Index Trong Microsoft SQL Server

Vũ Huy Tâm

SQL Seminar Hè 2011 - SQLViet blog

Hà nội 30/07/2011

Nội dung

- Cơ bản về index
- Các loại index
- So sánh tính năng từng loại index
- Nghỉ giải lao
- Tối ưu hóa sử dụng index
- Các kỹ thuật sử dụng index nâng cao
- Bảo trì index
- Kết luận

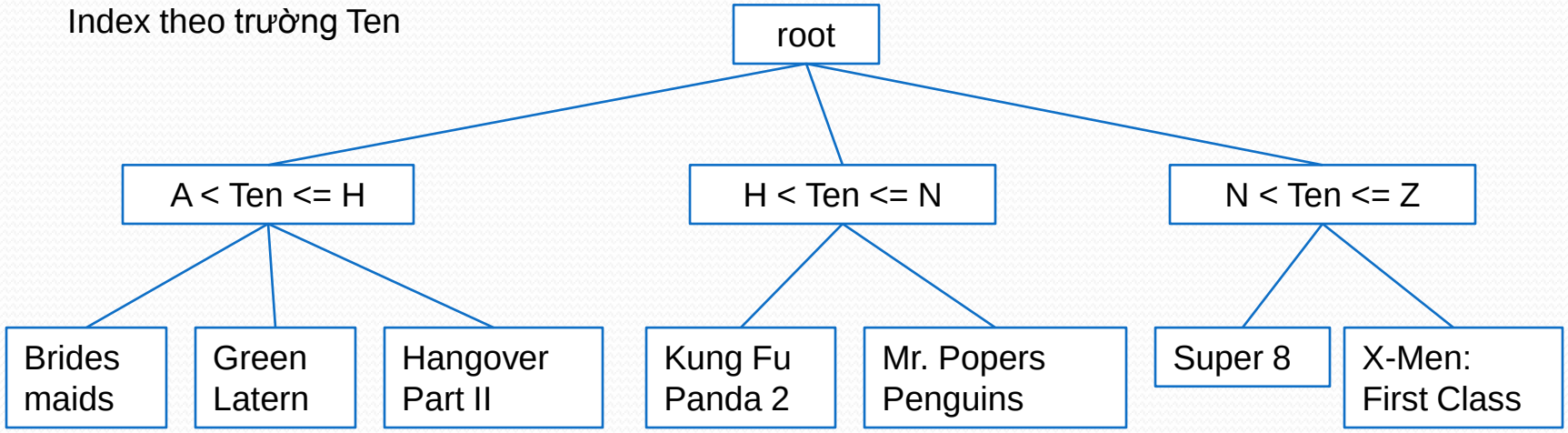
Cơ bản về index

- Tương tự như mục index ở cuối mỗi quyển sách
- Mục đích: tạo shortcut đến dữ liệu cần tìm
- Có cấu trúc dữ liệu dạng B-Tree
  - Khoảng cách từ gốc đến mọi node lá tương đương nhau

Ví dụ: thống kê doanh thu phim tại Mỹ tuần 17/6

FilmID	Ten	Doanh thu
1	Green Latern	$53.2M
2	Super 8	$73M
3	Mr. Popers Penguins	$18.4M
4	X-Men: First Class	$120M
5	Hangover Part II	$233M
6	Kung Fu Panda 2	$144M
7	Bridesmaids	$136M

Index theo trường Ten



Cơ bản về index (Cont'd)

- Tại sao cần index
  - Nâng cao hiệu năng thực hiện câu lệnh
  - Giảm khóa trên bảng
  - Thực thi ràng buộc unique constraint
- Hai loại index trình bày hôm nay:
  - Clustered index và nonclustered index
- Các loại index không được đề cập: XML index, spatial index, fulltext index

Clustered index

- Sắp xếp bảng theo thứ tự của khóa index
- Toàn bộ bảng trở thành cây index. Các node lá chứa khóa index và đồng thời chứa tất cả các trường còn lại

