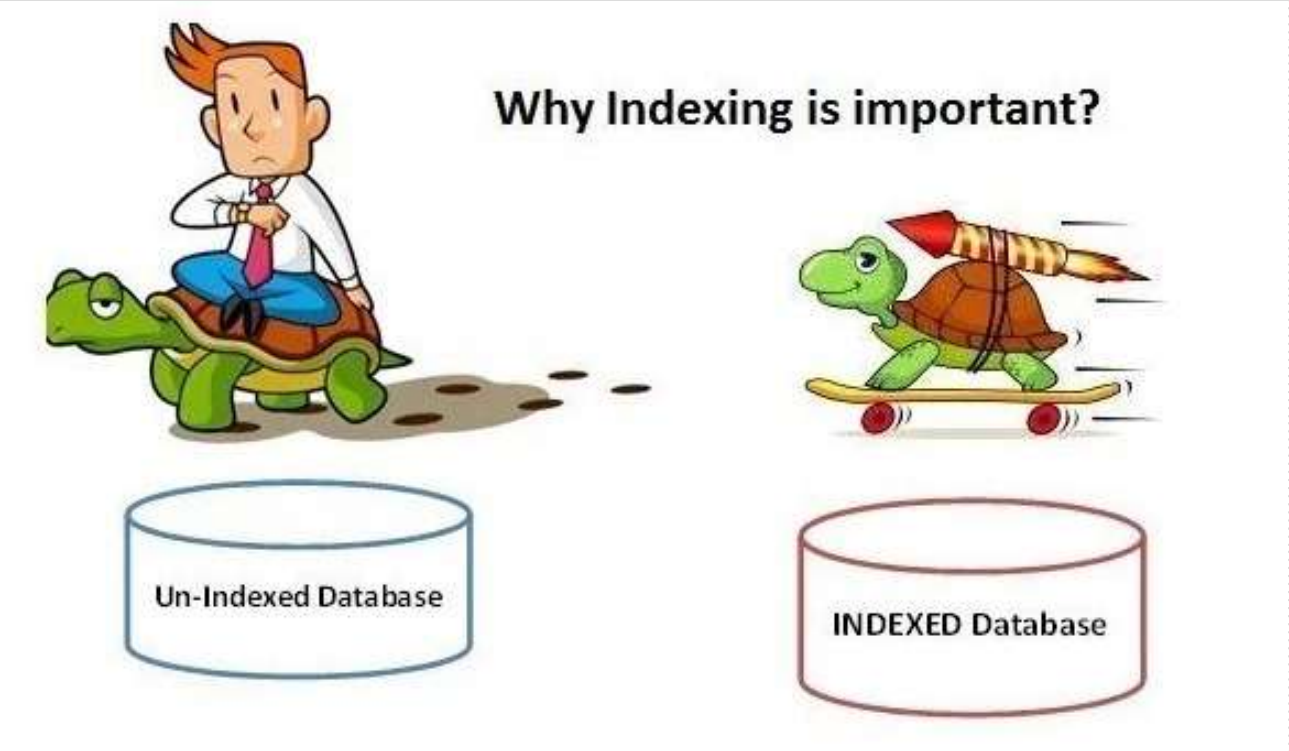
Index

NGUYEN HongPhuong
Email: phuongnh@soict.hust.edu.vn
Site: http://users.soict.hust.edu.vn/phuongnh
Face: https://www.facebook.com/phuongnhbk
Hanoi University of Science and Technology

Nội dung

- Index là gì?
- Index database dùng để làm gì?
- Cấu trúc của index
- Các kiểu index
- Dùng index database như thế nào cho hiệu quả?


Why Indexing is important?
Un-Indexed Database
INDEXED Database

Index là gì?

- Một cấu trúc dữ liệu được dùng để định vị và truy cập nhanh nhất vào dữ liệu trong các table hoặc view.
- Một cách tăng hiệu suất truy vấn database bằng việc giảm lượng truy cập vào bộ nhớ khi thực hiện truy vấn
- SQL Server cung cấp 2 loại index
  - Clustered
  - Non-clustered

Index database dùng để làm gì?

- Truy vấn: SELECT * FROM student WHERE last_name = 'May'
- Nếu không có index cho cột last_name, thì hệ thống sẽ quét qua tất cả các row của bảng student để so sánh và lấy ra row thỏa mãn

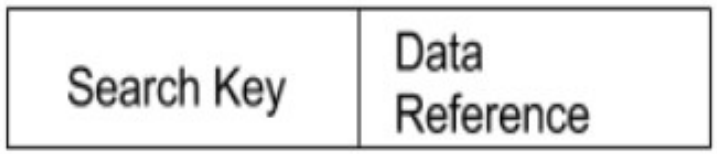
student

student_id	first_name	last_name	dob	gender	address	note	clazz_id
1234	David	Beckham	12/21/1997	Male	London, UK		1
1238	Theresa	May	08/06/1998	Female	London, UK		1
1452	David	Cameron	07/06/1997	Male	Bangor, UK		1
1497	Tony	Blair	03/01/1999	Male	Bath, UK		2
1516	John	Major	03/01/1998	Male	Bradford		2
1542	Margaret	Thatcher	05/08/1997	Female	Cambridge		2

- Index trỏ tới địa chỉ dữ liệu trong một bảng, tương tự mục lục của quyển sách, giúp truy vấn trở nên nhanh chóng
- Index có thể được tạo cho một hoặc nhiều cột trong bảng. Index thường được tạo mặc định cho primary key, foreign key. Ngoài ra, cũng có thể tạo thêm index cho các cột nếu cần.

