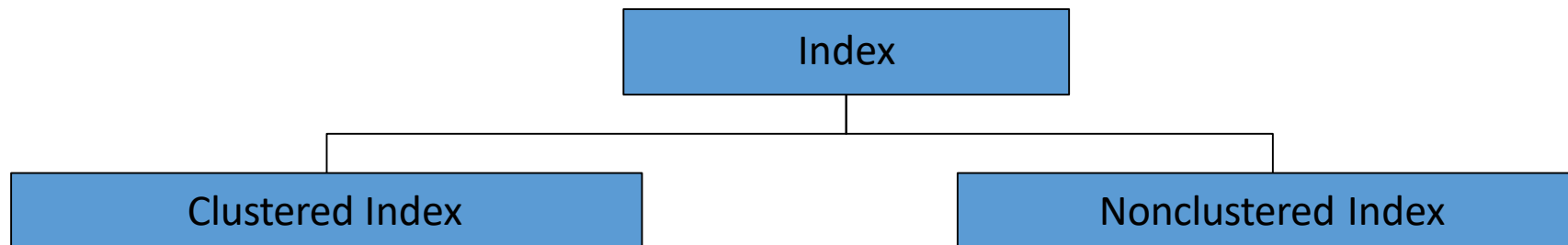
Cấu trúc chỉ mục

- Cây nhị phân chỉ mục



Phân loại chỉ mục

- Chỉ mục phân cụm - Clustered Indexes: xác định thứ tự lưu trữ thực sự của dữ liệu trong bảng. Khi tạo chỉ mục phân cụm trong bảng, dữ liệu trong bảng sẽ được sắp xếp lại về mặt vật lý theo một thứ tự liên tiếp.
- Chỉ mục không phân cụm - Nonclustered Indexes: xác định thứ tự lưu trữ dữ liệu logic của bảng. Chỉ mục không phân cụm sẽ chứa các giá trị khóa và các bộ định vị hàng để chỉ ra vị trí lưu trữ của dữ liệu trong bảng.
- Nên tạo chỉ mục Clustered trước khi tạo chỉ mục Nonclustered để chỉ mục Nonclustered không bị tạo lại sau khi chỉ mục Clustered được tạo.



Phân loại chỉ mục – so sánh

Clustered Indexes	Nonclustered Indexes
Được sử dụng cho các truy vấn trả về số lượng lớn tập kết quả	Được sử dụng cho truy vấn trả về số lượng nhỏ tập kết quả
Chỉ một chỉ mục clustered index có thể được tạo trên một bảng	Nhiều chỉ mục nonclustered index có thể được tạo trên bảng, tối đa là 249
Dữ liệu được sắp xếp vật lý dựa theo khóa clustered	Dữ liệu không được sắp xếp dựa theo khóa nonclustered
Nút lá của chỉ mục clustered chứa trang dữ liệu	Nút lá của chỉ mục nonclustered chứa trang dữ liệu

Cú pháp tạo chỉ mục

- Sử dụng câu lệnh “CREATE INDEX”.
- Cú pháp:

```
CREATE UNIQUE INDEX index_name  
ON table_name ( column1, column2,...);
```

Ví dụ:

```
CREATE UNIQUE INDEX user_email  
ON users (email);
```

Thủ tục lưu trữ - SPs

- Là tập hợp các câu lệnh transact-SQL được xem như một khối lệnh đơn nhằm thực hiện một tác vụ cụ thể.
- Hữu ích cho những tác vụ thực hiện lặp đi lặp lại
- Hỗ trợ các biến do người dùng khai báo, các điều kiện thực thi và các đặc trưng khác
- Ưu điểm:
 - Tăng tính bảo mật
 - Thực thi tiền biên dịch
 - Giảm thiểu lưu thông trong mô hình Client/Server
 - Khả năng sử dụng lại

Tạo thủ tục lưu

- Cú pháp:

```
DELIMITER //  
CREATE PROCEDURE yourProcedureName(IN  
  yourParameterName dataType,OUT  
  yourParameterName dataType  
)  
BEGIN  
  yourStatement1;  
  yourStatement2;  
  .  
  .  
  yourStatementN  
END;  
//  
DELIMITER ;
```

- Ví dụ:

```
DELIMITER //  
  
create procedure sp_ChechValue(IN value1 int,OUT value2 int)  
  
  begin  
  
    set value2=(select Amount from SumOfAll where Amount=value1);  
  
end;  
  
// delimiter ;
```

Tạo thủ tục lưu

- Khi truyền tham số dạng OUT mục đích là lấy dữ liệu trong Procedure và sử dụng ở bên ngoài.
- Khi truyền tham số vào dạng OUT phải có chữ @ đằng trước biến
- Hoạt động giống tham chiếu nên biến truyền vào dạng OUT không cần định nghĩa trước, chính vì vậy khởi đầu nó có giá trị NULL

Cú pháp điều khiển

- `begin ..end` : đánh dấu khối lệnh

```
begin
{statements | statement_block}
end
```
- `if ..else:`

```
if condition_expression
{statements | statement_block}
else
{statements | statement_block}
```


Khái niệm về khung nhìn - view

- Một view là một bảng ảo được tạo ra từ việc tập hợp các cột của một hoặc nhiều bảng
- Một view có thể bao gồm các cột của một view khác
- Dữ liệu trong view được tập hợp từ các bảng cơ sở tham chiếu đến khi định nghĩa view

