Chương 3

Xử lý câu truy vấn phân tán

ptndiem@cit.ctu.edu.vn

Nội dung

- **Giới thiệu tổng quan về xử lý câu truy vấn**
- Xử lý câu truy vấn tập trung
- Xử lý truy vấn phân tán

CSDL mẫu

CUSTOMER (**CID**, CNAME, STREET, CCITY);

BRANCH (**BNAME**, ASSETS, BCITY);

ACCOUNT (**A#**, *CID*, *BNAME*, BAL);

LOAN (**L#**, *CID, BNAME*, AMT);

TRANSACTION (**TID**, *CID, A#*, Date, AMOUNT);

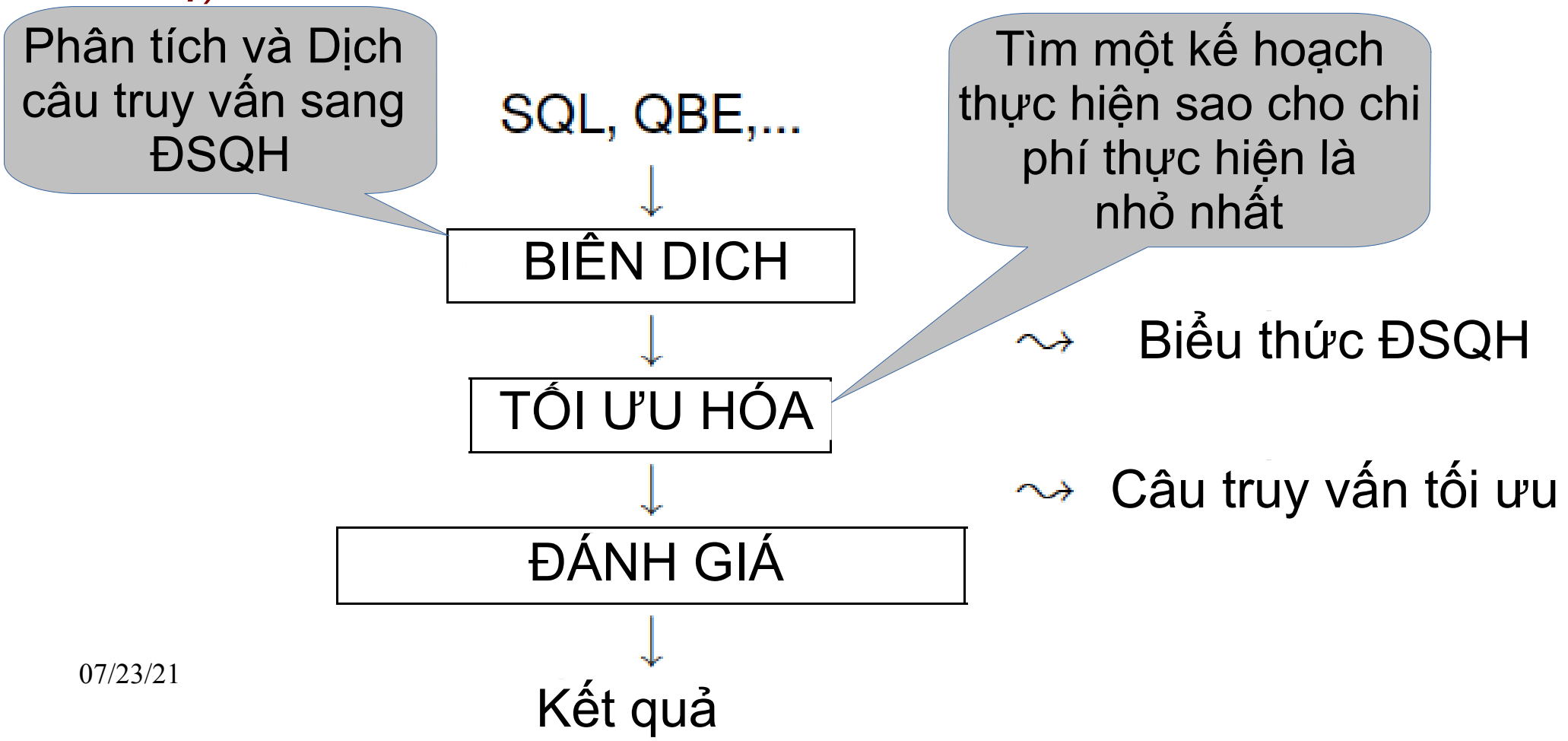
Mục tiêu xử lý câu truy vấn

- Cho một truy vấn. *Mục tiêu là đánh giá tính hiệu quả*.
  - Làm thế nào để chuyển từ SQL sang cây biểu thức đại số quan hệ?
  - Làm thế nào bộ phận tối ưu hóa có được nhiều kế hoạch thực thi (execution plan) có thể.
  - Làm thế nào lựa chọn trong số các kế hoạch này.
  - Làm thế nào để câu truy vấn sau đó được đánh giá.

=> Đây là những kỹ thuật cơ bản, được cài đặt trong bất kỳ DBMS quan hệ nào

Xử lý câu truy vấn

*Xử lý câu truy vấn (*Query Processing*): Là quá trình 3 bước chuyển đổi câu truy vấn cấp cao (ví dụ SQL) sang một câu truy vấn cấp thấp hơn* ***tương đương và hiệu quả hơn*** *(đại số quan hệ).*



Xử lý câu truy vấn

- Trong một hệ thống tập trung, mục tiêu của **bộ xử lý câu truy vấn** (query processor) có thể bao gồm:
  - Tối thiểu hóa thời gian trả lời truy vấn.
  - Tối đa hoá tính song song trong hệ thống.
  - Tối đa hóa thông lượng hệ thống.
  - Tối thiểu hóa tổng số tài nguyên được sử dụng (số lượng bộ nhớ, không gian đĩa, bộ nhớ cache, vv).
  - Các mục tiêu khác

=> Hệ thống có thể không thỏa mãn tất cả những mục tiêu này

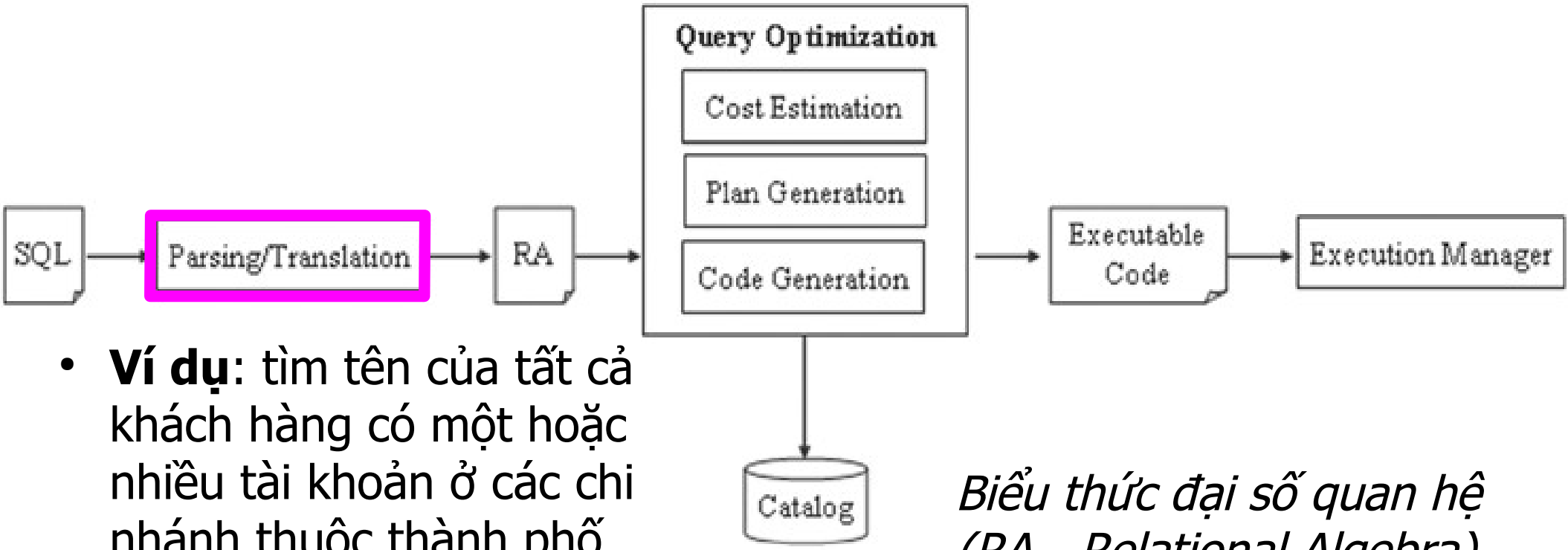
Nội dung

- Giới thiệu tổng quan về xử lý câu truy vấn
- **Xử lý câu truy vấn tập trung**
- Xử lý truy vấn phân tán

3 bước xử lý câu truy vấn

1. Phân tích và dịch câu truy vấn.
   - Kiểm tra cú pháp và các quan hệ
   - Nếu câu truy vấn chính xác, nó được viết lại dưới dạng biểu thức đại số quan hệ tương đương.
2. Tối ưu hóa:
   - Viết lại câu truy vấn thành các biểu thức tương đương
   - Lựa chọn thuật toán riêng biệt cho mỗi phép toán
   - Thu được các *kế hoạch thực thi* (execution plan) và chi phí ước lượng
   - Chọn *kế hoạch* tốt nhất.
3. Thực thi/Đánh giá: *kế hoạch thực thi* được biên dịch, được thực hiện và trả về kết quả.

Phân tích và dịch câu truy vấn



- **Ví dụ**: tìm tên của tất cả khách hàng có một hoặc nhiều tài khoản ở các chi nhánh thuộc thành phố Edina ?

SQL

```
Select c.Cname
From Customer c, Branch b, Account a
Where c.CID = a.CID
   AND a.Bname = b.Bname
   AND b.Bcity = 'Edina';
```

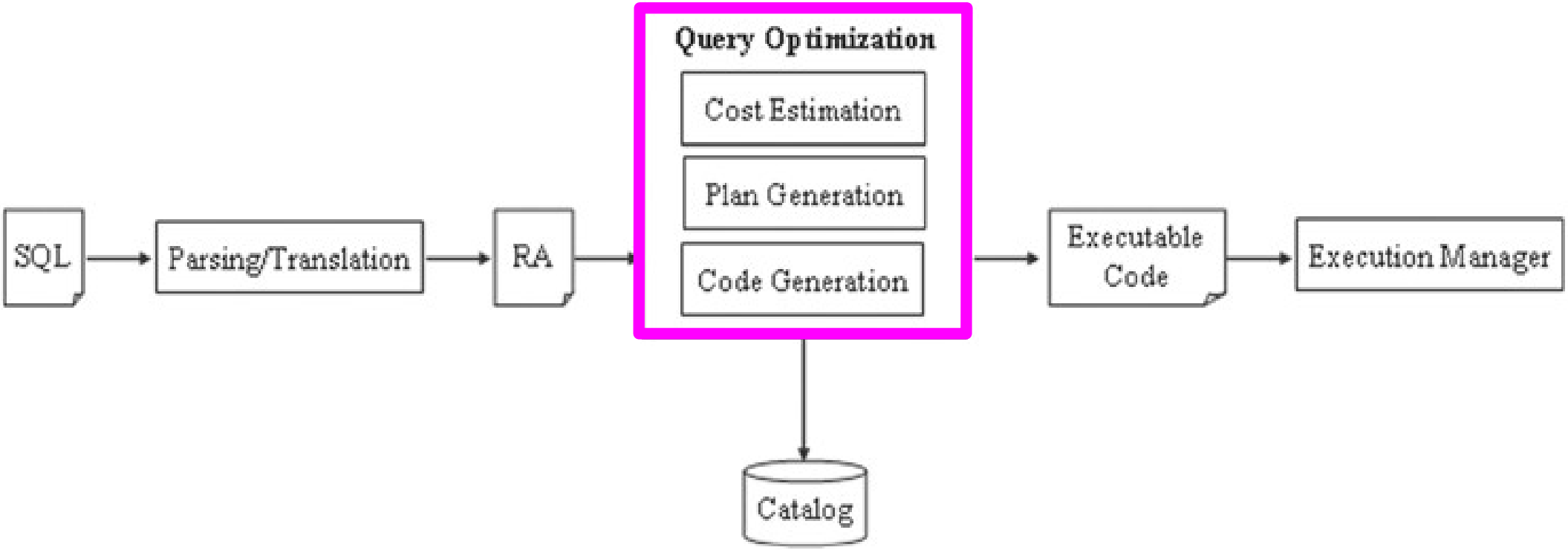
ĐSQH

Biểu thức đại số quan hệ (RA - Relational Algebra) mà bộ phân tích cú pháp có thể tạo ra :

$\pi_{cname}(\sigma_{Bcity='Edina'}$ (Customer * (Account * Branch)))

Phân tích và dịch câu truy vấn

- DBMS **không thực thi ngay** biểu thức ĐSQH trên.
- Biểu thức này phải đi qua một một tập các phép biến đổi và tối ưu hóa trước khi nó đã sẵn sàng để chạy
- **Bộ phận tối ưu hoá truy vấn** (query optimizer) là thành phần chịu trách nhiệm thực hiện điều này.