Cấu trúc của index

- Index gồm:
  - Cột Search Key: chứa bản sao các giá trị của cột được tạo Index
  - Cột Data Reference: chứa con trỏ trỏ đến địa chỉ của bản ghi có giá trị cột index tương ứng



Structure of an index

Các kiểu index

- B-tree
- Hash

B-tree

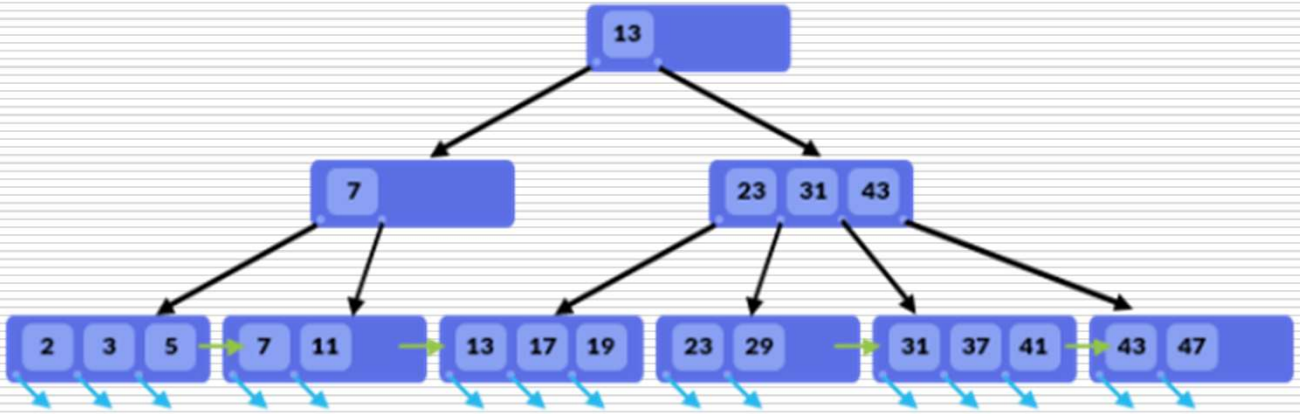
- Thông thường, không chỉ rõ loại index thì mặc định là sử dụng B-Tree.
- Cú pháp:
  - Tạo index
    - CREATE INDEX id_index ON table_name (column_name[, column_name…]) USING BTREE;
    - ALTER TABLE table_name ADD INDEX id_index (column_name[, column_name…])
  - Xóa index
    - DROP INDEX index_name ON table_name

B-tree

- Các đặc điểm của B-Tree Index:
  - Dữ liệu index được tổ chức và lưu trữ theo dạng tree, tức là có root, branch, leaf.
  - Giá trị của các node được tổ chức tăng dần từ trái qua phải.
  - B-tree index được sử dụng trong các biểu thức so sánh dạng: =, >, >=, <, <=, BETWEEN và LIKE. ⇒ Có thể tốt cho câu lệnh ORDER BY

B-tree

- Khi tìm kiếm dữ liệu, sẽ không scan toàn bộ bảng để tìm dữ liệu. Việc tìm kiếm trong B-Tree là 1 quá trình bắt đầu từ root node và tìm kiếm tới branch và leaf, đến khi tìm được tất cả dữ liệu thỏa mãn với điều kiện truy vấn thì dừng lại.



