Tối ưu hóa truy vấn I

Course: Database Tunning

Viet-Trung Tran

is.hust.edu.vn/~trungtv

Tối ưu truy vấn

- Tối ưu truy vấn: viết lại một truy vấn để chạy nhanh hơn
- Những cách tối ưu khác có thể có tác động không mong muốn tới hệ thống:
  - Thêm index
  - Đổi schema
  - Đổi độ dài của transaction
- Tối ưu truy vấn chỉ có các tác động có lợi, là điều phải làm đầu tiên nếu câu truy vấn chậm

Quá trình tối ưu

1. Phân tích cú pháp
    - Input: câu lệnh SQL
    - Output: biểu thức đại số quan hệ
2. Tối ưu
    - Input: biểu thức đại số quan hệ
    - Output: kế hoạch truy vấn
3. Thiết bị chạy câu truy vấn
    - Input: kế hoạch truy vấn
    - Output: kết quả truy vấn

VD: **phân tích cú pháp:**

- Input: SELECT balance
  FROM account
  WHERE balance < 2500
- Output: biểu thức đại số quan hệ

$$\sigma_{balance<2500}(\Pi_{balance}(account))$$

hoặc:

$$\Pi_{balance}(\sigma_{balance<2500}(account))$$

Tối ưu:

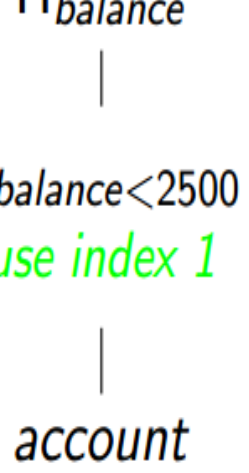
- Input: biểu thức đại số quan hệ

$$\Pi_{balance}(\sigma_{balance<2500}(account))$$

- Output: kế hoạch truy vấn

Kế hoạch truy vấn tối ưu được chọn theo 3 bước:

- Biến đổi tương đương biểu thu DSQH
- Chú thích của biểu thức DSQ
- Chi phí ước tính của các kế hoạch truy vấn khác nhau



- Biến đổi tương đương
  - Tương đương nếu chúng cùng đưa ra cùng một kết quả nhưng thứ tự thực hiện các phép toán khác nhau

  ➔ Tại sao cần biến đổi tương đương?

  Cùng kết quả nhưng thời gian chạy có thể khác nhau $\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$
- Các ng $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

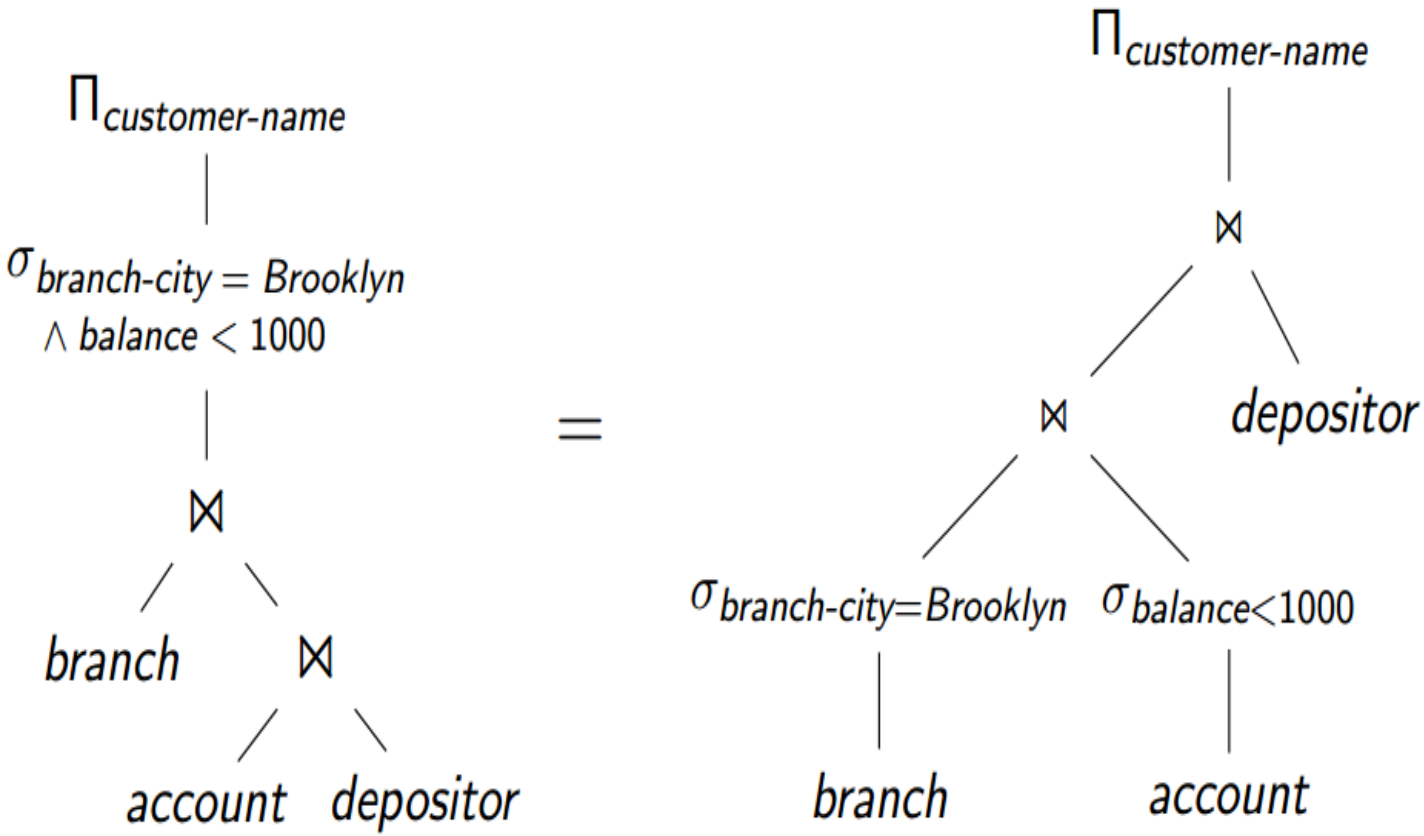
- VD:

  branch(branch-name, branch-city, assets)
  account(account-number, branch-name, balance)
  depositor(customer-name,account-number)

- Query:

```
SELECT customer-name
FROM branch, account, depositor
WHERE branch-city=Brooklyn AND
   balance < 1000 AND
   branch.branch-name = account.branch-name AND
   account.account-number = depositor.account-number
```

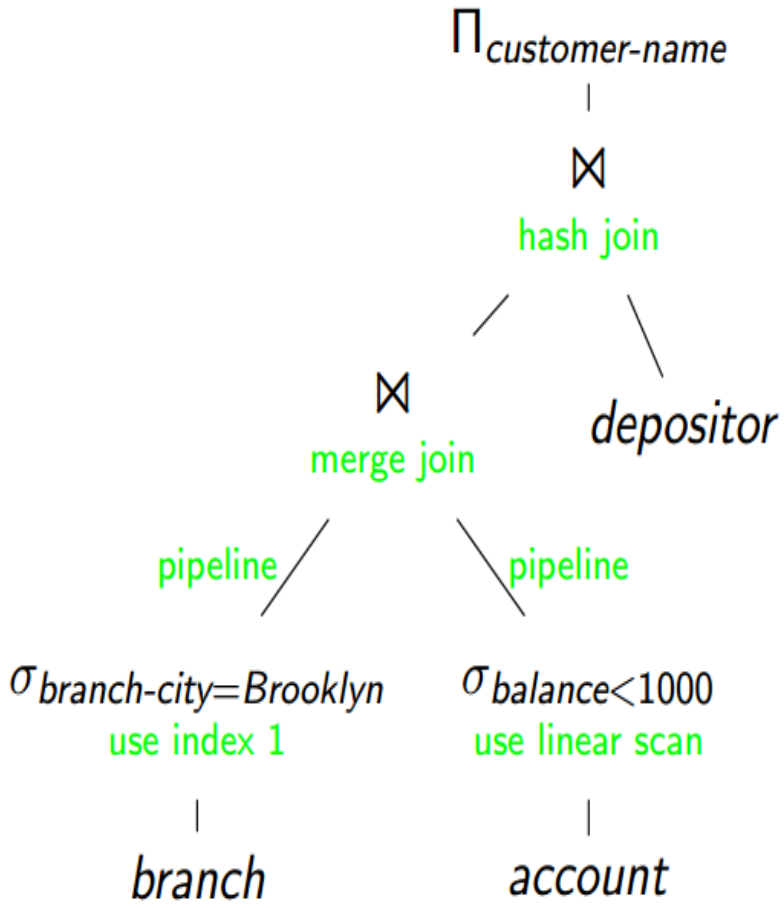
➢ Tối ưu cây truy vấn:



- Tạo kế hoạch truy vấn:
  - Biểu thức DSQH không phải là một kế hoạch truy vấn
  - Các quyết định bổ sung cần thiết:
    - ✓ Các index nào được sử dụng, VD: join hay select
    - ✓ Thuật toán nào được sử dụng: sort-merge hay hash join
    - ✓ …
  - Mỗi biểu thức DSQH có thể dẫn đến nhiều kế hoạch truy vấn
  - Một vài kế hoạch truy vấn có thể tốt hơn

VD:

(account physically sorted by branch-name; index 1 on branch-city sorts records with same value of branch-city by branch-name)



❑ Ước tính chi phí:

- ➢ Kế hoạch truy vấn nào là tốt nhất

→ Là vấn đề khó vì:

- Chi phí chỉ có thể ước tính, ko biết được chính xác
- Có thể có rất nhiều kế hoạch truy vấn

Thống kê cho ước tính chi phí:

VD thống kê:

- Số các bộ dữ liệu trên mỗi quan hệ
- Số block trên bộ nhớ trên mỗi quan hệ
- Số giá trị khác biệt trên mỗi thuộc tính
- Biểu đồ giá trị cho mỗi thuộc tính

→Thống kê được sử dụng để ước tính chi phí của hoạt động

→Vấn đề:

- Chi phí chỉ có thể được ước tính
- Cập nhật số liệu thống kê thường tốn kém nên chúng thường quá hạn

Chọn kế hoạch truy vấn rẻ nhất

- Vấn đề: ước tính chi phí cho tất cả kế hoạch truy vấn là quá đắt
- Giải pháp:
  - Cắt tỉa: dừng sớm để đánh giá 1 kế hoạch
  - heuristics: không đánh giá hết các kế hoạch
- CSDL thực tế áp dụng kết hợp:
  - Áp dụng heuristics để chọn kế hoạch có triển vọng
  - chọn kế hoạch rẻ nhất trong các kế hoạch triển vọng bằng cách cắt tỉa
- VD heuristic:
  - Sử dụng phép chọn càng sớm càng tốt
  - Sử dụng phép chiếu sớm
  - Tránh tích Descartes

Thiết bị chạy câu truy vấn

- Nhận kế hoạch truy vấn từ bộ phận tối ưu hóa
- Chạy kế hoạch truy vấn và trả về kết quả truy vấn cho người sử dụng

Tinh chỉnh truy vấn và tối ưu hóa truy vấn

- Tối ưu hóa là không hoàn hảo:
  - Các phép chuyển đổi chỉ là một phần trong các kế hoạch truy vấn
  - Chỉ một trong các kế hoạch đó được sử dụng
  - Chi phí của các kế hoạch truy vấn chỉ có thể ước tính
- Tinh chỉnh truy vấn: giúp đơn giản hóa việc tối ưu hóa truy vấn

Các câu truy vấn nào nên được viết lại?

- Các câu truy vấn mà chạy quá chậm
- Làm thế nào để tìm được các truy vấn này?
  - Các truy vấn mà truy xuất vào quá nhiều vùng nhớ, VD: câu truy vấn mà quét tất cả các bảng
  - Các câu truy vấn mà có các index liên quan không được sử dụng

Tối ưu hóa truy vấn:

- Tránh DISTINCT
- Truy vấn lồng thường không hiệu quả
- Bảng tạm có thể cải thiện tốc độ truy vấn
- Sử dụng các nhóm index để join
- HAVING và WHERE
- Đặc thù hệ thống: OR và thứ tự trong mệnh đề FROM

VD:

- Employee(ssnum,name,manager,dept,salary,numfriends)
  - clustering index on ssnum
  - non-clustering index on name
  - non-clustering index on dept
  - keys: ssnum, name
- Students(ssnum,name,course,grade)
  - clustering index on ssnum
  - non-clustering index on name
  - keys: ssnum, name
- Techdept(dept,manager,location)
  - clustering index on dept
  - key: dept
  - manager may manage many departments
  - a location may contain many departments

DISTINCT

VD: tìm các nhân viên làm việc tại phòng Hệ thống thông tin

```
SELECT DISTINCT ssnum
FROM Employee
WHERE dept = 'information systems'
```

→DISTINCT là không cần thiết

- Ssnum là key của Employee, nên cũng là key của một phần Employee
- Chú ý: kể từ lúc 1 index được xác định trong ssnum, gần như là không có chi phí trong các ví dụ cụ thể này