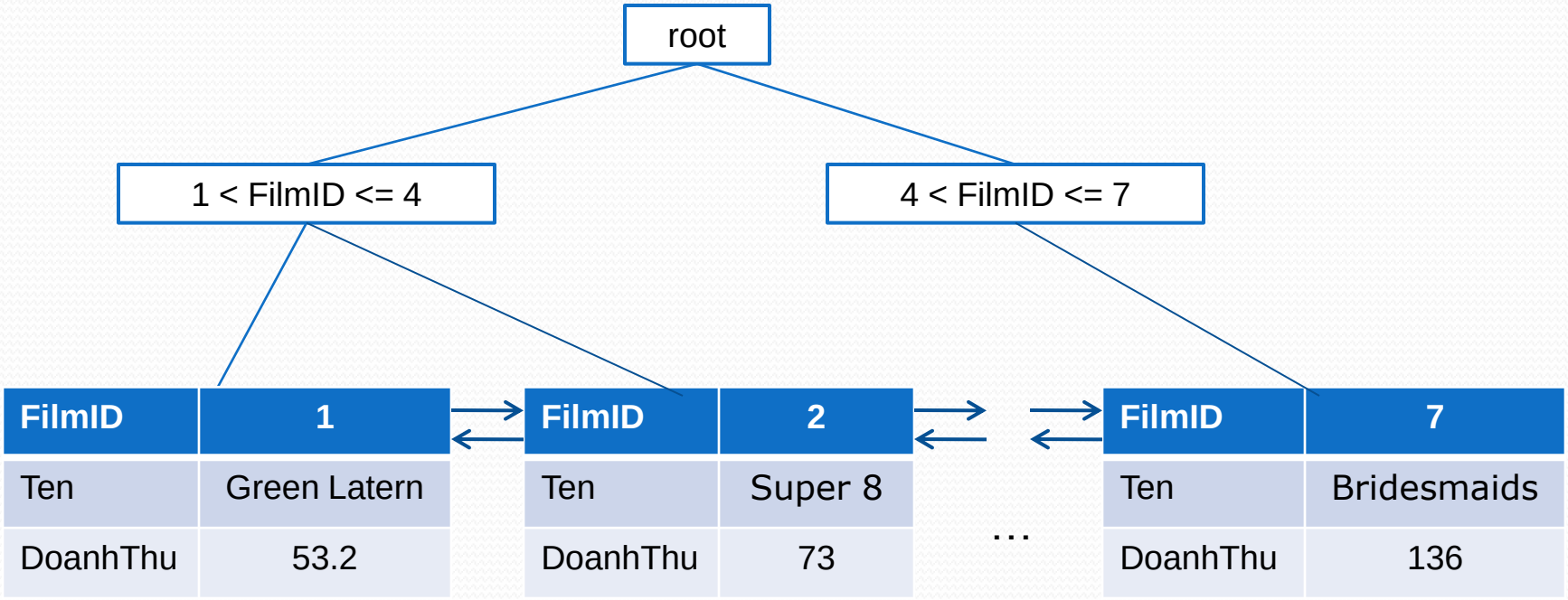
Clustered index (Cont'd)

Ví dụ

FilmID	Ten	Doanh thu
1	Green Latern	$53.2M
2	Super 8	$73M
3	Mr. Popers Penguins	$18.4M
4	X-Men: First Class	$120M
5	Hangover Part II	$233M
6	Kung Fu Panda 2	$144M
7	Bridesmaids	$136M

Clustered index (Cont'd)

Clustered Index theo trường FilmID



Clustered index (Cont'd)