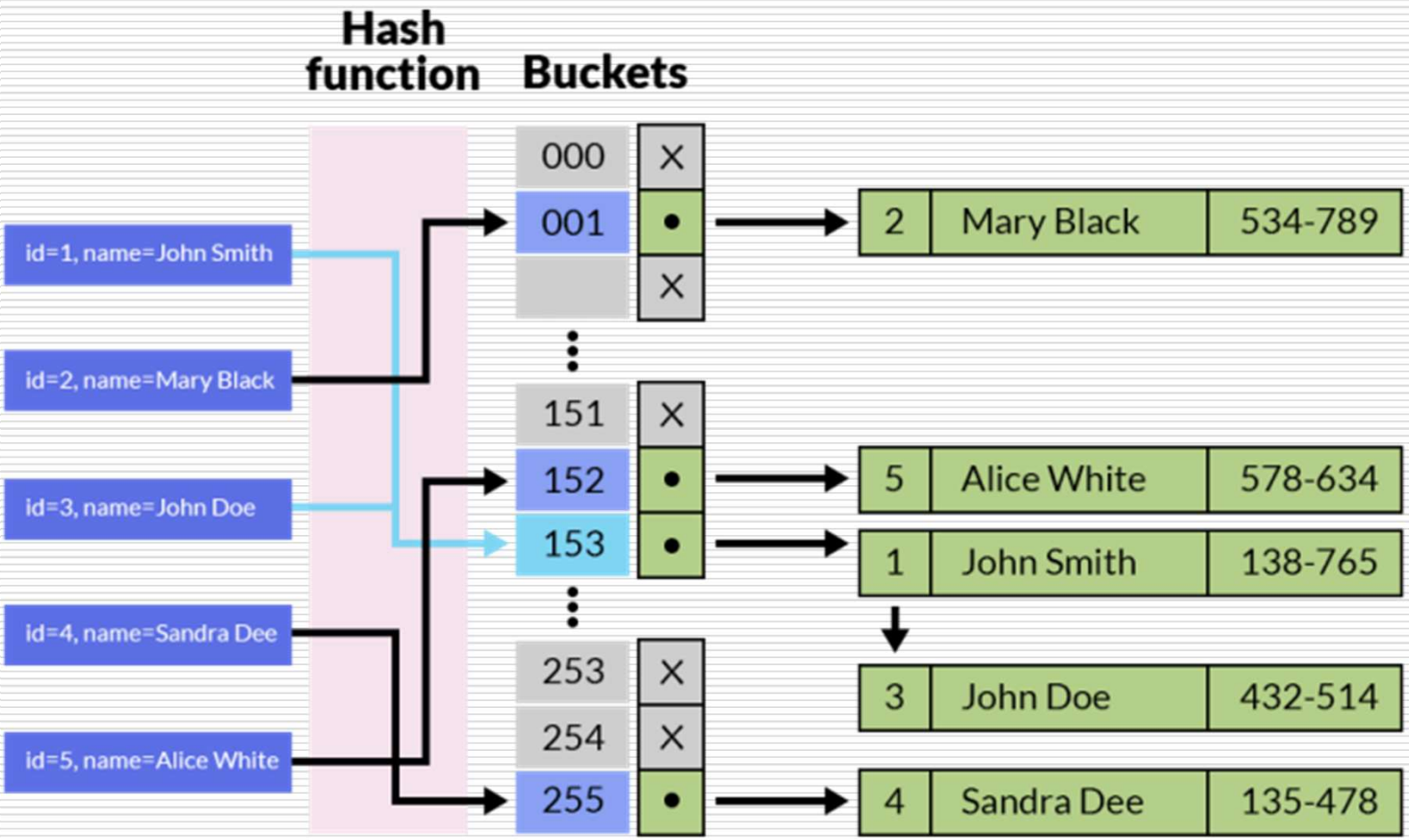
Hash

- Hash index dựa trên giải thuật Hash Function (hàm băm). Tương ứng với mỗi khối dữ liệu (index) sẽ sinh ra một bucket key(giá trị băm) để phân biệt.
- Cú pháp:
  - Create index
    - CREATE INDEX id_index ON table_name (column_name[, column_name…]) USING HASH;
    - ALTER TABLE table_name ADD INDEX id_index (column_name[, column_name…]) USING HASH;

Hash

- Các đặc điểm của Hash Index:
  - Hash index chỉ nên sử dụng trong các biểu thức toán tử là = và <>. Không sử dụng cho toán từ tìm kiếm 1 khoảng giá trị như > hay < .
  - Không thể tối ưu hóa toán tử ORDER BY bằng việc sử dụng Hash index bởi vì nó không thể tìm kiếm được phần từ tiếp theo trong Order.
  - Hash có tốc độ nhanh hơn kiểu B-Tree.

Hash


Hash function
Buckets
id=1, name=John Smith
id=2, name=Mary Black
id=3, name=John Doe
id=4, name=Sandra Dee
id=5, name=Alice White
000
001
151
152
153
253
254
255
2
Mary Black
534-789
5
Alice White
578-634
1
John Smith
138-765
3
John Doe
432-514
4
Sandra Dee
135-478

Storage Engine

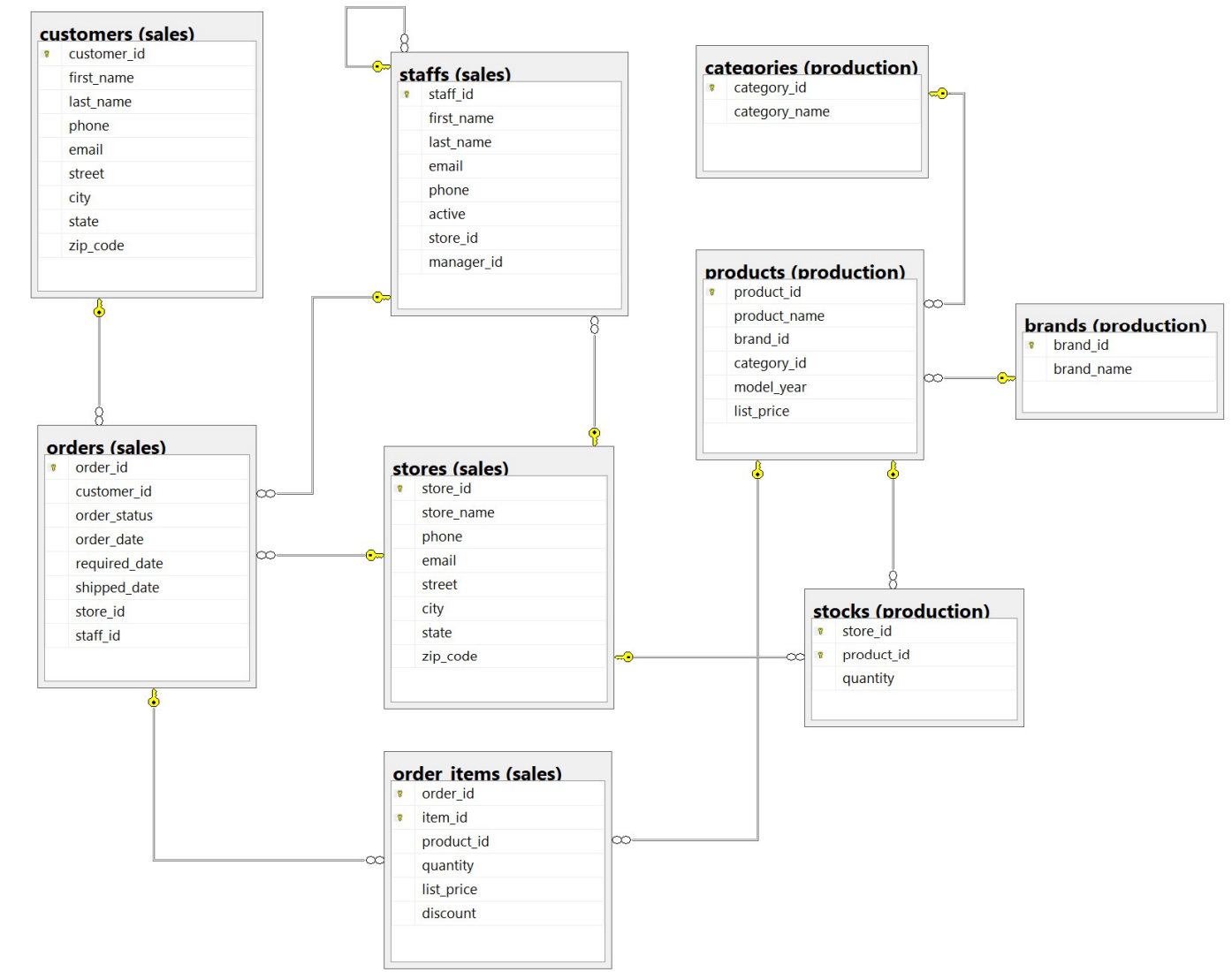
- Việc chọn index theo kiểu B-Tree hay Hash, ngoài mục đích sử dụng, thì nó còn phụ thuộc vào Storage Engine có hỗ trợ loại index đó hay không.
- Storage Engine và các kiểu index được hỗ trợ
  - InnoDB BTREE
  - MyISAM BTREE
  - MEMORY/HEAP HASH, BTREE
  - NDB HASH, BTREE