Các truy vấn lồng không tương quan

- Nhiều hệ thống xử lý truy vấn lồng không hiệu quả
- "không tư[illegible]uy vấn ngoài không đư[illegible]

```
SELECT ssnum
FROM Employee
        WHERE dept IN (SELECT dept FROM Techdept)
```

- Có thể dẫn đến đánh giá không hiệu quả:
  - Kiểm tra mỗi nhân viên nếu họ ở trong Techdept
  - Index trong Employee.dept không được sử dụng
- Câu truy vấn hiệu quả:
  - ✓ Tìm employee với mỗi dep[illegible]
  - ✓ Sử dụng index trong Emp[illegible]

```
SELECT ssnum
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
```

Bảng tạm

- Nhược điểm:
  - Bắt hệ thống hoạt động theo thứ tự không tối ưu
  - Tạo bảng tạo trong vài hệ thống khác gây nên hoạt động cập nhật danh mục, có thể gây thắt nút cổ chai
  - Hệ thống có thể mất cơ hội sử dụng index
- Ưu điểm:
  - Để viết lại những câu truy vấn lồng tương quan phức tạp
  - Tránh ORDER BY, và quét trong trường hợp đặc biệt

VD: Tìm tất cả nhân viên IT có thu nhập trên 40000

```
SELECT * INTO Temp
FROM Employee
WHERE salary > 40000

SELECT ssnum
FROM Temp
WHERE Temp.dept = 'IT'
```

→ Index trên dept không được sử dụng

→ Mất phí tạo bảng tạm

→ Truy vấn tối ưu

```
SELECT ssnum
FROM Employee
WHERE Employee.dept = 'IT'
      AND salary > 40000
```

Join: sử dụng cụm index và các giá trị

- SQL: tìm tất cả cá học sinh là employee
- SQL không hiệu quả:

```
SELECT Employee.ssnum
FROM Employee, Student
WHERE Employee.name = Student.name
```

- SQL hiệu quả:

```
SELECT Employee.ssnum
FROM Employee, Student
WHERE Employee.ssnum = Student.ssnum
```

→ Join trên 2 khối index và cho phép merge join

→ Bình đẳng số được đánh giá nhanh hơn là bình đẳng chuỗi

Không sử dụng HAVING chỗ mà WHERE đã đủ

- Query: tìm lương trung bình của phòng IT
- Không hi

```
SELECT AVG(salary) as avgsalary, dept
FROM Employee
GROUP BY dept
HAVING dept = 'IT'
```

→ Vấn đề: có thể tính toán trung bình của tất cả employee của các phóng khác

- Hiệu qu

```
SELECT AVG(salary) as avgsalary, dept
FROM Employee
WHERE dept = 'IT'
GROUP BY dept
```

Sử dụng view với sự quan tâm:

- View: truy vấn nhìn đơn giản hơn, nhưng thường chậm hơn

Creating a view:

```
CREATE VIEW Techlocation
AS SELECT ssnum, Techdept.dept, location
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
```

Using the view:

```
SELECT location
FROM Techlocation
WHERE ssnum = 452354786
```

System expands view and executes:

```
SELECT location
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
      AND ssnum = 452354786
```

- Truy vấn: lấy tên phòng có nhân viên mã là 452354786 (người làm ở phỏng công nghệ)
- SQL không hiệu quả:

```
SELECT dept
FROM Techlocation
WHERE ssnum = 452354786
```

- SQL mở rộng:

```
SELECT dept
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
      AND ssnum = 452354786
```

- SQL hiệu quả (không có join):

```
SELECT dept
FROM Employee
WHERE ssnum = 452354786
```

Đặc thù hệ thống: index và OR

- Vài hệ thống không bao giờ sử dụng index khi các điều kiện là OR-connected
- VD: tìm nhân viên với tên Smith hoặc người trong phòng thu m

```
SELECT Employee.ssnum
FROM Employee
WHERE Employee.name = 'Smith'
OR Employee.dept = 'acquisitions'
```

Fix: use UNION instead of OR

```
SELECT Employee.ssnum
FROM Employee
WHERE Employee.name = 'Smith'
UNION
SELECT Employee.ssnum
FROM Employee
WHERE Employee.dept = 'acquisitions'
```

Đặc thù hệ thống: Order trong mệnh đề FROM

- Order trong mệnh đề FROM không thích hợp
- Tuy nhiên: với các join dài (hơn 8 bảng) và trong một vài vấn đề trật tự hệ thống
- Làm thế nào đề tìm ra: kiểm tra kế hoạch truy vấn