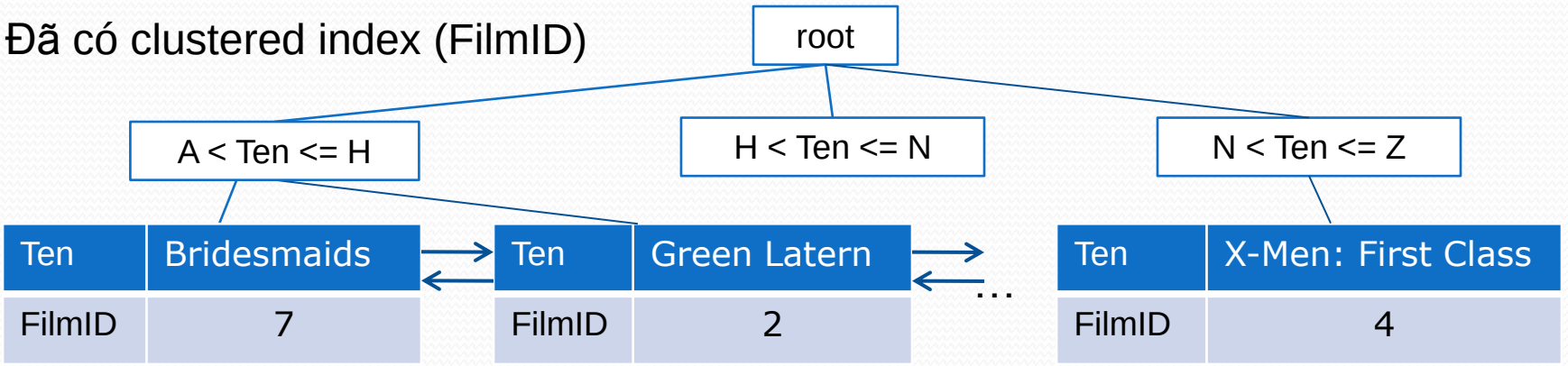
- Chỉ có thể tối đa một clustered index cho mỗi bảng
- Clustered index có thể chứa một hoặc nhiều trường
- Khi tạo Primary Key, một cách mặc định clustered index được tạo kèm với nó
  - Đây là cách thông thường để tạo clustered index

Nonclustered index

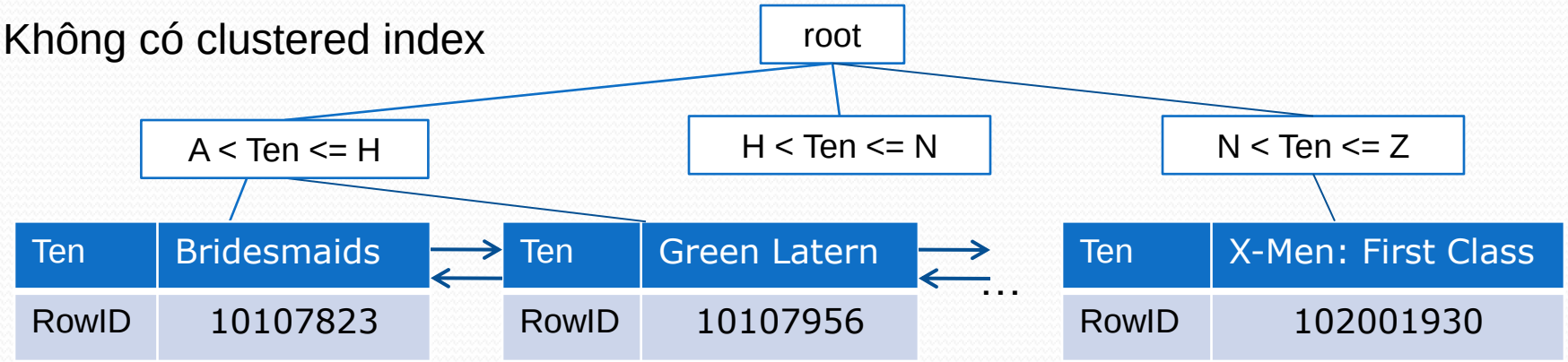
- Mỗi node lá chứa khóa index và con trỏ trỏ đến trang dữ liệu chứa bản ghi tương ứng
  - Nếu bảng có clustered index, con trỏ này chính là khóa clustered index
  - Nếu bảng không có clustered index, con trỏ này là RowID, một dạng định danh bản ghi kết hợp của fileID + pageID + offset
- Bảng có thể có nhiều nonclustered index
- Index có thể chứa một hoặc nhiều trường
- Được lưu trữ tách rời khỏi bảng