Dùng Index Database thế nào cho hiệu quả?

- Nên index những cột được dùng trong WHERE, JOIN và ORDER BY
- Không dùng index trong các trường hợp:
  - Các bảng nhỏ, chứa ít dữ liệu
  - Các bảng được update và insert dữ liệu thường xuyên
  - Các cột có quá nhiều giá trị NULL
  - Các cột thường xuyên được update

Thực hành với SQL Server

- CSDL mẫu BikeStores



Tạo bảng mới production.parts

```
CREATE TABLE production.parts(
    part_id   INT NOT NULL,
    part_name VARCHAR(100)
);


INSERT INTO
    production.parts(part_id, part_name)
VALUES
    (1,'Frame'),
    (2,'Head Tube'),
    (3,'Handlebar Grip'),
    (4,'Shock Absorber'),
    (5,'Fork');
```

- Bảng parts không có PK, nên các bản ghi được lưu trữ theo cấu trúc có thứ tự gọi là heap (đống)
- Câu lệnh tìm bản ghi có id là 5
- Xem ước lượng kế hoạch thực hiện trong SQL Server Management Studio
  - Chọn Display Estimated Execution Plan (nhấn Ctrl + L)

```
SELECT
    part_id, part_name
FROM
    production.parts
WHERE
    part_id = 5;
```