Nội dung

- Giới thiệu tổng quan về xử lý câu truy vấn
- **Tối ưu hoá câu truy vấn**
- Xử lý truy vấn phân tán

Tối ưu hoá câu truy vấn

- Tối ưu hoá gồm ba bước:
  - Ước lượng chi phí,
  - Sinh ra kế hoạch (plan) thực thi và
  - Sinh mã câu truy vấn.
- Trong một số DBMS (ví dụ: DB2), có một bước bổ sung được gọi là "**viết lại câu truy vấn** (query Rewrite)"
  - Được thực hiện trước khi tối ưu hóa được thực hiện.
  - Bộ phận tối ưu hóa viết lại câu truy vấn bằng cách:
    - loại bỏ điều kiện dư thừa,
    - loại bỏ các biểu thức con thừa và
    - đơn giản hóa các biểu thức phức tạp như lồng nhau

Tối ưu hoá câu truy vấn

- Những sửa đổi trong **câu truy vấn viết lại** được thực hiện bất kể thống kê cơ sở dữ liệu.
- Thống kê được sử dụng trong bước tối ưu để tạo ra một kế hoạch tối ưu.
- **Nhắc lại**: một kế hoạch tối ưu có thể không nhất thiết là kế hoạch tốt nhất cho truy vấn.

Ví dụ: $\pi_{cname}(\sigma_{Bcity='Edina'}(Customer \times (Account \times Branch)))$

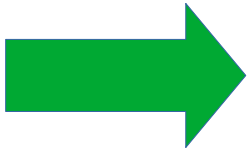
không hiệu quả vì tích Cartesian sinh ra quan hệ trung gian lớn

=> dùng kết nối tự nhiên

$\pi_{cname}((Customer * Account) *(Account * (\sigma_{Bcity='Edina'}(Branch))))$

=> loại bỏ dư thừa

$\pi_{cname}(Customer * (Account * \sigma_{Bcity='Edina'}(Branch)))$



Tối ưu hoá câu truy vấn

Customer : 1000 dòng
Branch : 50 dòng, Có 1 chi nhánh ở Edina
Trung bình, mỗi khách hành có 2 tài khoản
Trung bình mỗi chi nhánh có 40 tài khoản.
=> Account có ? Dòng => 2000

$\pi_{cname}(\sigma_{Bcity='Edina'}(Customer \times (Account \times Branch)))$

R1 ← Account x Branch => 100.000 dòng
R2 ← Customer x R1 => 100.000.000 dòng
Mỗi dòng 100B => R2 ~ 10GB

$\pi_{cname}(\sigma_{Bcity='Edina'}((Customer * Account) * Branch))$

R1 ← Customer *Account => 2000
R2 ← R1 * Branch => 2000
Mỗi dòng 100B
=> R2 ~ 200KB

Tích Cartesian sinh ra quan hệ trung gian lớn
=> dùng kết nối tự nhiên

Tối ưu hoá câu truy vấn

- ***Tối ưu hoá cái gì*** ?
  - Tối ưu hoá việc sử dụng tài nguyên: bộ xử lý, truy xuất đĩa, giao tiếp (D-DB)
  - Tối ưu hoá:
    - thời gian trả lời câu truy vấn;
    - số lượng câu truy vấn được xử lý trên một đơn vị thời gian (tốc độ).
  - Tối ưu hóa truy vấn nhằm mục đích giảm tối thiểu hàm chi phí:

Chi phí I / O + chi phí CPU→ Min

=> Sử dụng các thông tin nào cho tối ưu ?

Tối ưu hoá câu truy vấn

- **Các thông tin sử dụng để tối ưu**
  - Lược đồ luận lý của cơ sở dữ liệu, mô tả các bảng, các ràng buộc toàn vẹn
  - Lược đồ vật lý của cơ sở dữ liệu, chỉ mục và các đường dẫn, kích thước của khối (block)
  - **Thống kê**: kích thước của các bảng, của chỉ mục, sự phân phối các giá trị, tỷ lệ cập nhật ...
  - Đặc điểm của hệ thống: tính song song, bộ vi xử lý chuyên dụng
  - **Giải thuật**: chúng có thể khác nhau tùy thuộc vào hệ thống, ví dụ giải thuật kết nối, chọn, sắp xếp
    - Tất cả các phép toán ĐSQH đều có thể tốn chi phí

    => Cần chọn giải thuật phù hợp

Tối ưu hoá câu truy vấn

- Cho biểu thức:

  $\pi_{cname}(Customer * (Account * \sigma_{Bcity='Edina'}(Branch)))$

- Có thể có nhiều biểu thức khác tương đương với biểu thức đã cho

  => Tất cả các biểu thức khác nhau cho câu truy vấn được đánh giá bởi bộ phận tối ưu hóa truy vấn để tìm biểu thức truy vấn tối ưu ?

Xây dựng biểu thức ĐSQH tương đương

- Cho một câu truy vấn với nhiều toán tử đại số quan hệ:
  - Có nhiều lựa chọn có thể được sử dụng để thể hiện câu truy vấn.
  - Những lựa chọn được tạo ra bằng cách áp dụng **các tính chất ĐSQH**:
    - kết hợp,
    - giao hoán,
    - Idempotent (luỹ đẳng),
    - phân phối,...

Các tính chất của ĐSQH

- Toán tử một ngôi (Uop - chọn) có tính giao hoán :

  Uop1(Uop2(R)) ≡ Uop2(Uop1(R))

  $\sigma_{Bname='Main'}(\sigma_{Assets>12000000}(Branch)) \equiv \sigma_{Assets>12000000}(\sigma_{Bname='Main'}(Branch))$

- Toán tử một ngôi có tính chất luỹ đẳng

  Uop((R)) ≡ Uop1(Uop2((R))

  $\sigma_{Bname='Main' \wedge Assets>12000000}(Branch) \equiv \sigma_{Bname='Main'}(\sigma_{Assets>12000000}(Branch))$

- Toán tử 2 ngôi (Bop) có tính kết hợp

  R Bop1 (S Bop2 T) ≡ (R Bop1 S) Bop2 T

- Toán tử 2 ngôi có tính giao hoán **trừ phép trừ**

  R Bop1 S ≡ S Bop1 R

Các tính chất của ĐSQH

- Toán tử một ngôi phân phối đối với một số toán tử 2 ngôi

Uop(R Bop S) ≡ (Uop(R)) Bop (Uop(S))

$\sigma_{sl>5000}$ ($\pi_{cname,,sal}$(Customer) UNION $\pi_{Ename,\ sal}$ (CUSloyee)) ≡

$\sigma_{sl>5000}$ ($\pi_{cname,sal}$(Customer)) UNION $\sigma_{sl>5000}$ ($\pi_{Ename,\ sal}$ (CUSloyee))

- Toán tử một ngôi có thể được tính toán (tính chất ngược lại phân phối) đối với một số toán tử hai ngôi:

(Uop(R)) Bop (Uop(S)) ≡ Uop(R Bop S)

$\sigma_{sl>5000}$ ($\pi_{cname,sal}$(Customer)) UNION $\sigma_{sl>5000}$ ($\pi_{Ename,\ sal}$ (CUSloyee))
≡ $\sigma_{sl>5000}$ ($\pi_{cname,,sal}$(Customer) UNION $\pi_{Ename,\ sal}$ (CUSloyee))

Tối ưu hoá câu truy vấn

- Tối ưu hoá gồm ba bước:
  - **Ước lượng chi phí,**
  - Sinh ra kế hoạch (plan) thực thi và
  - Sinh mã câu truy vấn.

Ước lượng chi phí

- Áp dụng các tính chất trên vào một biểu thức ĐSQH
  => có thể tạo nhiều biểu thức tương đương cho câu truy vấn ĐSQH
- **Ví dụ**: $(\sigma_{Bname='Main'}(Account)) * Branch$

  $Branch * (\sigma_{Bname='Main'}(Account))$

  ....

  => Bộ phận tối ưu hoá truy vấn chịu trách nhiệm lựa chọn biểu thức tối ưu nhất.

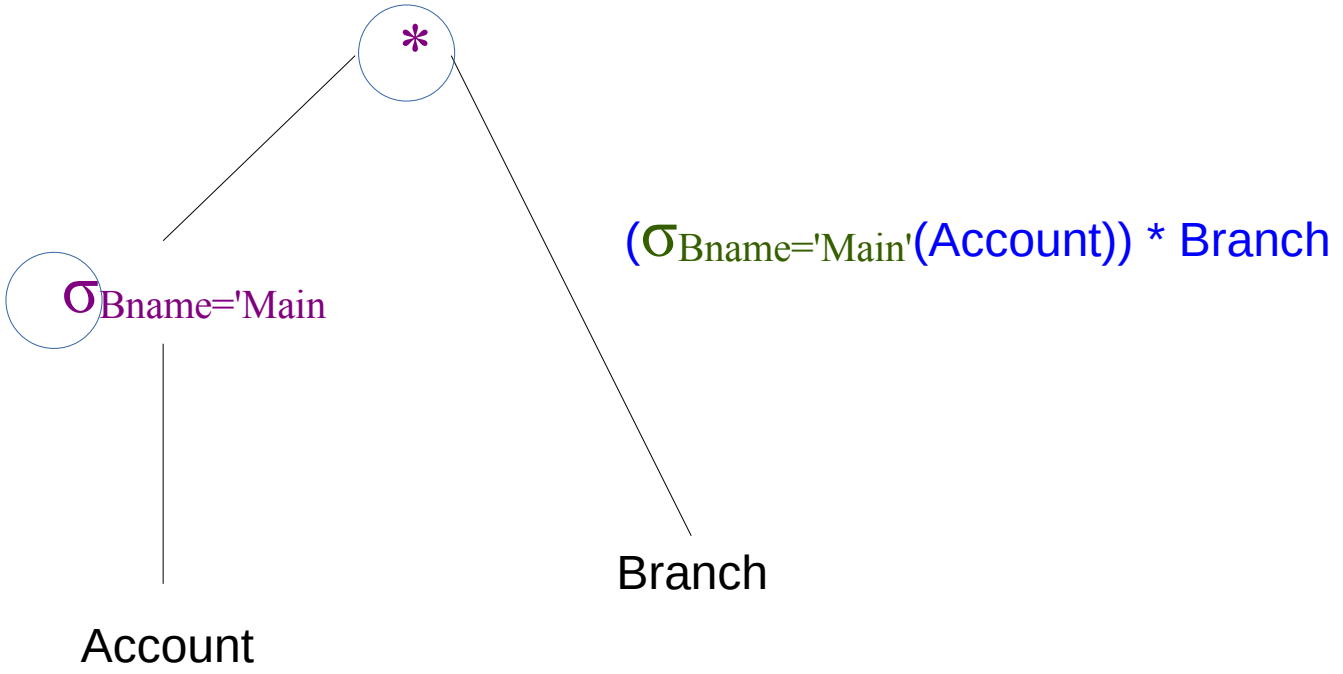
  => để thuận tiện trong việc ước lượng chi phí, khái niệm **cây biểu thức ĐSQH** (query tree) được sử dụng