Nonclustered index (Cont'd)

Đã có clustered index (FilmID)



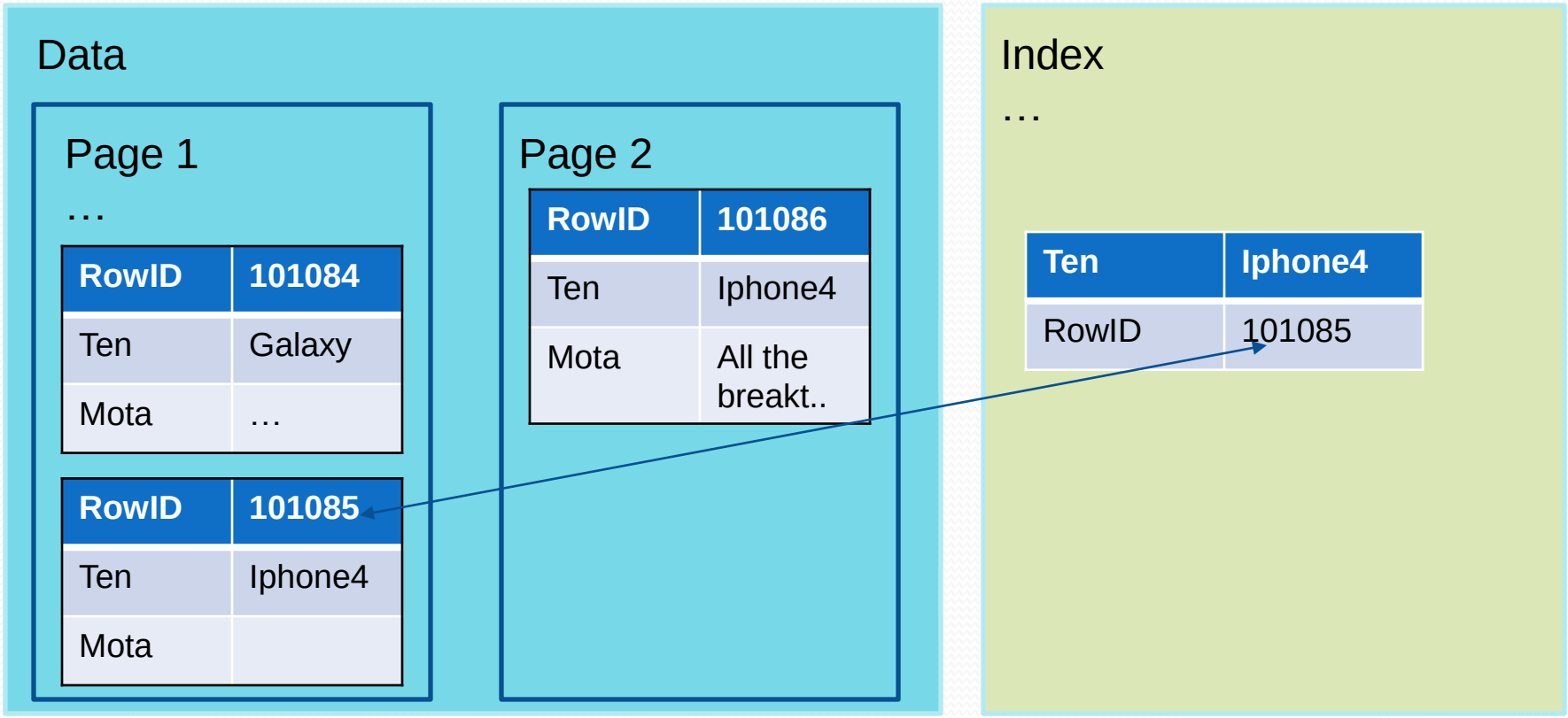
Không có clustered index



Clustered vs. Nonclustered

- Clustered index:
  - Tránh bookmark lookup
  - Nâng cao độ ổn định cho nonclustered index
  - Chỉ được phép tạo một clustered index
- Non-clustered index:
  - bookmark lookup ⇨ giảm hiệu năng
  - Cho phép tạo nhiều index trên bảng
  - Lưu trữ độc lập với bảng ⇨ tăng khả năng xử lý song song