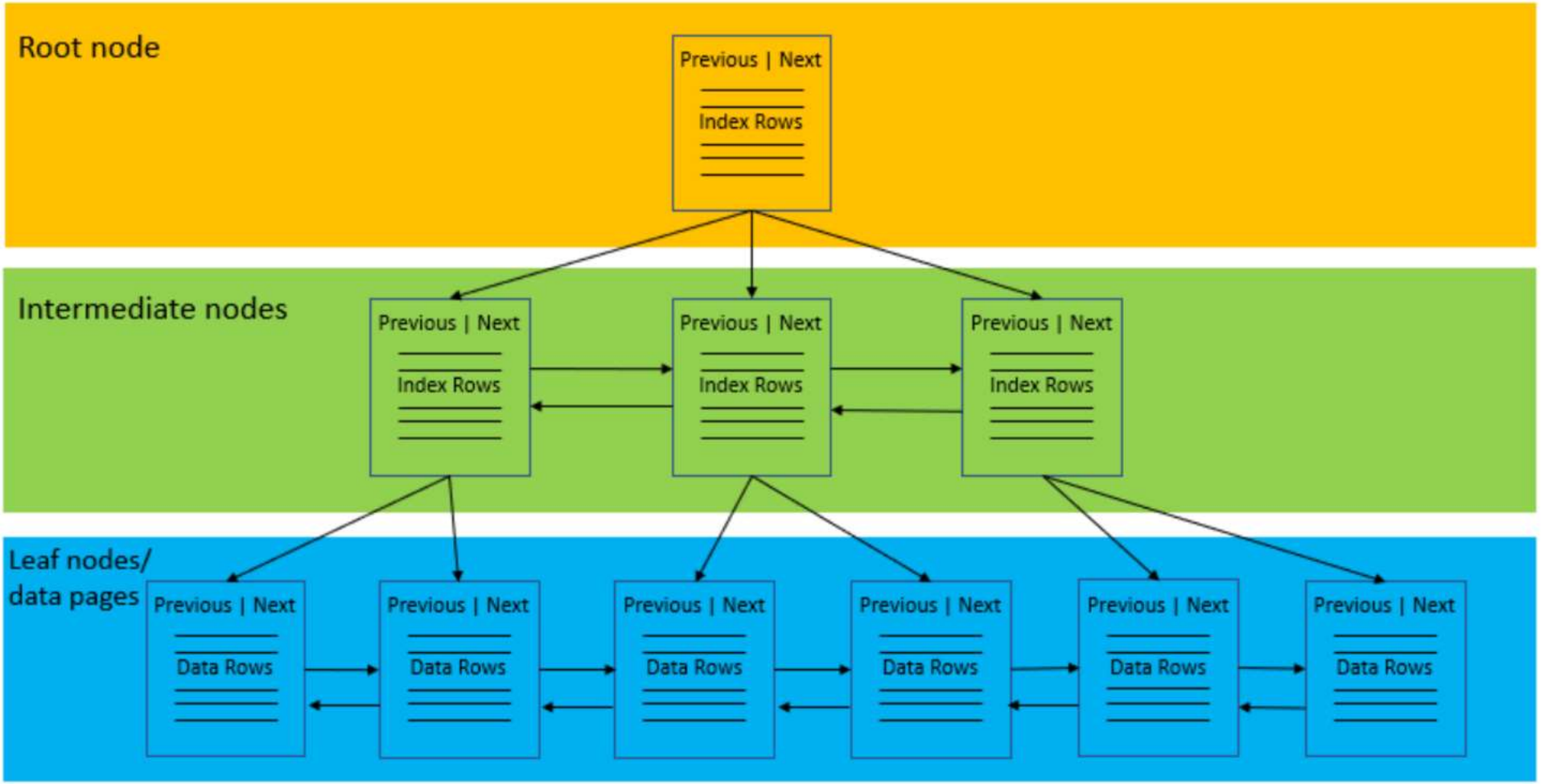
Hai loại index trong SQL Server

- Clustered index
- Non-clustered index

Clustered index

- Lưu trữ các bản ghi trong một cấu trúc được sắp xếp dựa trên giá trị khóa của nó
- Mỗi bảng chỉ có một clustered index vì các bản ghi dữ liệu chỉ có thể được sắp xếp theo một thứ tự
- Bảng có clustered index được gọi là clustered table
- Clustered index tổ chức dữ liệu theo kiểu B-tree

Clustered index

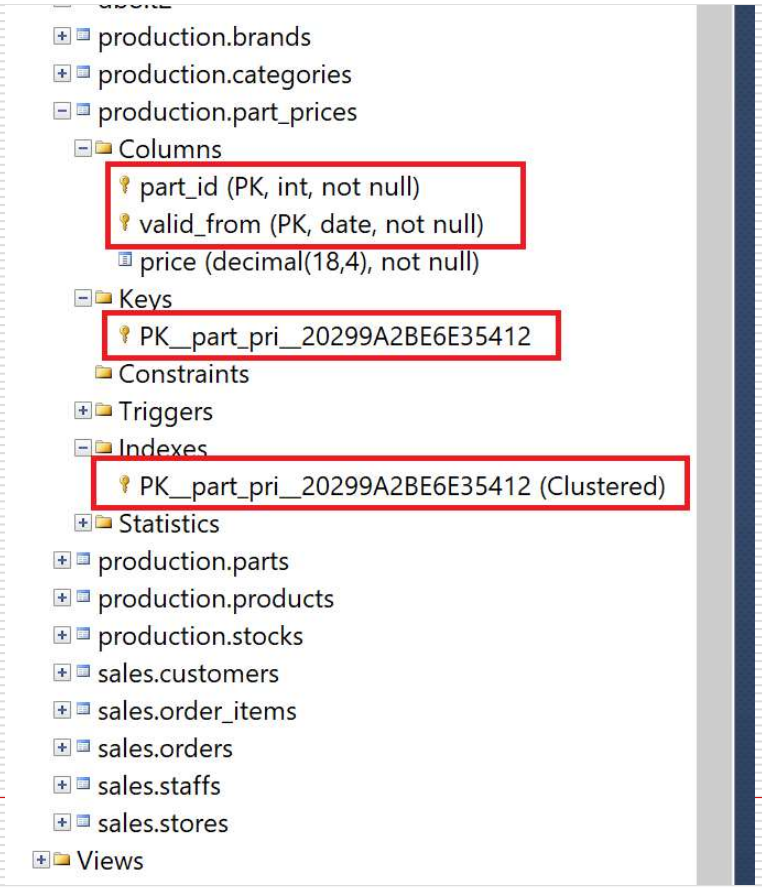


Clustered index

- Nút gốc và nút trung gian chứa các trang chỉ mục để lưu trữ các chỉ mục của các bản ghi
- Nút lá chứa các trang dữ liệu (data pages) của bảng.
- Các trang trong mỗi cấp của index được liên kết theo cấu trúc danh sách liên kết đôi

Clustered index

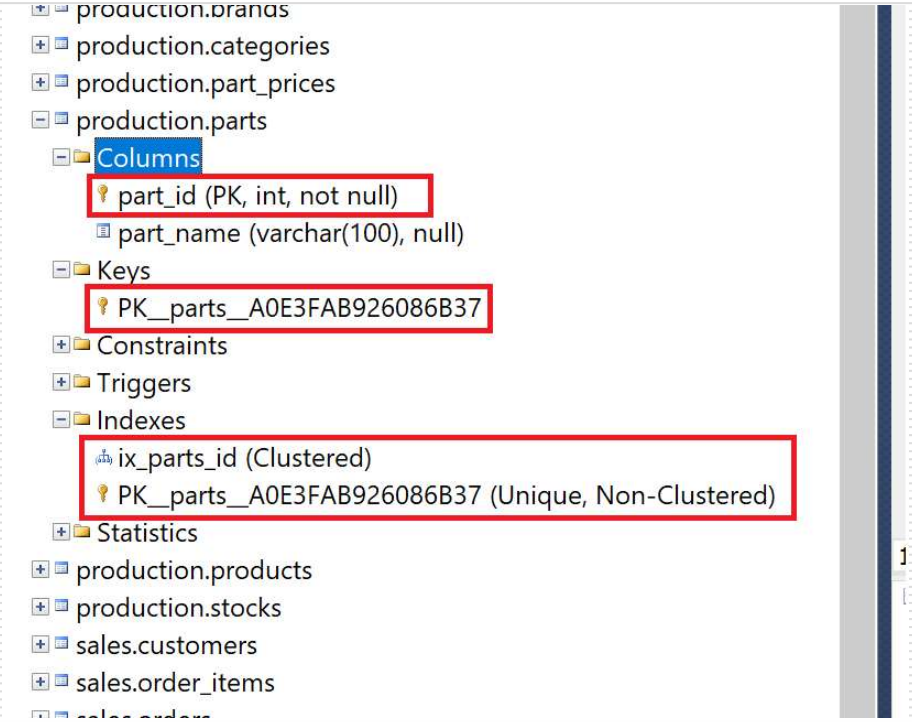
- Khi tạo bảng có khóa chính PK, SQL Server tự động tạo clustered index trên các cột của PK.
- Câu lệnh tạo bảng part_prices có PK gồm 2 cột:




```
CREATE TABLE production.part_prices(
    part_id int,
    valid_from date,
    price decimal(18,4) not null,
    PRIMARY KEY(part_id, valid_from)
);
```