Cây biểu thức ĐSQH

- Các nút lá là các quan hệ
- Các nút trong là các phép toán ĐSQH
- Toán tử một ngôi nhận vào 1 QH vả trả về 1 QH
- Toán tử 2 ngôi nhận vào 2 QH và trả về 1 QH
- Các kết quả từ các toán tử của một mức được sử dụng bởi các toán tử của mức tiếp theo trong cây cho đến khi **kết quả cuối cùng được tập hợp ở nút gốc**.
- **Ví dụ**: Vẽ cây ĐSQH biểu diễn cho biểu thức:

$$(\sigma_{Bname='Main'}(Account)) * Branch$$

Cây biểu thức ĐSQH

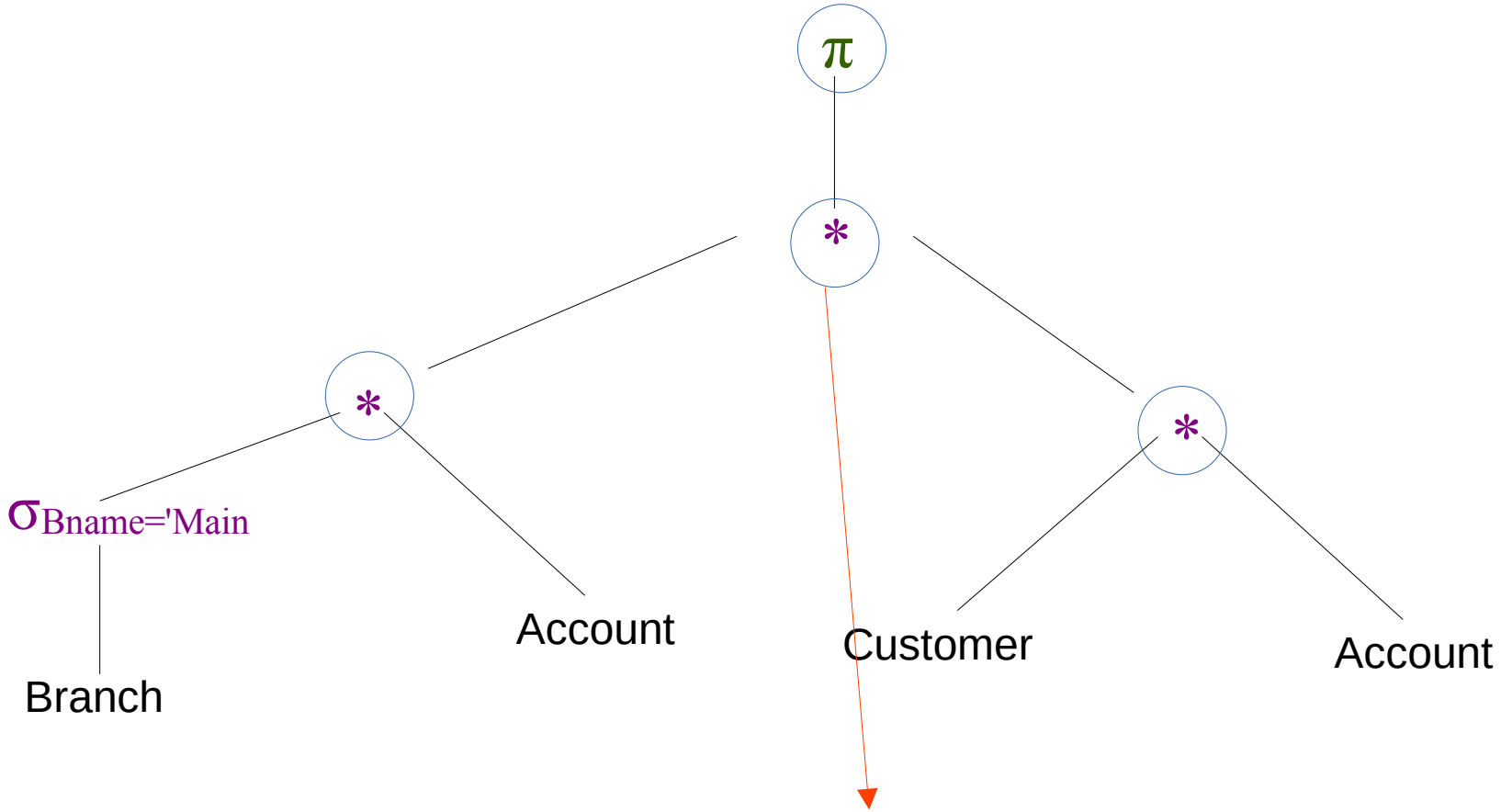


Bài tập: Vẽ cây BT ĐSQH tương ứng với 2 biểu thức sau:

1. $\pi_{cname}((Customer * Account) *(Account * (\sigma_{Bcity='Edina'}(Branch))))$

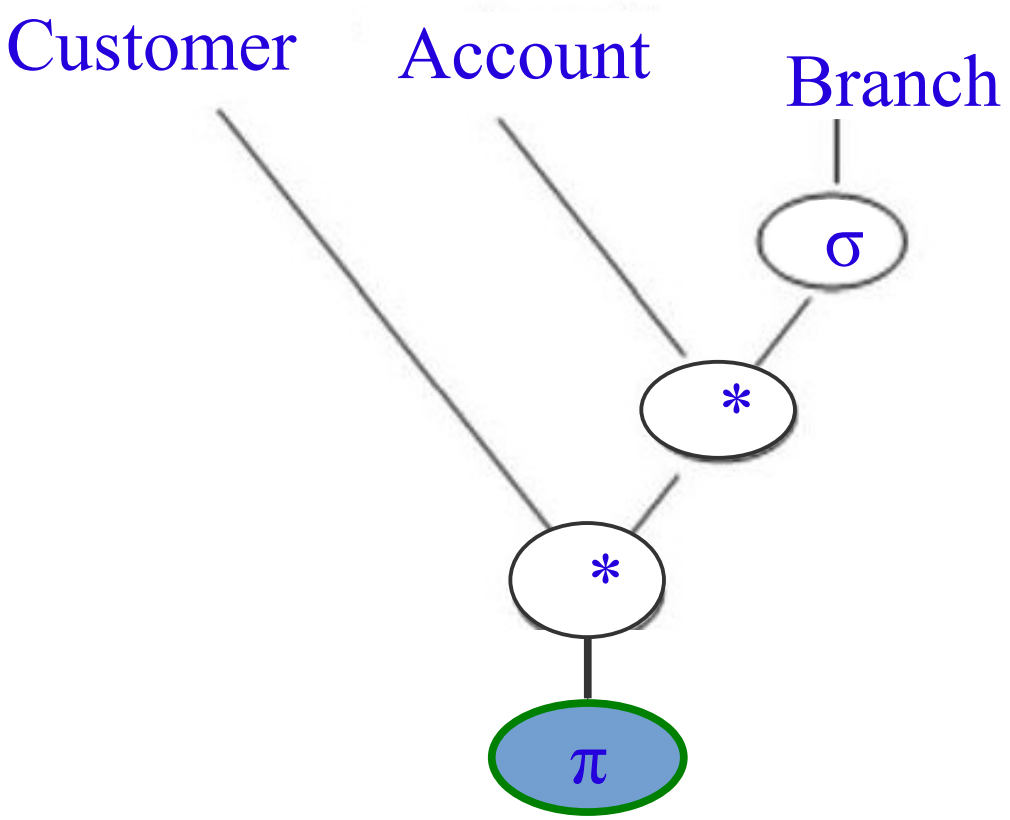
2. $\pi_{cname}(Customer * (Account * \sigma_{Bcity='Edina'}(Branch)))$

Cây biểu thức ĐSQH



1. $\pi_{cname}((Customer * Account) *(Account * (\sigma_{Bcity='Edina'}(Branch))))$

Cây biểu thức ĐSQH



2. $\pi_{cname}(Customer * (Account * \sigma_{Bcity='Edina'}(Branch)))$

Cây biểu thức ĐSQH

3.Tìm số tiền đã vay của khách hàng ở chi nhánh Cần thơ

$$\pi_{AMT}(LOAN * (\sigma_{Bcity='Cantho'}(Branch)))$$

4.Tìm số tiền đã giao dịch của khách hàng Nguyễn Văn Linh ở chi nhánh Cần thơ.

$$A \leftarrow Account * (\sigma_{Bcity='Cantho'}(Branch))$$

$$B \leftarrow \sigma_{Cname='NVL'}(Customer)$$

$$\pi_{AMOUNT}((A * B) * Transaction)$$

Ước lượng chi phí

- Bộ phận tối ưu hoá sẽ phân tích tất cả các cây trong không gian giải pháp và **chọn một cây tối ưu cho câu truy vấn** (cây có chi phí nhỏ nhất).
- Ước lượng chi phí gồm 2 bước : cắt tỉa và phân tích chi phí
  - *Bước đầu tiên trong ước lượng chi phí là **cắt tỉa*** (pruning):
    - *Các cây "xấu" bị loại bỏ* bằng cách áp dụng một vài luật
    - Một trong các luật được xem xét là "trước khi kết nối các quan hệ, chúng phải được giảm kích thước bằng cách áp dụng các phép chọn và chiếu"
  - *Bước tiếp theo là **phân tích chi phí***, các dữ liệu thống kê được dùng

Ước lượng chi phí

- **Ví dụ:**

  Sử dụng lại ví dụ phần trước với câu SQL sau:

```
Select c.Cname
From Customer c, Branch b, Account a
Where c.CID = a.CID
    AND a.Bname = b.Bname
    AND b.Bcity = 'Edina';
```