Index bị xáo trộn khi cập nhật

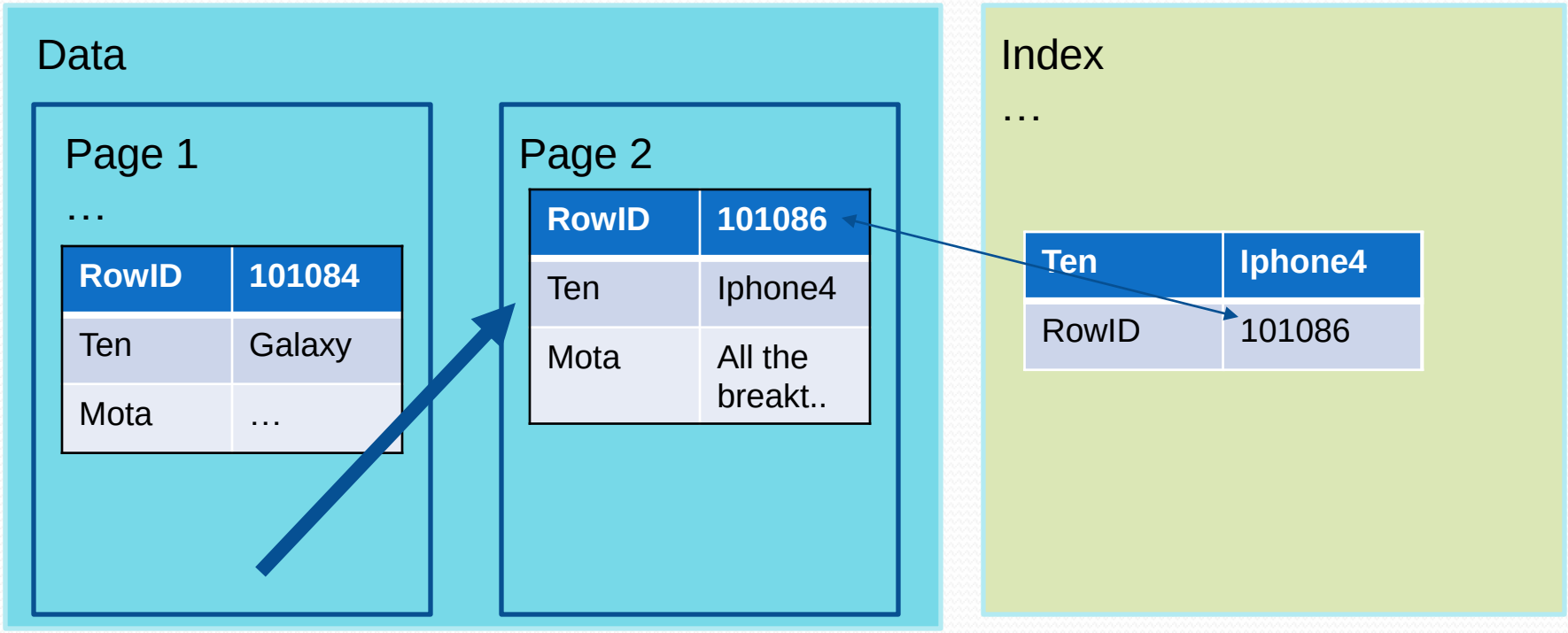


- Trường Mota của bản ghi Iphone4 trống, bản ghi vẫn lưu trữ đủ trong page 1
- Node index của Iphone4 chứa con trỏ là RowID hiện tại của bản ghi Iphone4