Clustered index

- Nếu thêm khóa chính vào một bảng đã có một clustered index, SQL Server sẽ bắt buộc PK sử dụng non-clustered index




```
ALTER TABLE production.parts
ADD PRIMARY KEY(part_id);
```

Clustered index

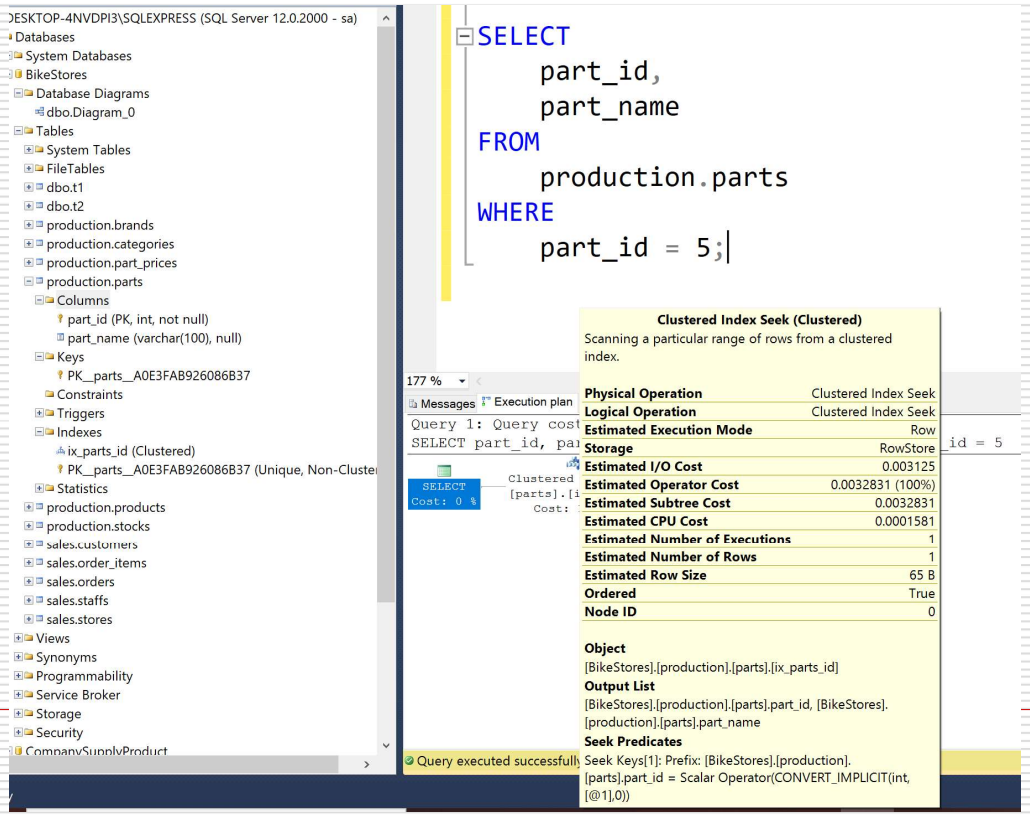
- Tạo clustered index
  - Trong trường hợp bảng ko có PK

```
CREATE CLUSTERED INDEX index_name
ON schema_name.table_name (column_list);
```

```
CREATE CLUSTERED INDEX ix_parts_id
ON production.parts (part_id);
```

- Truy vấn

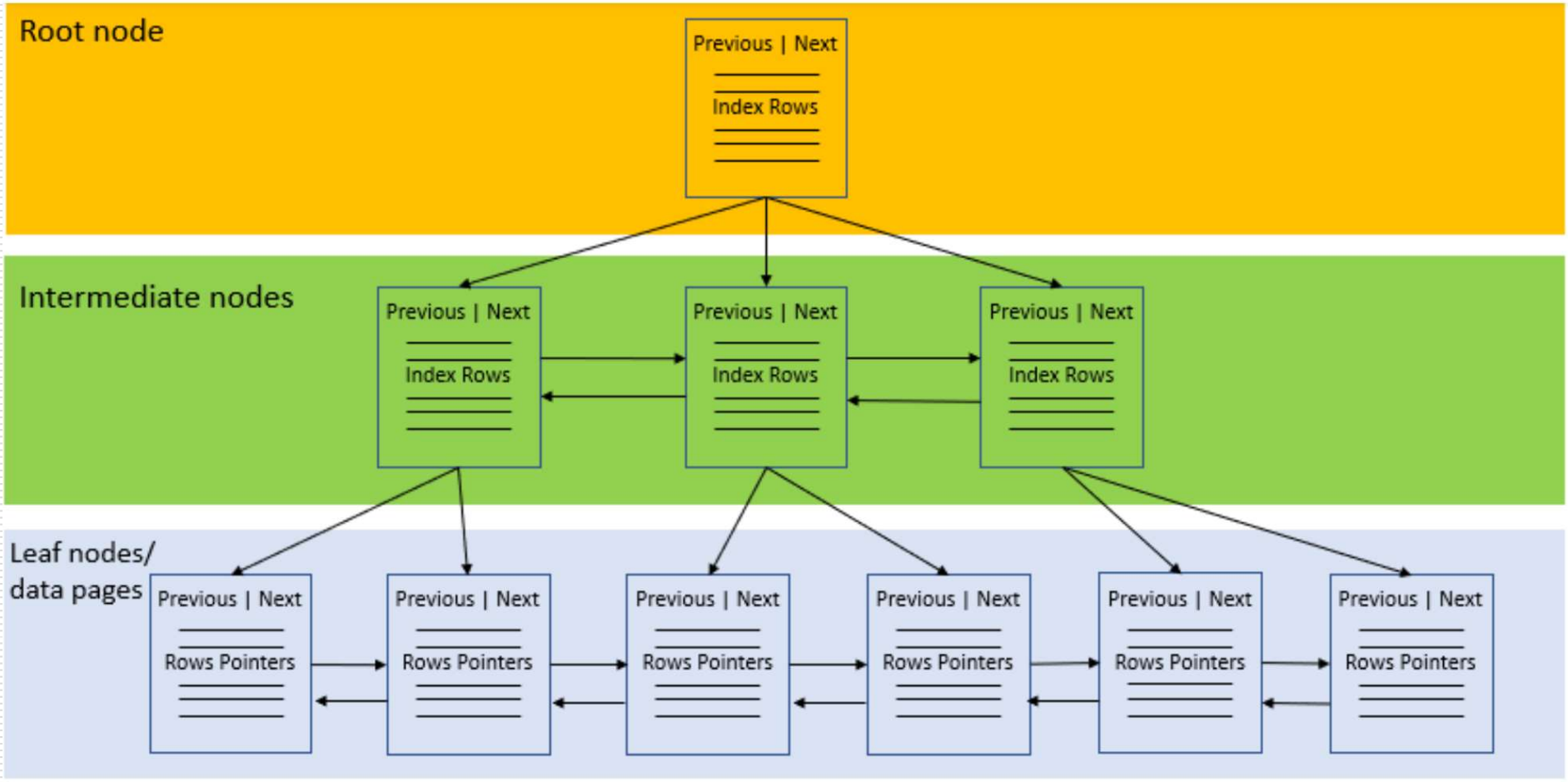
```
SELECT part_id,   part_name
FROM production.parts
WHERE part_id = 5;
```