Ước lượng chi phí

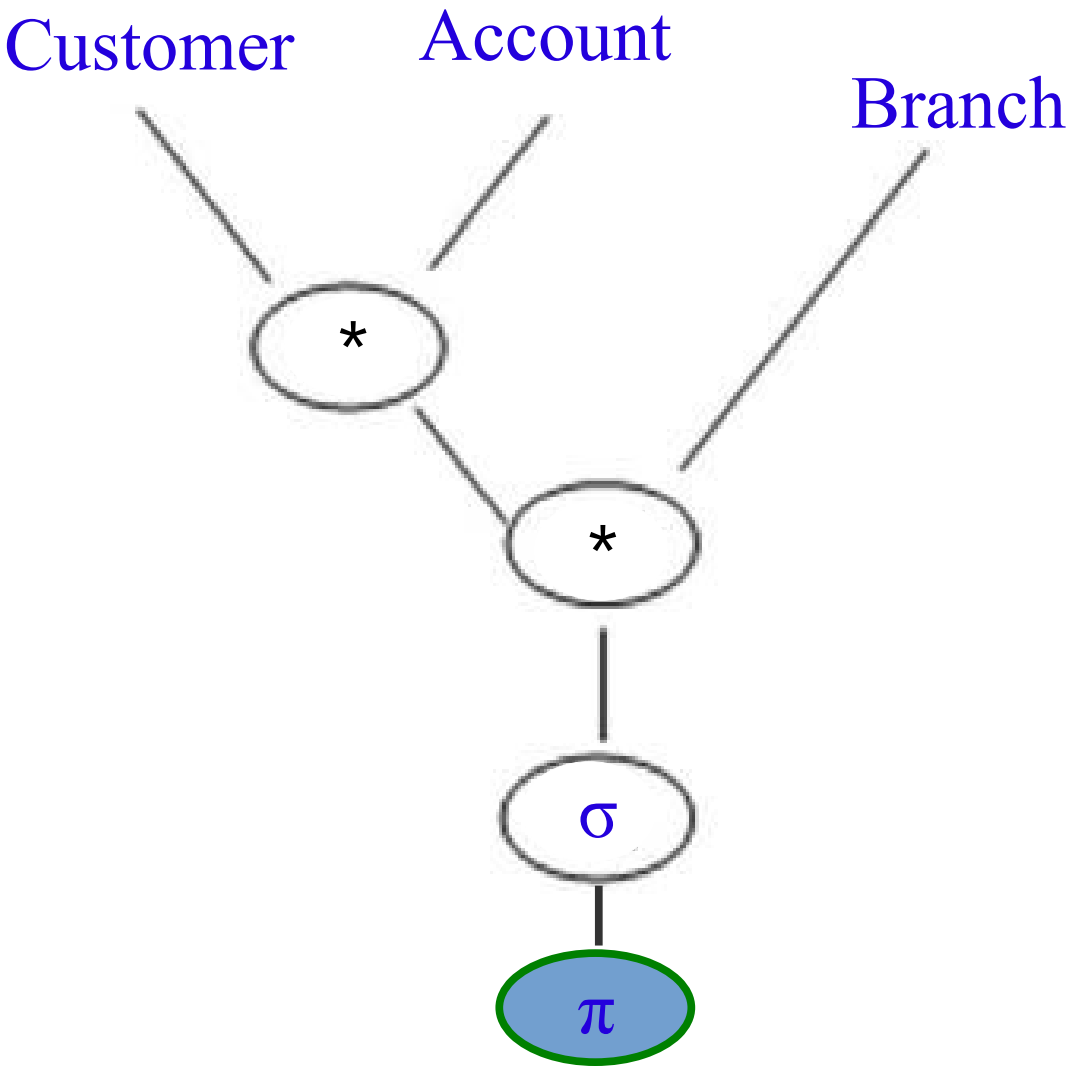
- **B1: Cắt tỉa**
  - Xét các biểu thức tương đương câu SQL đã cho

1. $\pi_{cname}(\sigma_{Bcity='Edina'}((Customer * Account) * Branch))$
2. $\pi_{cname}((Customer * Account) * (\sigma_{Bcity='Edina'}(Branch)))$
3. $\pi_{cname}(Customer * (\sigma_{Bcity='Edina}(Account * Branch)))$
4. $\pi_{cname}(\sigma_{Bcity='Edina'}(Customer * (Account * Branch)))$
5. $\pi_{cname}(Customer * (Account * (\sigma_{Bcity='Edina'}(Branch))))$