Index bị xáo trộn khi cập nhật

```
UPDATE SanPham
SET Mota =  'All the breakthrough technology in iPhone 4 is…'
WHERE Ten = 'Iphone4'
```

Index bị xáo trộn khi cập nhật



- Kích thước bản ghi Iphone4 tăng, vượt quá không gian còn trống của trang. Bản ghi được chuyển sang trang mới
- Node index cũng phải cập nhật con trỏ theo

Unique và non-unique index

- unique (duy nhất) và non-unique (không duy nhất) là các thuộc tính của index
- Mỗi index đều có thể unique hoặc non-unique
  - Tuy nhiên, clustered index thường là unique
  - Khi clustered index không unique, mỗi node được gắn thêm một chuỗi 4 byte (thực chất là một số INT dương) để trở thành unique.
  - Khi đó số bản ghi tối đa cho bảng ~ 2 tỷ
- Khi khai báo ràng buộc unique constraint, một unique index được tạo để thực thi ràng buộc này

Unique và non-unique index

	UNIQUE	NON-UNIQUE
CLUSTERED	✓	✓☹
NON-CLUSTERED	✓	✓

Index seek và Index scan

- Index seek: khi hệ thống có thể nhảy thẳng đến node cần tìm
  - Đây là thao tác tối ưu
- Index scan: khi hệ thống cần quét cả cây index để lấy ra các node cần tìm
  - Không tối ưu bằng index seek, nhưng tốt hơn table scan

Chọn cột đánh index

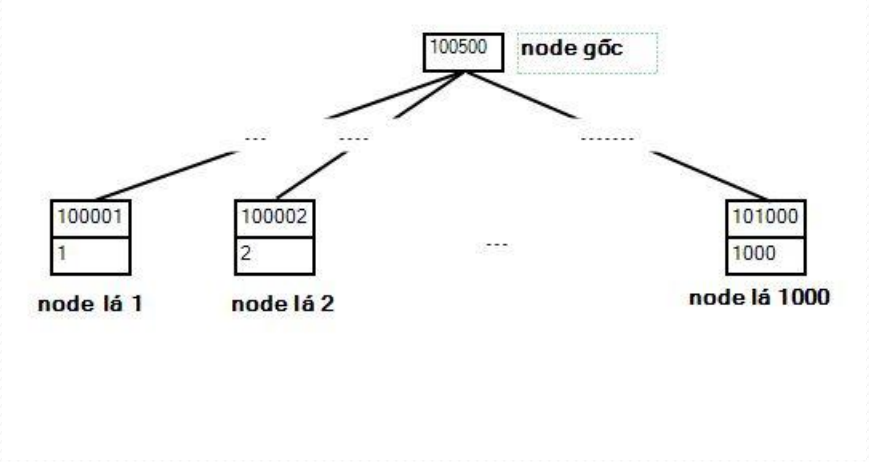
- Cột là ứng cử viên tốt cho index khi:
  - Được sử dụng thường xuyên trong điều kiện tìm kiếm (mệnh đề WHERE)
  - Được sử dụng trong điều kiện JOIN hai bảng
  - Độ lựa chọn (selectivity) đủ cao
- Ưu tiên clustered index cho cột:
  - Tăng tuần tự
  - Kích thước không quá lớn
  - Được tìm kiếm với tần suất cao
  - Thường được tìm kiếm theo dải giá trị