Beam Forces : Table				
	Story	Beam	Load	Loc
►	STORY4	B2	TH3 MAX	0.15
	STORY4	B2	TH3 MAX	0.4909091
	STORY4	B2	TH3 MAX	0.4909091
	STORY4	B2	TH3 MAX	0.9818182
	STORY4	B2	TH3 MAX	0.9818182
	STORY4	B2	TH3 MAX	1.472727

View

	Story	Beam
►	STORY4	B2
	STORY4	B2
	STORY4	B2
	STORY4	B2

Ưu điểm của view

- Bảo mật thông qua quyền truy cập cá nhân
- Tùy biến việc hiển thị dữ liệu
- Kết hợp dữ liệu từ các bảng hay các view
- Các thao tác trên bản ghi
- Toàn vẹn ràng buộc kiểm tra
- Với người sử dụng cuối:
 - Dễ dàng để hiểu được kết quả
 - Dễ dàng thu được dữ liệu mong muốn
- Với nhà phát triển:
 - Dễ dàng hạn chế việc nhận dữ liệu trả về
 - Dễ dàng bảo trì ứng dụng

Phân loại view

- Standard View: Sử dụng các cột từ một hoặc nhiều bảng
- Indexed View: View được tạo chỉ mục unique clustered index
- Partitioned View: View được tạo bằng cách kết hợp dữ liệu đã được chia ra của các bảng từ một hoặc nhiều máy chủ

View hệ thống

- Các thuộc tính của một đối tượng như bảng hay view được lưu trữ vào một bảng hệ thống đặc biệt là các siêu dữ liệu – metadata
- Siêu dữ liệu này gọi là khung nhìn hệ thống
- Các view hệ thống được chèn vào tự động trong cơ sở dữ liệu do người dùng tạo
- Một số view hệ thống:
 - Catalog Views
 - Information Schema Views
 - Compatibility Views
 - Replication Views
 - Dynamic Management Views
 - Notification Services Views

Hướng dẫn khi tạo view

- View chỉ có thể được tạo trong CSDL hiện tại.
- Một View có thể được xây dựng dựa vào các view khác và có thể lồng vào nhau tới 32 mức. Mỗi View có thể bao gồm tới 1024 cột.
- Những giá trị mặc định, những quy tắc và bất lỗi không thể được liên kết với View.
- Câu truy vấn để khai báo View không thể bao gồm các mệnh đề ORDER BY, COMPUTE, COMPUTE BY hoặc từ khóa INTO

Tạo view

- Cú pháp:

```
CREATE VIEW view_name AS  
SELECT column1, column2.....  
FROM table_name  
WHERE [condition];
```

- Ví dụ:

```
CREATE VIEW CUSTOMERS_VIEW AS  
SELECT name, age  
FROM CUSTOMERS
```

Xóa view

- Việc xóa một View không tác động đến các bảng cơ sở mà View tham chiếu đến
- Cú pháp:
`DROP VIEW view_name`
- Ví dụ:
`DROP VIEW CUSTOMERS`

Tùy chọn khi CREATE VIEW

- Tùy chọn CHECK OPTION
- Tùy chọn SCHEMABINDING
- Tùy chọn ENCRYPTION

Tùy chọn “CHECK OPTION”

- Được sử dụng để đảm bảo rằng nếu bạn muốn sửa hoặc thêm dữ liệu thông qua View thì những dữ liệu đó phải thỏa mãn tất cả các điều kiện trong câu lệnh Select khi định nghĩa view

- **Cú pháp:**

```
CREATE [OR REPLACE VIEW] view_name
```

```
AS
```

```
    select_statement
```

```
    WITH CHECK OPTION;
```

- **Ví dụ:**

```
CREATE VIEW CustDetail
```

```
AS
```

```
SELECT* FROM Customer WHERE City in ('Hanoi','Hai phong', 'HCM')
```

```
WITH CHECK OPTION;
```

➤ UPDATE CustDetail SET City='Quang Ninh' WHERE CustID='c001'

➤ **not Execute**

“UPDATE” view

- Dữ liệu trong một hàng của view có thể được cập nhật bằng câu lệnh UPDATE
- Khi một view được cập nhật, bảng cơ sở được cập nhật tương ứng

- **Cú pháp:**

UPDATE <view_name>

SET column1=value1,....

WHERE <search_condition>

“DELETE” view

- Các hàng của view có thể xóa bằng cách sử dụng câu lệnh DELETE
- Khi các hàng được xóa trong view thì các hàng tương ứng cũng được xóa từ bảng cơ sở.
- **Cú pháp:**
DELETE FROM <view_name>
WHERE <search_condition>

Tóm tắt bài học

- Trình bày được khái niệm chỉ mục (index)
- Tạo mới, sửa và xóa chỉ mục
- Sử dụng chỉ mục trong truy vấn SQL
- Trình bày được khái niệm khung nhìn (view)
- Tạo mới, sửa và xóa khung nhìn
- Trình bày được khái niệm thủ tục lưu (stored procedure)
- Tạo mới, sửa và xóa thủ tục lưu

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: CSS



R a i s i n g t h e b a r