  - Vẽ các cây biểu thức ĐSQH tương đương các biểu thức đã cho

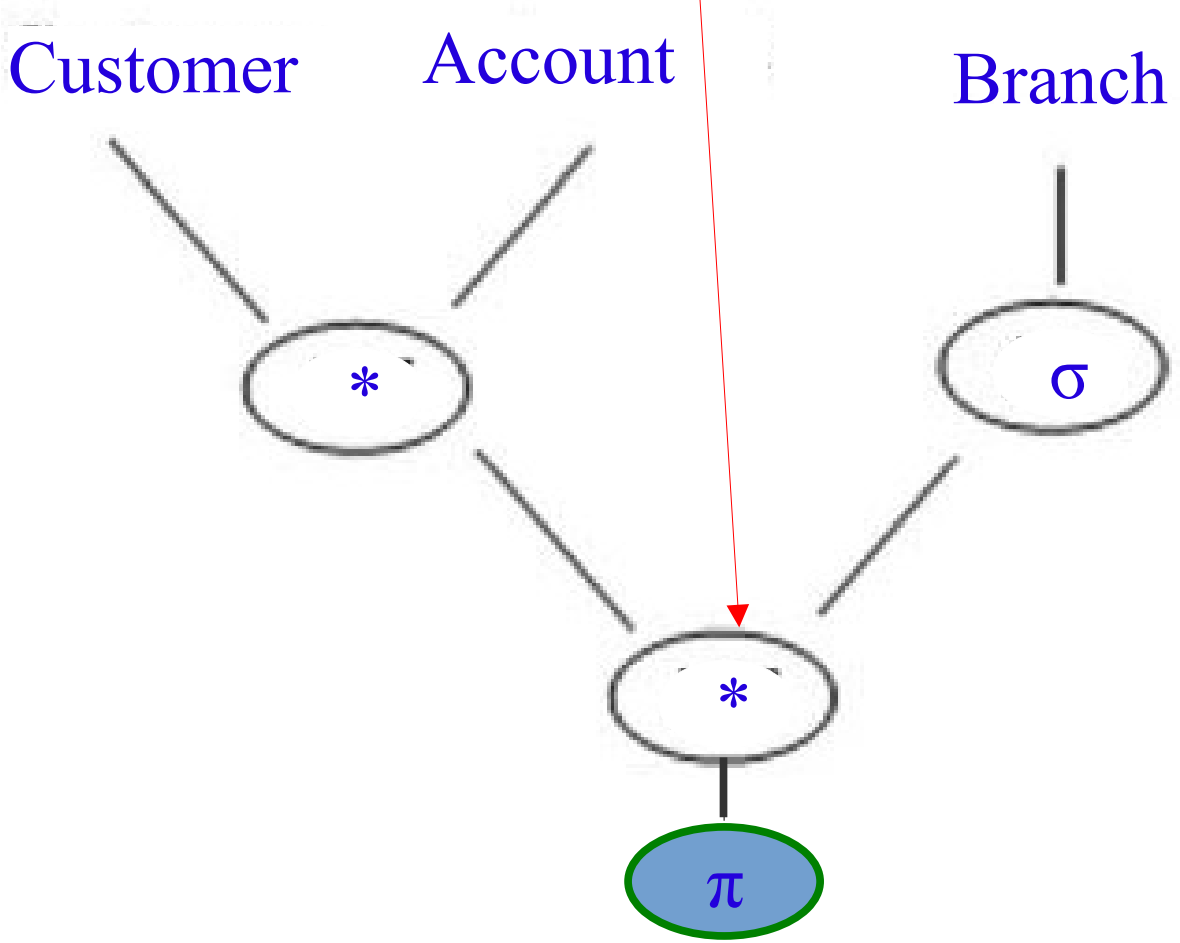
Cắt tỉa

1. $\pi_{cname}(\sigma_{Bcity='Edina'}((Customer * Account) * Branch))$



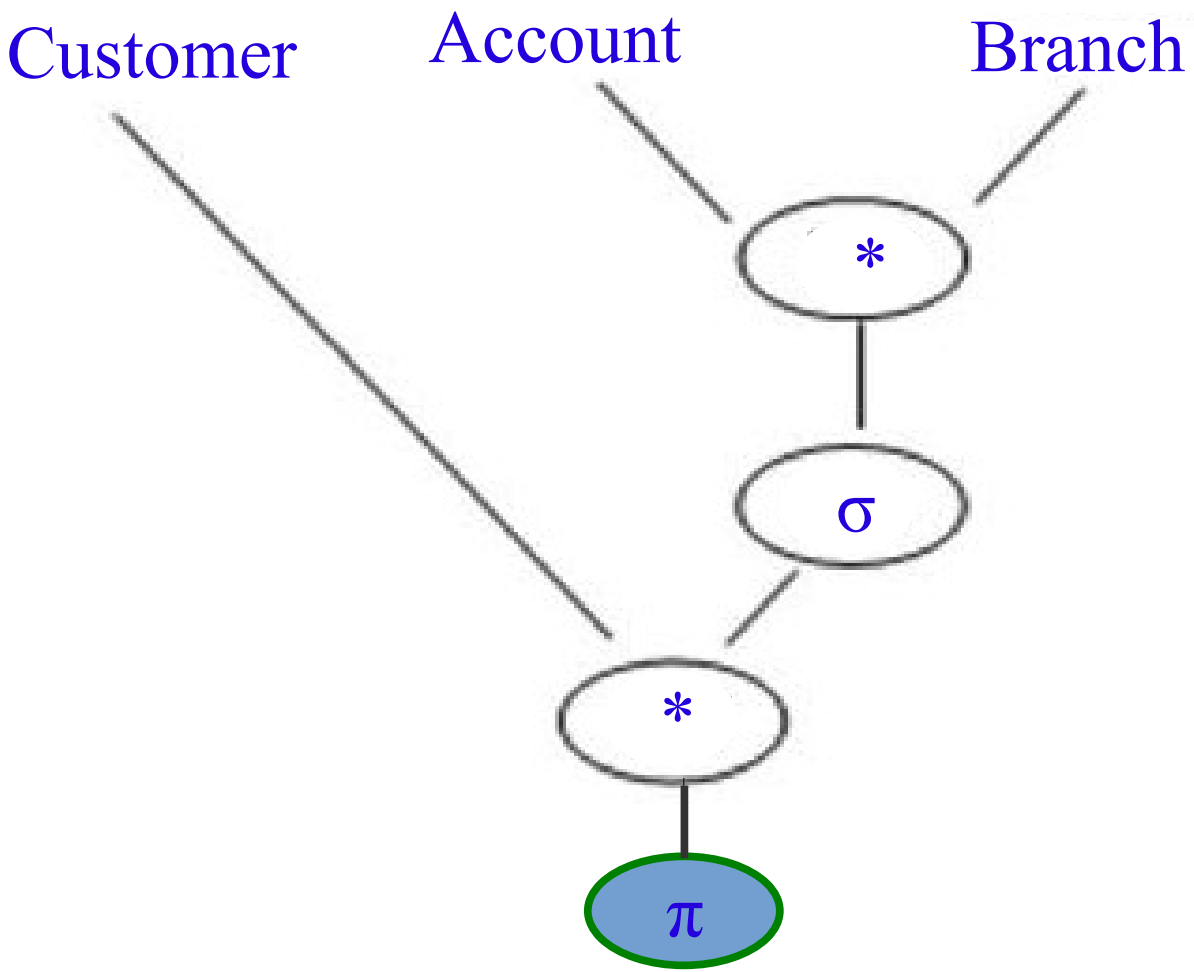
Cắt tỉa

2. $\pi_{cname}((Customer * Account) * (\sigma_{Bcity='Edina'}(Branch)))$



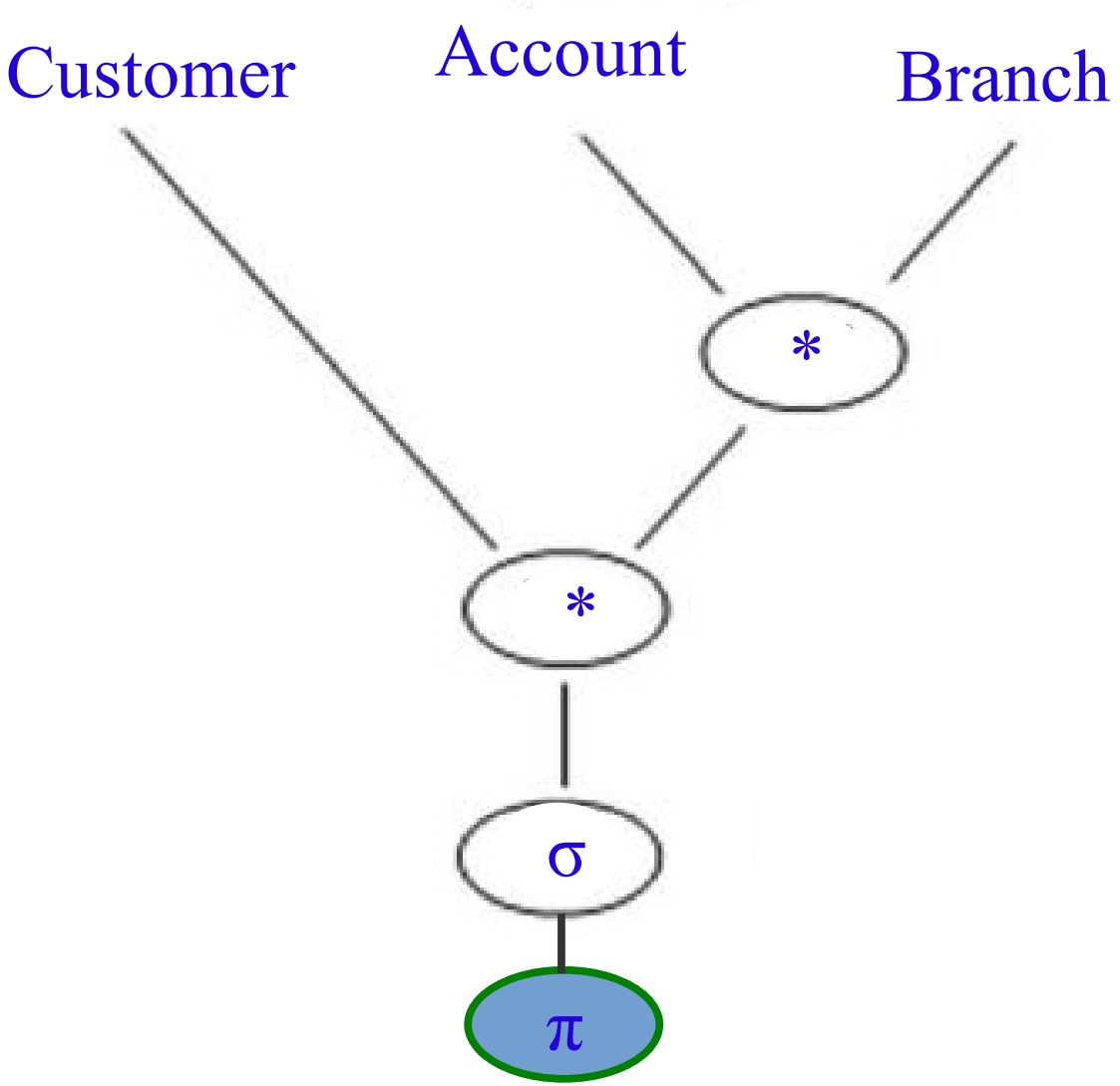
Cắt tỉa

3. $\pi_{cname}(Customer * (\sigma_{Bcity='Edina}(Account * Branch)))$



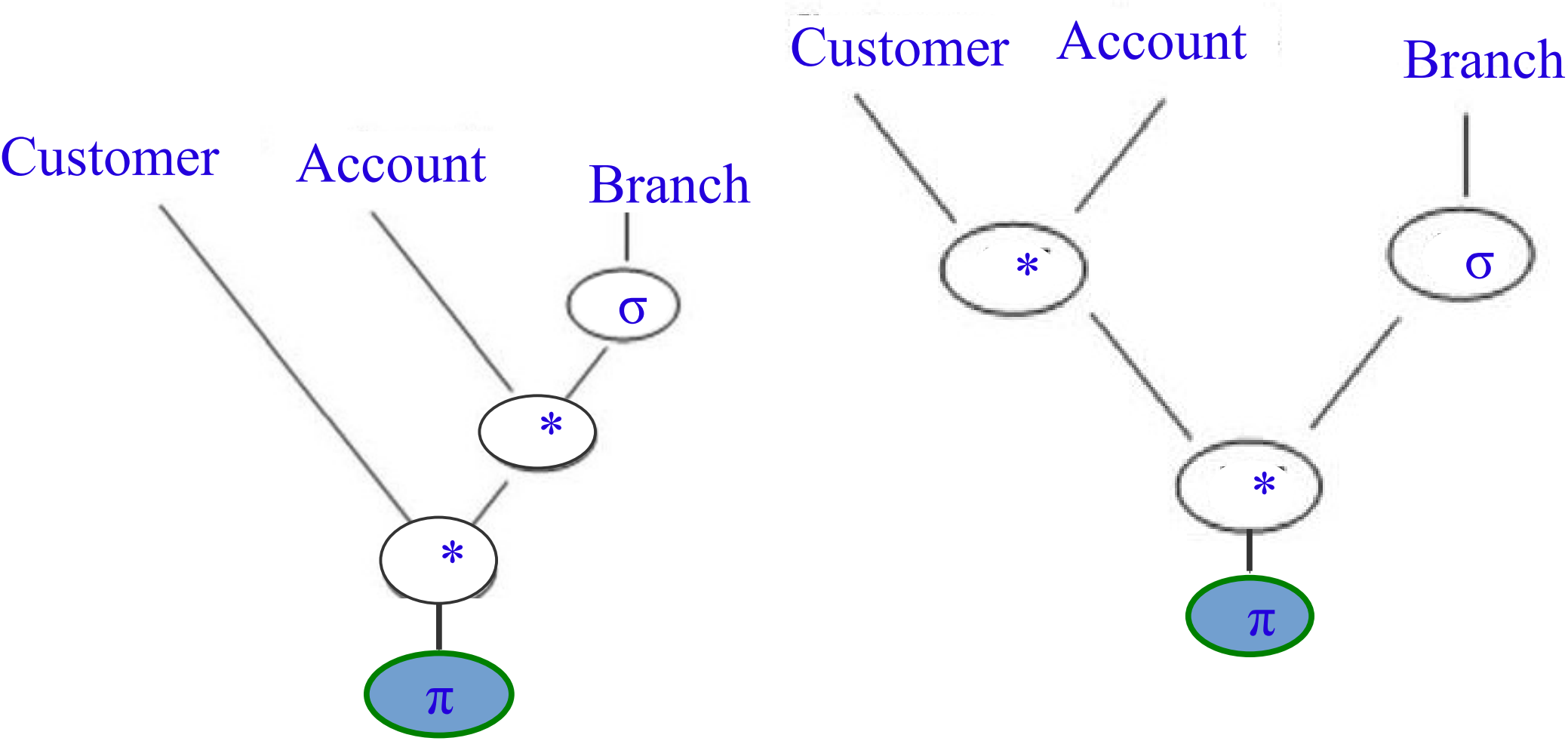
Cắt tỉa

4. $\pi_{cname}(\sigma_{Bcity='Edina'}(Customer * (Account * Branch)))$



Cắt tỉa

5. $\pi_{cname}(Customer * (Account * (\sigma_{Bcity='Edina'}(Branch))))$



Cắt tỉa

- Áp dụng luật các phép chọn chiếu cần thực hiện trước kết nối lên các cây biểu thức ĐSQH

  => **Các trường hợp 2 và 5 giữ lại**, các trường hợp khác bị loại

- **Bước 2: Phân tích chi phí => sử dụng các dữ liệu thống kê**

Phân tích chi phí

- **Bước 2: Phân tích chi phí => sử dụng các dữ liệu thống kê**

  Cho:

  - Có 500 khách hàng trong ngân hàng.
  - Trung bình, mỗi khách hàng có hai tài khoản.
  - Có 100 chi nhánh trong ngân hàng.
  - Có 10 chi nhánh tại thành phố Edina.
  - 10% khách hàng có tài khoản tại các chi nhánh tại Edina.
  - Cần t đơn vị thời gian để xử lý mỗi bộ của mỗi quan hệ trong bộ nhớ

    => Có 1000 tài khoản trong ngân hàng và 100 tài khoản tại các chi nhánh ở Edina

Phân tích chi phí

$\pi_{cname}((Customer * Account) * (\sigma_{Bcity='Edina'}(Branch)))$

Chi phí cho trường hợp 2

Phép toán	Chi phí	Số dòng
Customer * Account → R1	500 * 1000t	1000 bộ
$\sigma_{Bcity='Edina'}$(Branch) → R2	100t	10 bộ
R1 * R2 → Kết quả	1000 * 10t	100 bộ
Tổng chi phí	**510.100t**	

$\pi_{cname}(Customer * (Account * (\sigma_{Bcity='Edina'}(Branch))))$

Chi phí cho trường hợp 5

Phép toán	Chi phí	Số dòng
$\sigma_{Bcity='Edina'}$(Branch) → R1	100t	10 bộ
R1 * Account → R2	10 *1000t	100 bộ
R2 * customer → Kết quả	100 * 500t	100 bộ
Tổng chi phí	**60.100t**	

=> Cây trường hợp 5 được chọn

=> Luôn thực hiện các phép toán thu hẹp quan hệ trước các phép toán mở rộng quan hệ

Bài tập 1: Ước lượng chi phí

- **Cho lược đồ**:
  RAP(**ten**, diachi, quanly)
  PHONG(**maP**, *ten*, socho)
- Có 300 bộ trong RAP.
- Có 1200 bộ trong PHONG.
- Giả sử 5% các phòng có trên 150 chổ
- Cần t đơn vị thời gian để xử lý mỗi bộ của mỗi quan hệ trong bộ nhớ

- **Câu truy vấn**: Địa chỉ các rạp chiếu phim có phòng trên 150 chỗ ngồi ?

SQL

```
SELECT diachi
FROM RAP, PHONG
WHERE RAP.ten = PHONG.ten
        AND  socho > 150
```

Bài tập 1: Ước lượng chi phí

- Dịch câu lệnh sau sang ĐSQH.

```
SELECT diachi
FROM RAP, PHONG
WHERE RAP.ten = PHONG.ten
        AND  socho > 150
```

- *Biểu diễn các biểu thức ĐSQH dưới dạng cây biểu thức. Hãy phân tích chi phí của 2 biểu thức trên và cho biết biểu thức nào tốt hơn ?*

Bài tập 2: Ước lượng chi phí

- **Cho lược đồ**:

VANG (**v#**, ten, giong_nho), tên vang là duy nhất
N_KTRA (**i#**, tenKtra)
K_TRA(**v#, i#**, ngay, kq)

- Có 50 bộ trong VANG, Có 40 bộ trong N_KTRA, 500 bộ trong K_TRA
- Trung bình mỗi vang có 10 kiểm tra
- Cần t đơn vị thời gian để xử lý mỗi bộ của mỗi quan hệ trong bộ nhớ

Câu truy vấn:

1) Tên người kiểm tra đã kiểm tra vang tên aligoté ?
2) Tên vang có test "không tốt", biết rằng trung bình cứ 10 kiểm tra thì có 3 kiểm tra là không tốt?

Yêu cầu :

Viết biểu thức SQL, sinh các biểu thức ĐSQH tương đương câu SQL, biểu diễn các biểu thức ĐSQH dưới dạng cây biểu thức, áp dụng cắt tỉa cho câu 1, ước lượng chi phí.

Tối ưu hoá câu truy vấn

- Tối ưu hoá gồm ba bước:
  - Ước lượng chi phí,
  - **Sinh ra kế hoạch (plan) thực thi**
  - Sinh mã câu truy vấn.

Sinh plan thực thi

- Trong bước tối ưu hóa này, kế hoạch truy vấn hay kế hoạch thực thi (đơn giản là plan) được tạo ra.
- Một plan thực thi là một sự mở rộng cây truy vấn (cây ĐSQH) bao gồm **đường dẫn truy cập** (Access path) cho tất cả các phép toán trong cây.
  - Đường dẫn truy cập cung cấp thông tin chi tiết về cách mỗi phép toán trong cây được thực hiện.

Sinh plan thực thi

- Đối với mỗi câu truy vấn, hệ thống lựa chọn giữa nhiều plan thực thi và chọn plan tốt nhất.
- Các plan khác nhau về trình tự các phép toán, đường dẫn truy cập.
- Chọn plan tốt nhất:
  - Xây dựng các plan có thể
  - Đánh giá chi phí của chúng. Đối với mỗi plan, cần ước tính chi phí cho mỗi phép toán và kích thước của kết quả (**Mục tiêu là** giảm càng nhanh càng tốt kích thước của các dữ liệu được thao tác).
  - Chọn plan tốt nhất. ***Vấn đề****: Có nhiều plan có thể*
  - Sử dụng heuristics để tránh khai thác toàn bộ các khả năng.