Độ lựa chọn

Độ lựa chọn = Số giá trị khác biệt / Số bản ghi

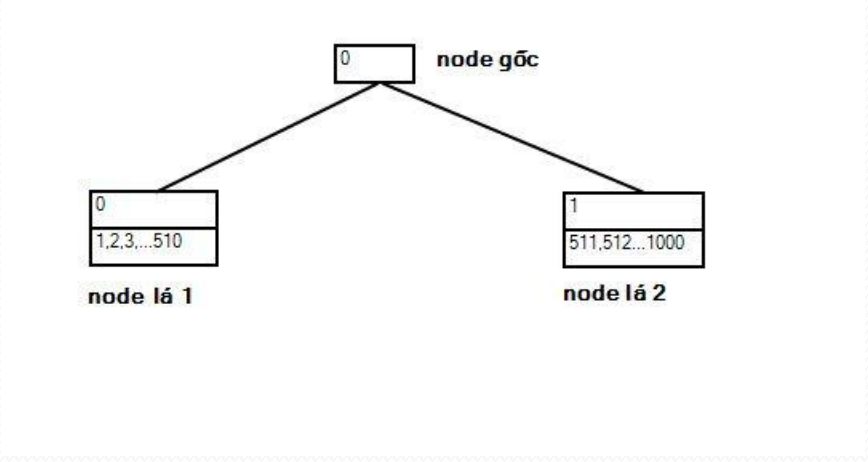
Độ lựa chọn cao

Cây index trên trường số CMT



Độ lựa chọn thấp

Cây index trên trường giới tính



Để index được sử dụng

- Độ lựa chọn (selectivity) đủ cao
  - Với nonclustered index, khi selectivity quá thấp bộ Optimizer bỏ qua index do chi phí lớn
- Tránh chuyển đổi kiểu dữ liệu (type conversion)
- Tránh áp dụng hàm lên cột index
- Cột đầu tiên trong khóa index phải được sử dụng cho tìm kiếm (với composite index)

Index với lệnh JOIN

- Index giúp giảm không gian tìm kiếm ⇨ chọn thuật toán hiệu quả hơn

Các kỹ thuật index nâng cao

- Covering index
  - Lưu thêm các cột dữ liệu vào node index
  - Giúp tránh truy nhập vào bảng để lấy dữ liệu
- Filtered index
  - Index cho một số bản ghi nhất định
- Index intersection
  - Nhiều index cùng tham gia lọc dữ liệu
- Di chuyển index sang filegroup khác với bảng
  - Đọc index và đọc bảng diễn ra song song

Bảo trì index

- Các thao tác cập nhật (INSERT/UPDATE/DELETE) làm index bị phân mảnh
- Hai dạng phân mảnh:
  - Hai node kế tiếp không được lưu trữ liền kề nhau
  - Trang (page) chứa nhiều không gian trống
- Phân mảnh làm tăng số trang cần đọc cho cùng lượng dữ liệu ⇨ giảm hiệu năng truy vấn
- Thông tin về phân mảnh:

```
sys.dm_db_index_physical_stats
```

Bảo trì index (Cont'd)

- Rebuild và Reorganize index:
  - Dùng để cấu trúc lại index, do đó giảm thiểu được phân mảnh
  - Re-org thao tác nhanh hơn nhưng không hiệu quả khi index bị phân mảnh nặng
  - Khi độ phân mảnh <=30% ⇨ REORGANIZE
  - Khi độ phân mảnh > 30% ⇨ REBUILD

Bảo trì index

- Cập nhật Statistics:
  - Statistics chứa thông tin về phân bố dữ liệu của cột ⇨ giúp Optimizer chọn phương án thực thi thích hợp
  - Sau quá trình cập nhật dữ liệu, statistics bị outdated dẫn đến Optimizer chọn phương án sai
- Thống kê về sử dụng index:

```
sys.dm_db_index_usage_stats
```

Kết luận

- Index là công cụ quan trọng trợ giúp các truy vấn vào database
  - Phần lớn sự cố về performance liên quan đến index
- Mặt trái của index :
  - Chiếm không gian đĩa
  - Tăng chi phí của các thao tác cập nhật dữ liệu
- Hệ thống OLTP:
  - Cần điều hòa các lợi ích
  - Index có chọn lọc
- Hệ thống Data warehouse: index tự do hơn

Cám ơn các bạn đã đến dự!

www.sqlviet.com/blog