Non-clustered index

- Là cấu trúc dữ liệu giúp cải thiện tốc độ truy xuất dữ liệu từ các bảng
- Khác clustered index: sắp xếp và lưu trữ dữ liệu riêng biệt với các bản ghi trong bảng.
- Là bản sao dữ liệu của các cột được chọn từ một bảng được liên kết.
- Sử dụng cấu trúc B-tree để tổ chức dữ liệu
- Một bảng có thể có một hoặc nhiều non-clustered index. Mỗi non-clustered index có thể bao gồm một hoặc nhiều cột của bảng.

Non-clustered index



Non-clustered index

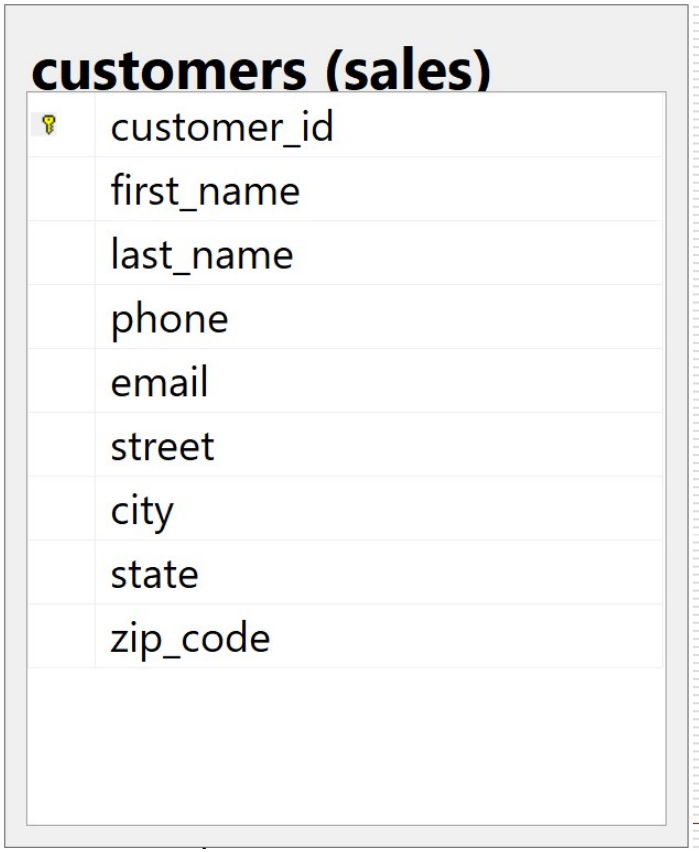
- Bên cạnh việc lưu trữ các giá trị khóa index, các nút lá cũng lưu trữ các con trỏ trỏ tới các bản ghi có chứa các giá trị khóa.
- Những con trỏ bản ghi này còn được gọi là các định vị hàng (row locators)

Non-clustered index

- Tạo non-clustered index

```
CREATE [NONCLUSTERED] INDEX index_name
ON table_name(column_list);
```

- Bảng customers là một clustered table vì nó có PK customer_id



Non-clustered index

- Tìm kiếm khách hàng có địa chỉ ở 'Atwater'

```
SELECT customer_id, city
FROM sales.customers
WHERE city = 'Atwater';
```

- Xem ước lượng kế hoạch thực thi, trình tối ưu hóa truy vấn quét clustered index để tìm các bản ghi, do bảng customers không có index cho cột city
- Gõ lệnh sau, rồi xem lại ước lượng

```
CREATE INDEX ix_customers_city
ON sales.customers(city);
```

Non-clustered index

- Tạo non-clustered index cho nhiều cột
  - Tìm khách hàng có họ là Berg và tên là Monika

```
SELECT customer_id, first_name, last_name
FROM sales.customers
WHERE last_name = 'Berg' AND first_name = 'Monika';
```