Đường dẫn truy cập

- Ví dụ: đường dẫn truy cập của phép kết nối có thể là:
  - nối kết dùng vòng lặp lồng nhau (Nested-Loop Join)
  - nối kết dùng hàm băm (hash join)
  - nối kết dùng sắp xếp – trộn (sort merge join)

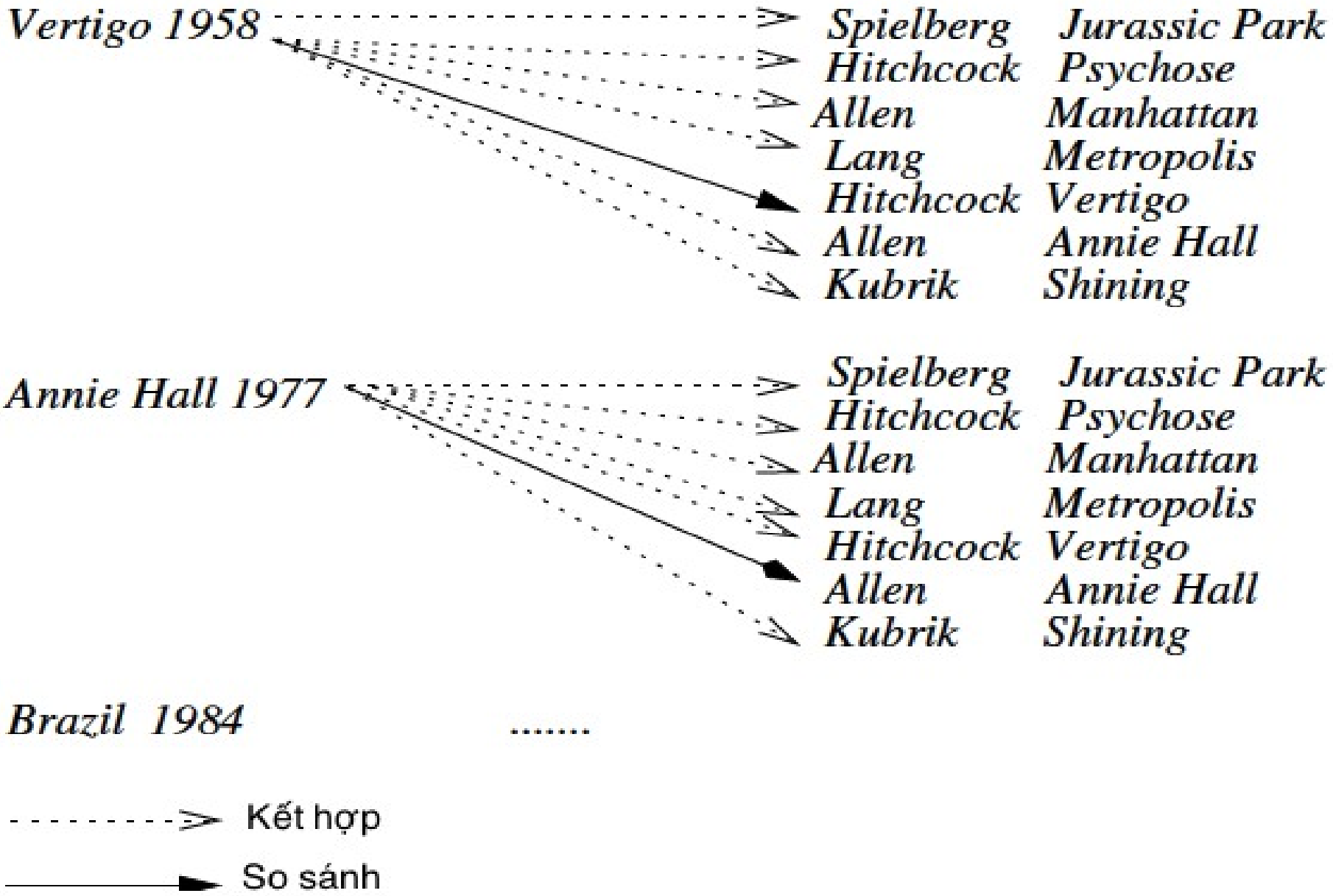
  => kết nối là phép toán tốn nhiều chi phí nhất

Kết nối dùng vòng lặp lồng nhau

```
For Each r in R Do /* r is a tuple in R */
    For Each s in S Do /* s is a tuple in S */
        If r.a = s.b
        then add r||s to the output
    Next s
Next r
```

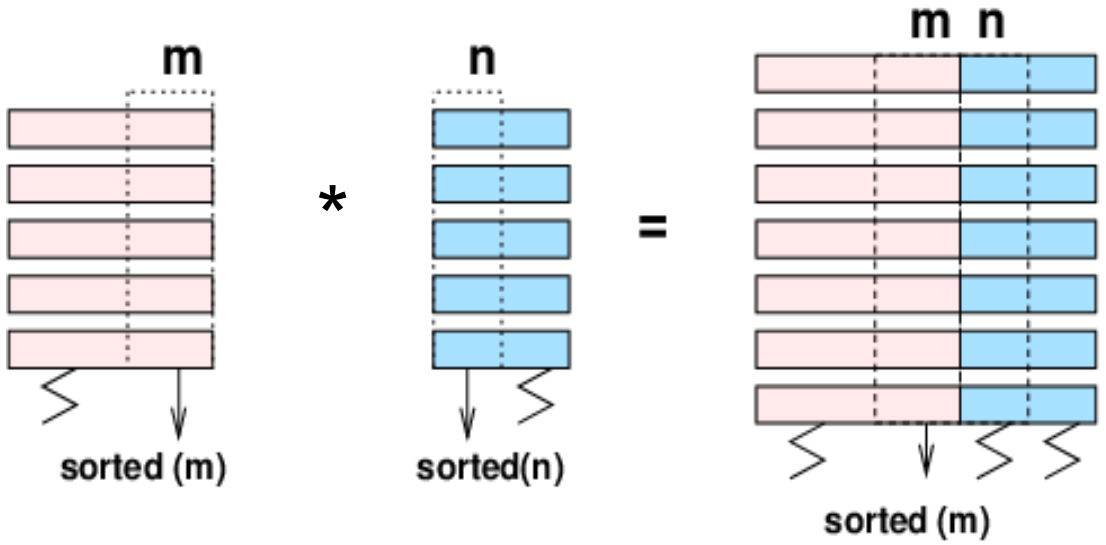
"||" biểu diễn cho phép toán nối.

Kết nối dùng Vòng lặp lồng nhau

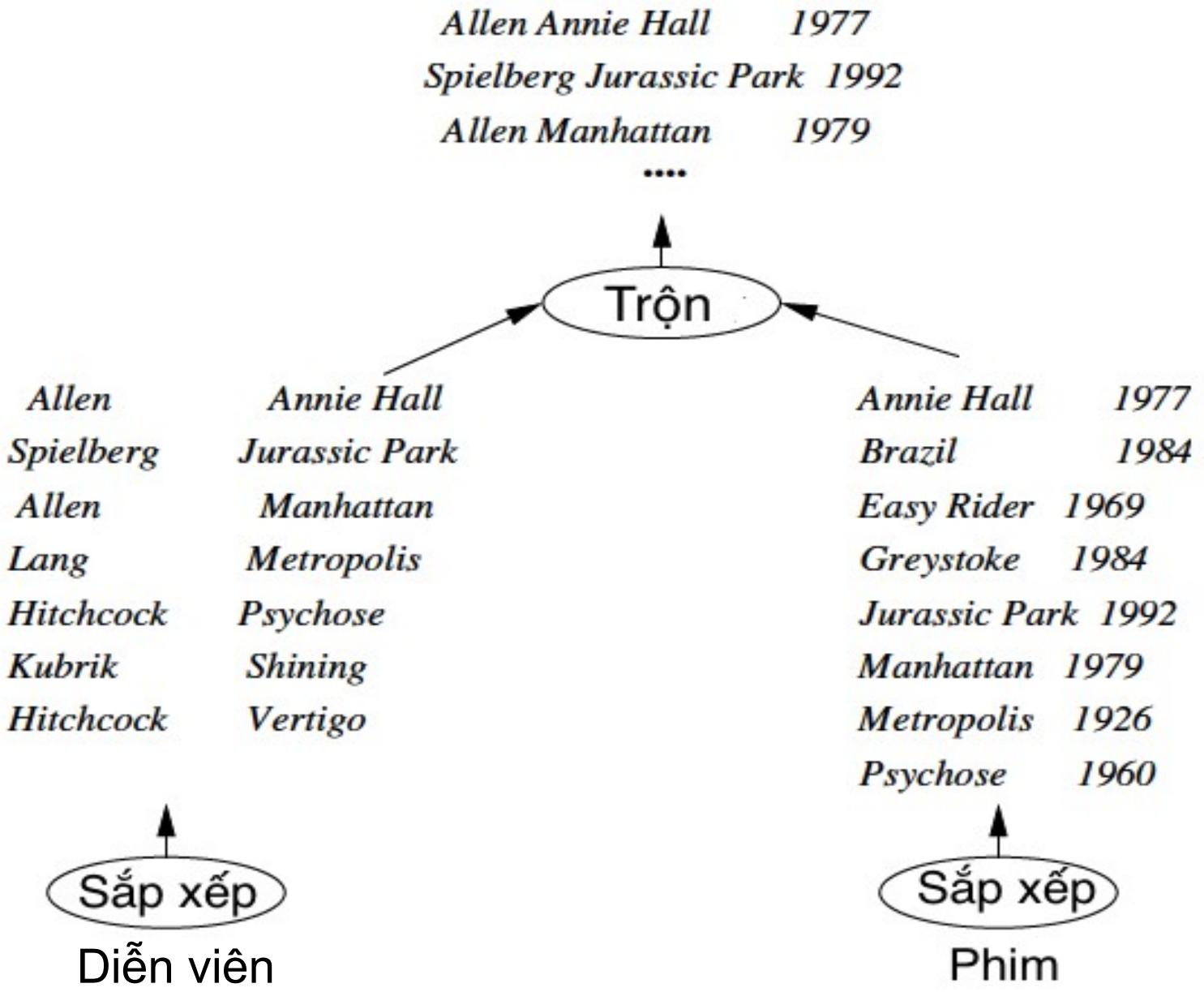


Kết nối dùng sắp xếp - trộn

- Hiệu quả hơn vòng lặp lồng nhau đối với các bảng kích thước lớn
  - Sắp xếp 2 bảng trên thuộc tính kết nối
  - Trộn 2 bảng
- Chính giai đoạn sắp xếp sẽ tiêu tốn nhiều chi phí

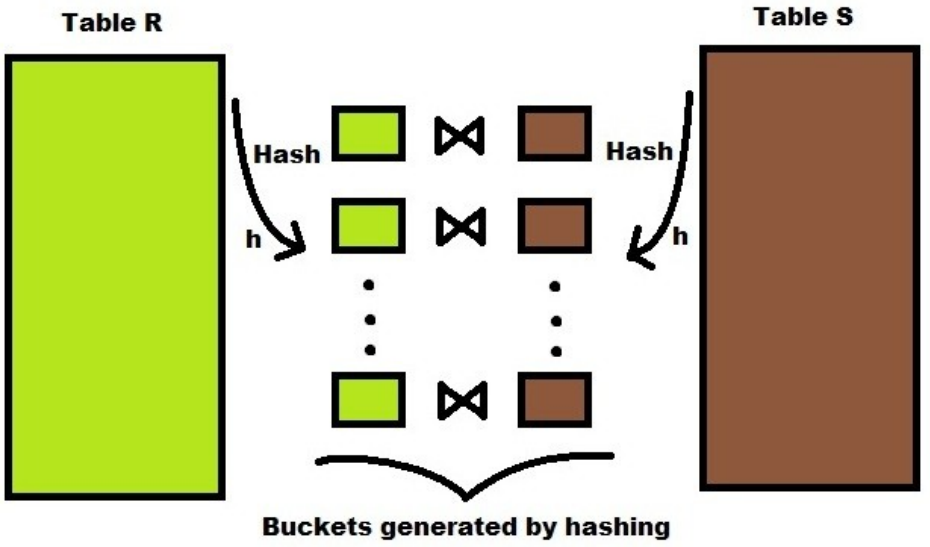


Kết nối dùng xắp xếp - trộn



Kết nối dùng hàm băm (hash join)

- Về lý thuyết, giải thuật này thường là hiệu quả nhất.
- Thực hiện nhanh khi một trong hai bảng là nhỏ (1, 2, 3 lần kích thước của bộ nhớ).
- Không mạnh (hiệu quả phụ thuộc vào nhiều yếu tố).

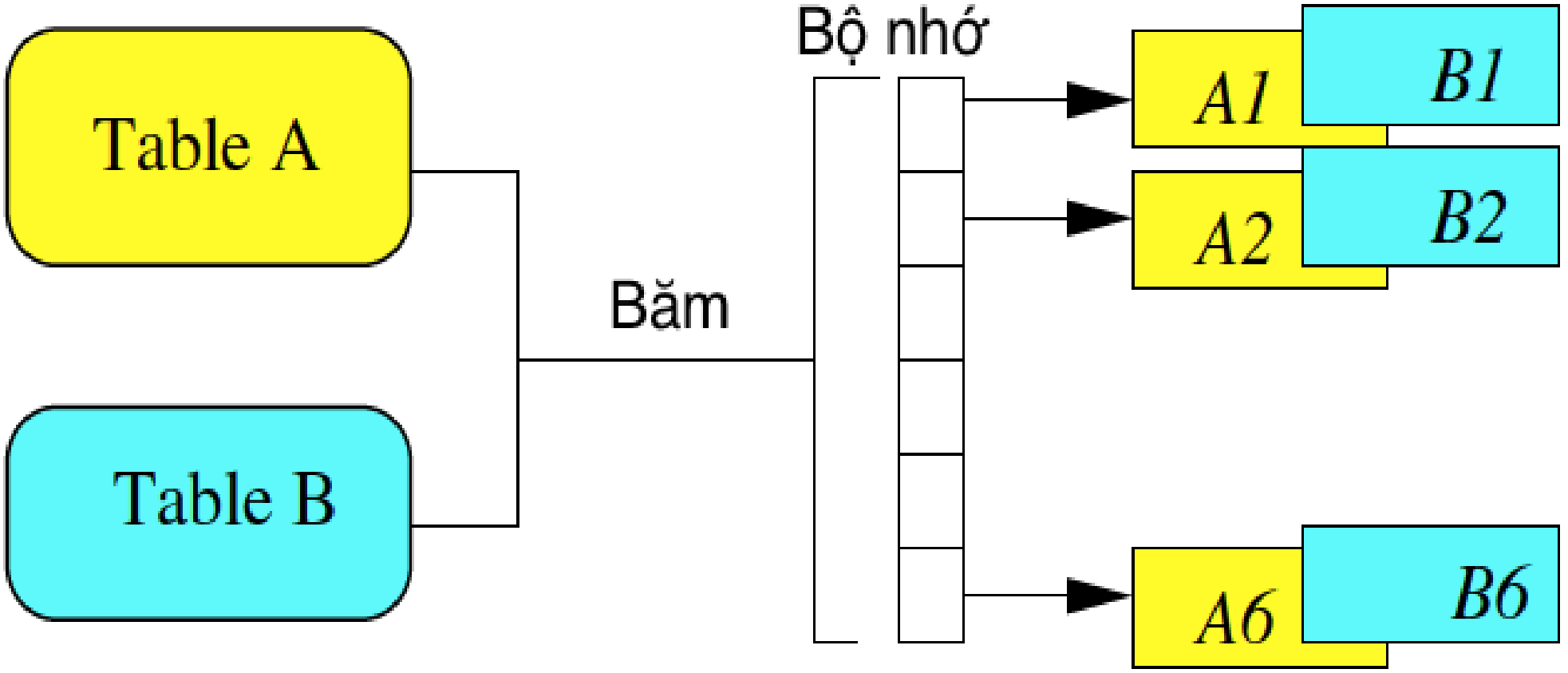


Kết nối dùng hàm băm (hash join)

- Đặc điểm chính: gồm 2 giai đoạn băm và kết nối
  - Băm:
    - Băm bảng nhỏ hơn (A) trong hai bảng (A và B) thành n đoạn. Mỗi bộ được gán vào 1 đoạn bằng hàm băm. Thuộc tính nối kết là tham số hàm băm
    - Băm bảng còn lại (B), cùng cách như A, thành n đoạn khác
  - Kết nối
    - Các đoạn của A và B được kết hợp thành cặp, và thực hiện kết nối.
    - **Các bộ trong một đoạn của A chỉ so sánh với các bộ trong đoạn tương ứng của B => thực hiện nhanh**

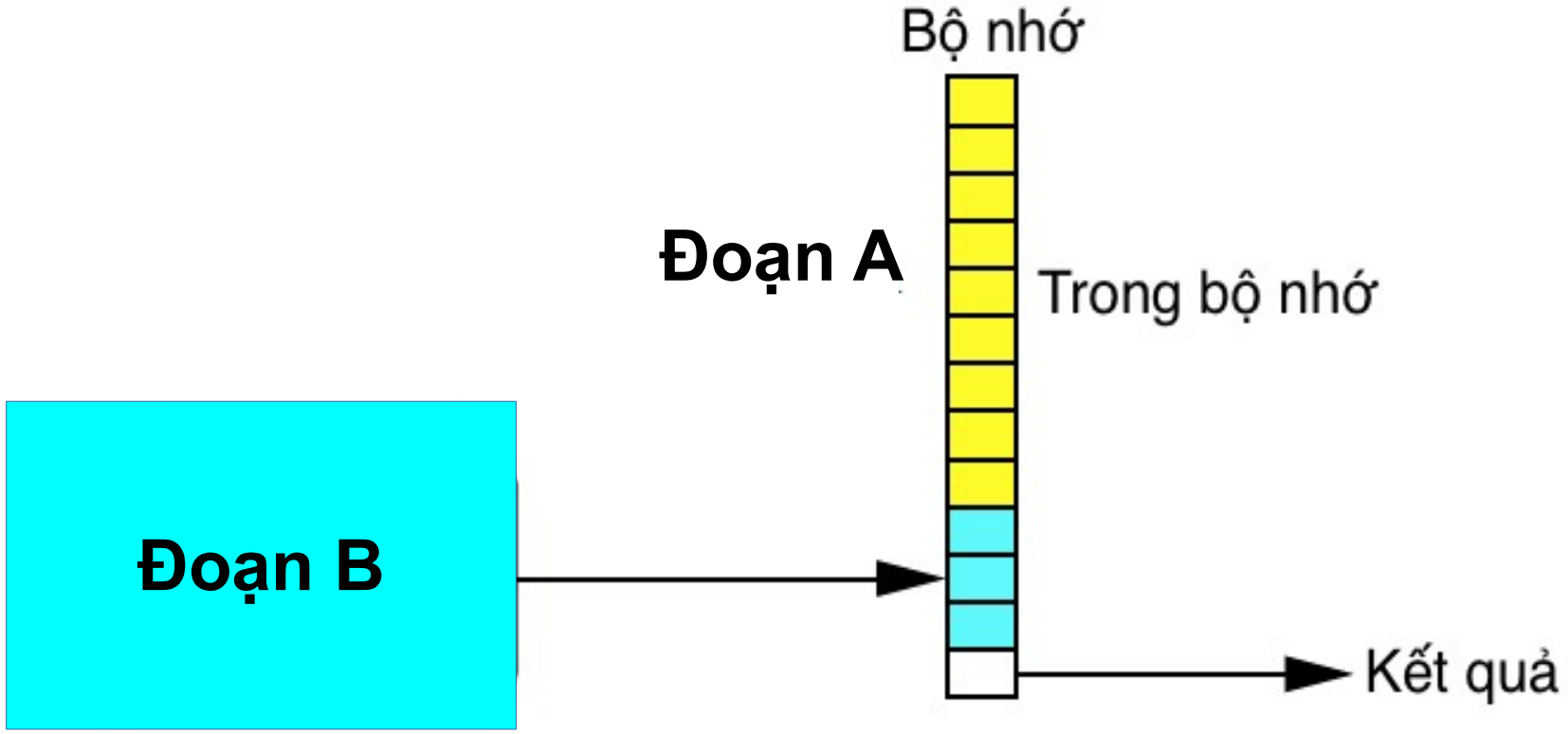
Kết nối dùng hàm băm (hash join)

- Giai đoạn băm:



Kết nối dùng hàm băm (hash join)

- Giai đoạn kết nối:



Kết nối dùng vòng lặp có chỉ mục

- Giả sử bảng **A có chỉ mục và B không có chỉ mục**
- Duyệt qua tất cả các bộ của bảng không có chỉ mục (bảng B) trước
  - Với mỗi bộ, sử dụng chỉ mục trên A và kiểm tra xem giá trị thuộc tính nối kết của B có trong chỉ mục của A không.
  - Nếu tìm thấy, 2 bộ sẽ được nối kết thành bộ mới và thêm bộ mới này vào kết quả

=> Loại bỏ việc phải duyệt qua tất cả các bộ của A

Kết nối dùng vòng lặp có chỉ mục

- Giả sử bảng **A và B đều có chỉ mục trên thuộc tính nối kết**
  - Thực hiện như trường hợp có 1 chỉ mục bởi vì chỉ có 1 chỉ mục được dùng
  - Hoặc sử dụng kết nối sắp xếp - trộn

Ví dụ đường dẫn truy cập

- Xét 2 quan hệ R và S với các thông tin sau:
  - Có "M = 1000" các trang đĩa trong R.
  - Có "N = 500" các trang đĩa trong S.
  - R có K=100.000 bộ
  - S có L=40.000 bộ
  - Tất cả các bộ đều nhỏ hơn kích thước 1 trang đĩa, và để đơn giản hóa tính toán, giả sử rằng một hoặc nhiều bộ vừa khích trong 1 trang đĩa để không chồng chéo hoặc lãng phí không gian.
  - Sử dụng ba bộ đệm (buffer) để xử lý các bộ xử lý bộ trong bộ nhớ; mỗi bộ đệm chỉ lớn bằng một trang đĩa.
  - Buffer đầu tiên để xử lý các trang của R, cái thứ hai để xử lý các trang của S, và cái thứ ba để chuẩn bị các bộ của kết quả.
  - Khi cái thứ 3 đầy, nội dung của nó được ghi lên trang đĩa để chuẩn bị chỗ trống cho các bộ kết quả khác.

Ví dụ đường dẫn truy cập

- **Kết nối sử dụng vòng lặp**:
  - Cần phải tốn M thao tác I/O đĩa để mang tất cả các trang của R (từng cái một) vào bộ nhớ.
  - Đối với mỗi bộ của R, chúng ta phải đưa tất cả các trang của S (từng cái một) vào bộ nhớ và sau đó kiểm tra từng bộ trong S.
  - Do đó, số lượng I/O đĩa sẽ là: M + K * N, với M=1000, K=100.000, N= 500 => chi phí thực tế là:

    "1000 + 100,000 * 500" =50.0001.000 I/O đĩa
  - Nếu mỗi I/O đĩa mất khoảng 9mili giây, thì cách tiếp cận này sẽ mất hơn 125 giờ để hoàn thành.
    => Đây là kết quả **không thể chấp nhận** cho hai quan hệ tương đối nhỏ ngay cả đối với đĩa tốc độ cao ngày nay.

Ví dụ đường dẫn truy cập

- **Kết nối sử dụng vòng lặp lồng nhau**:
  - Để giảm chi phí của vòng lặp lồng nhau, chúng ta có thể sử dụng một tiếp cận vòng lặp lồng nhau hướng khối (block-oriented nested-loop)
  - Trong cách tiếp cận này, chúng ta cũng phải mang tất cả các trang của R vào bộ nhớ.
  - Đối với mỗi trang của R, mang tất cả các trang trong S.
  - Trong trường hợp này, chi phí tham gia là:

1000 + 1000 * 500 = 501.000 I/O đĩa.