  - Xem ước lượng kế hoạch thực thi, rồi gõ lệnh sau, và chạy lại bước trên

```
CREATE INDEX ix_customers_name
ON sales.customers(last_name, first_name);
```

Đổi tên index trong SQL Server

- Câu lệnh sử dụng thủ tục lưu trữ sp_rename

```
EXEC sp_rename  index_name, new_index_name, N'INDEX';
```

Hoặc:

```
EXEC sp_rename @objname = N'index_name', @newname =
N'new_index_name',  @objtype = N'INDEX';
```

- Ví dụ:

```
EXEC sp_rename
    @objname = N'sales.customers.ix_customers_city',
    @newname = N'ix_cust_city' ,
    @objtype = N'INDEX';

EXEC sp_rename
    N'sales.customers.ix_customers_city',
    N'ix_cust_city' ,
    N'INDEX';
```

Đổi tên index trong SQL Server

- Hoặc sử dụng SQL Server Management Studio, kích chuột phải,….

Unique index trong SQL Server

- Unique index có thể một hoặc nhiều cột.
  - Nếu một cột, các giá trị trong cột là duy nhất
  - Nếu nhiều cột, sự kết hợp các giá trị trong các cột này là duy nhất
- Unique index có thể là clustered hoặc non-clustered index
- Cú pháp:

```
CREATE UNIQUE INDEX index_name
ON table_name(column_list);
```

Unique index

- Ví dụ: tạo unique index cho cột email
  - Trước hết, kiểm tra để đảm bảo ko có 2 địa chỉ email trùng lặp

```
SELECT email, COUNT(email)
FROM sales.customers
GROUP BY email
HAVING COUNT(email) > 1;
```

```
CREATE UNIQUE INDEX ix_cust_email
ON sales.customers(email);
```

Unique index

- Thử tạo 1 bảng có 2 cột, rồi tạo unique index trên cả 2 cột đó
- Sau đó, insert dữ liệu vào
- Nếu áp dụng unique index trên cột có nhiều giá trị NULL thì có được ko?
- Unique index vs. unique constraint

Vô hiệu hóa index trong SQL Server

- Trước khi update trên bảng, vô hiệu hóa index giúp tăng tốc quá trình này

```
ALTER INDEX index_name
ON table_name
DISABLE;
```

- Vô hiệu hóa tất cả các index

```
ALTER INDEX ALL ON table_name
DISABLE;
```

Vô hiệu hóa index trong SQL Server

- Nếu vô hiệu hóa một index, trình tối ưu sẽ ko sử dụng index đó để tạo kế hoạch thực hiện truy vấn
- Vô hiệu hóa index trên một bảng, SQL Server vẫn giữ định nghĩa chỉ mục trong siêu dữ liệu và thống kê index trong các non-clustered index
- Vô hiệu hóa index trên view, SQL Server sẽ xóa tất cả dữ liệu index
- Nếu vô hiệu hóa một clustered index của một bảng, ko thể truy cập dữ liệu của bảng sử dụng SELECT, INSERT, UPDATE, DELETE cho đến khi xây dựng lại/xóa clustered index.

Vô hiệu hóa index trong SQL Server

- Ví dụ:

```
ALTER INDEX ix_cust_city
ON sales.customers
DISABLE;

SELECT
    first_name,
    last_name,
    city
FROM
    sales.customers
WHERE
    city = 'San Jose';
```

```
ALTER INDEX ALL
ON sales.customers
DISABLE;

SELECT * FROM sales.customers;
```

Kích hoạt index trong SQL Server

- Sau khi vô hiệu hóa index để UPDATE, cần kích hoạt lại index
  - Cần xây dựng lại index để phản ánh dữ liệu mới trong bảng
- Sử dụng một trong hai lệnh sau
  - ALTER INDEX
  - DBCC DBREINDEX

Kích hoạt index trong SQL Server

- Lệnh ALTER INDEX và CREATE INDEX

```
ALTER INDEX index_name
ON table_name
REBUILD;
```

```
CREATE INDEX index_name
ON table_name(column_list)
WITH(DROP_EXISTING=ON)
```

```
ALTER INDEX ALL ON table_name
REBUILD;
```

Kích hoạt index trong SQL Server

- Lệnh DBCC DBREINDEX

```
DBCC DBREINDEX (table_name, index_name);
```

```
ALTER INDEX ALL ON sales.customers
REBUILD;
```

Xóa index trong SQL Server

- Dùng lệnh DROP INDEX

```
DROP INDEX [IF EXISTS] index_name
ON table_name;
```

- Câu lệnh DROP INDEX ko thể xóa các index được tạo bởi PK hoặc ràng buộc unique
  - Để xóa các index liên quan đến các ràng buộc này, sử dụng lệnh ALTER TABLE DROP CONSTRAINT
- Xóa nhiều index khỏi một/nhiều bảng, dung lệnh:

```
DROP INDEX [IF EXISTS]
   index_name1 ON table_name1,
   index_name2 ON table_name2,
   ...;
```

Filtered index trong SQL Server

- Đôi khi, việc đánh index toàn bộ các bản ghi theo cột nào đó ko hiệu quả, do chỉ truy vấn một phần nào đó bản ghi của bảng.
- Một filtered index là một non-clustered index với một biểu thức cho phép chỉ định những bản ghi nào sẽ được thêm vào index
- Cú pháp:

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

Filtered index trong SQL Server

- Ví dụ: dùng bảng customers, cột phone có nhiều giá trị NULL

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

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

```
SELECT
    first_name,
    last_name,
    phone
FROM
    sales.customers
WHERE phone = '(281) 363-3309';
```

Filtered index trong SQL Server

- INCLUDE

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