=> Ngay cả với sự cải tiến này, thời gian cần thiết là 75 phút.

Ví dụ đường dẫn truy cập

- **Kết nối sử dụng vòng lặp lồng nhau**:
  - Một cải tiến khác có thể cho tiếp cận hương khối là tăng số lượng các bộ đệm để lưu giữ tất cả các trang của R trong bộ nhớ và có hai bộ đệm bổ sung, một cho các trang của S và một cho đầu ra.
  - Trong trường hợp này, chi phí sẽ nhỏ:

    1000 + 500 = 1500 I/O đĩa (hoặc 0.225 phút).
  - Thực tế, chúng ta có thể không có đủ không gian vùng đệm để đặt tất cả các trang trong R vào bộ nhớ cùng một lúc. Trong trường hợp này chúng ta phải mang một tập hợp con các trang R vào bộ nhớ lcùng nhau

=> Các DBMS cấp phát một số lượng lớn buffer để kết nối các quan hệ tương đối lớn.

Ví dụ đường dẫn truy cập

- **Kết nối sử dụng hàm băm**:
  - Chi phí của kết nối sử dụng hàm băm là tổng chi phí của cả 2 giai đoạn băm và kết nối
  - Trong giai đoạn băm, mỗi quan hệ được đọc và được ghi lại vào đĩa. Do đó, chi phí giai đoạn này là "2 * (M + N)."
  - Trong giai đoạn kết nối, mỗi quan hệ được đọc lại một lần nữa. Do đó, tổng chi phí "3 * (M + N) = 3 * (1000 + 500) = 4500 I/O đĩa hoặc 40,5 giây.
  - So với vòng lặp lồng nhau thì chi phí cho tiếp cận này nhỏ hơn nhiều.

Tối ưu hoá câu truy vấn

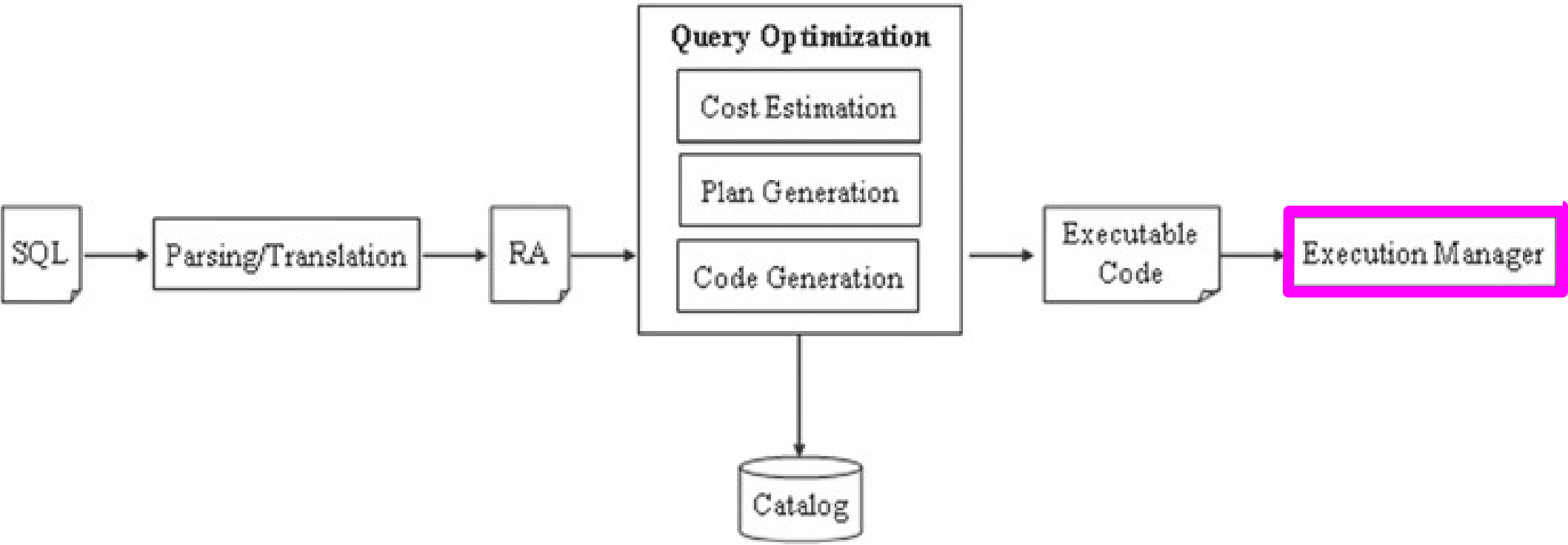
- Tối ưu hoá gồm ba bước:
  - Ước lượng chi phí,
  - Sinh ra kế hoạch (plan) thực thi
  - **Sinh mã câu truy vấn.**

Sinh mã câu truy vấn

- Bước cuối cùng trong tối ưu hoá truy vấn là sinh mã (code).
- Mã là đại diện cuối cùng của câu truy vấn và sẽ được thực thi hoặc thông dịch tùy thuộc vào loại hệ điều hành hoặc phần cứng.
- Mã truy vấn được chuyển sang bộ phận quản lý thực thi (EM - execution manager) để thực thi.

Đánh giá / thực thi

- Bước sau cùng trong quy trình xử lý câu truy vấn là câu truy vấn được thực thi và trả về kết quả



Nội dung

- Giới thiệu tổng quan về xử lý câu truy vấn
- Xử lý câu truy vấn tập trung
- **Xử lý truy vấn phân tán**

Xử lý câu truy vấn phân tán

- *Là sự chuyển đổi một câu truy vấn cấp cao (SQL) trên cơ sở dữ liệu phân tán (một tập các quan hệ toàn cục) thành một câu truy vấn cấp thấp tương đương và hiệu quả hơn (đại số quan hệ) trên các đoạn quan hệ.*
- Xử lý truy vấn phân tán phức tạp hơn vì:
  - Phân đoạn / nhân bản các quan hệ
  - Chi phí truyền thông: là thời gian cần thiết để trao đổi dữ liệu giữa các nút tham gia thực hiện câu truy vấn.
  - Thực thi song song
- Tối ưu hóa truy vấn nhằm mục đích giảm tối thiểu hàm chi phí:

  Chi phí I / O + chi phí CPU + chi phí truyền thông → Min

Xử lý câu truy vấn phân tán

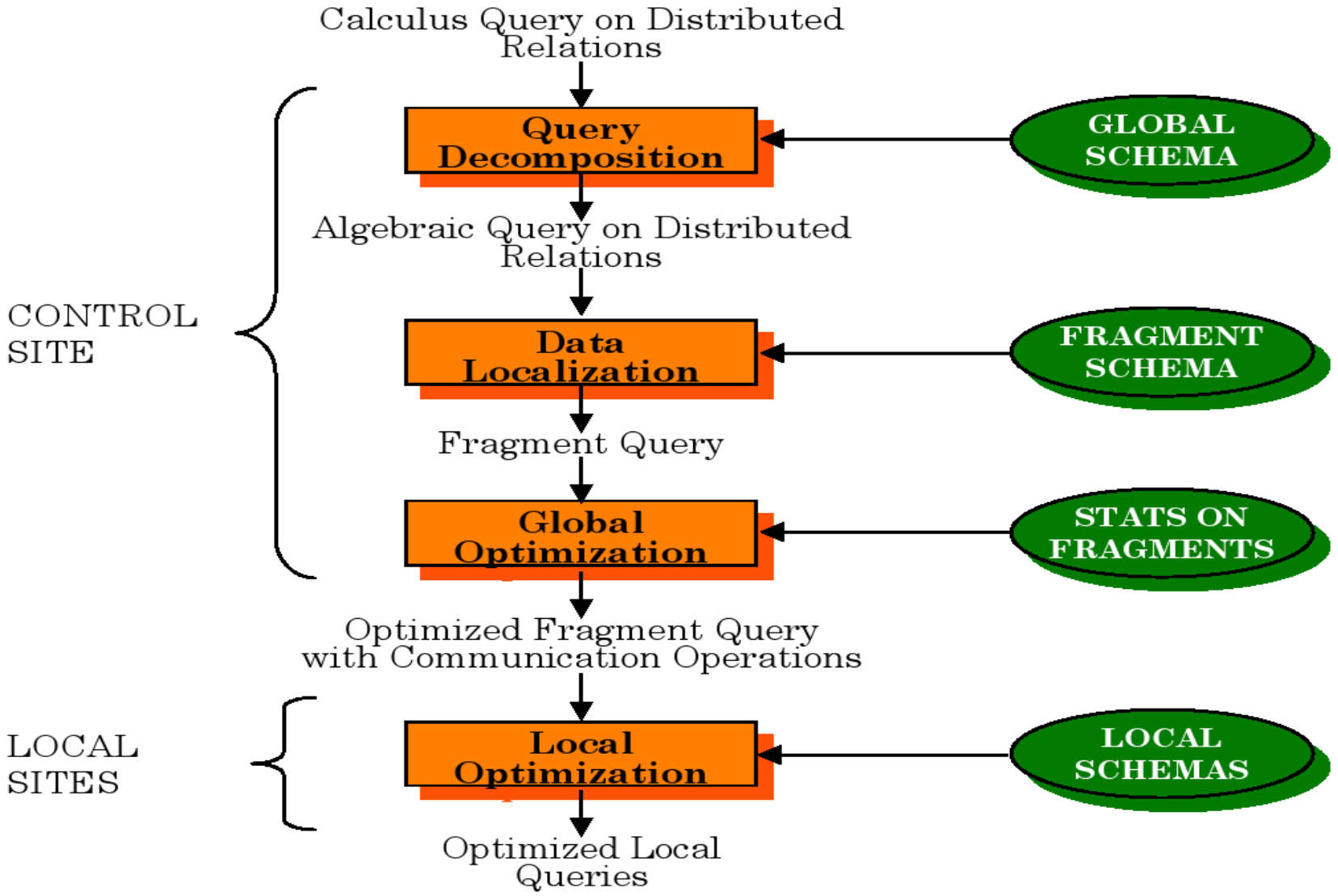
- Trong môi trường phân tán, xử lý một câu truy vấn bao gồm
  - tối ưu hóa và
  - sinh plan

  } *ở mức toàn cục cũng như tại cơ sở dữ liệu cục bộ*
- Nút nơi câu truy vấn được nhập vào hệ thống được gọi **nút khách hàng hoặc nút điều khiển**.
- Các **nút khách hàng** cần
  - Xác thực người dùng hoặc ứng dụng muốn truy cập vào các quan hệ bao gồm trong câu truy vấn;
  - Kiểm tra cú pháp của câu truy vấn và từ chối nếu nó chưa đúng;
  - Dịch câu truy vấn sang đại số quan hệ
  - Tối ưu hóa toàn cục câu truy vấn =>chọn BT tốt nhất;.

Xử lý câu truy vấn phân tán

- Câu truy vấn toàn cục được dựa trên lược đồ toàn cục.
- Các quan hệ được sử dụng trong câu truy vấn toàn cục có thể được phân tán (phân đoạn và /hoặc nhân bản) trên nhiều CSDL cục bộ.
- Mỗi CSDL cục bộ chỉ làm việc với khung nhìn cục bộ tại nút của nó và không biết dữ liệu được lưu trữ tại nút của nó có liên quan như thế nào đến khung nhìn toàn cục.
- **Nút điều khiển** chịu trách nhiệm sử dụng tự điển dữ liệu toàn cục (GDD) để xác định thông tin phân phối và xây dựng lại khung nhìn toàn cục từ các đoạn vật lý cục bộ.

Xử lý câu truy vấn phân tán



Source: (J. Gamper, 2009)

Phân rã - Decomposition

- Giống như trong các hệ thống tập trung
  - Chuẩn hoá (normalization)
  - Loại bỏ dư thừa và
  - Viết lại biểu thức ĐSQH

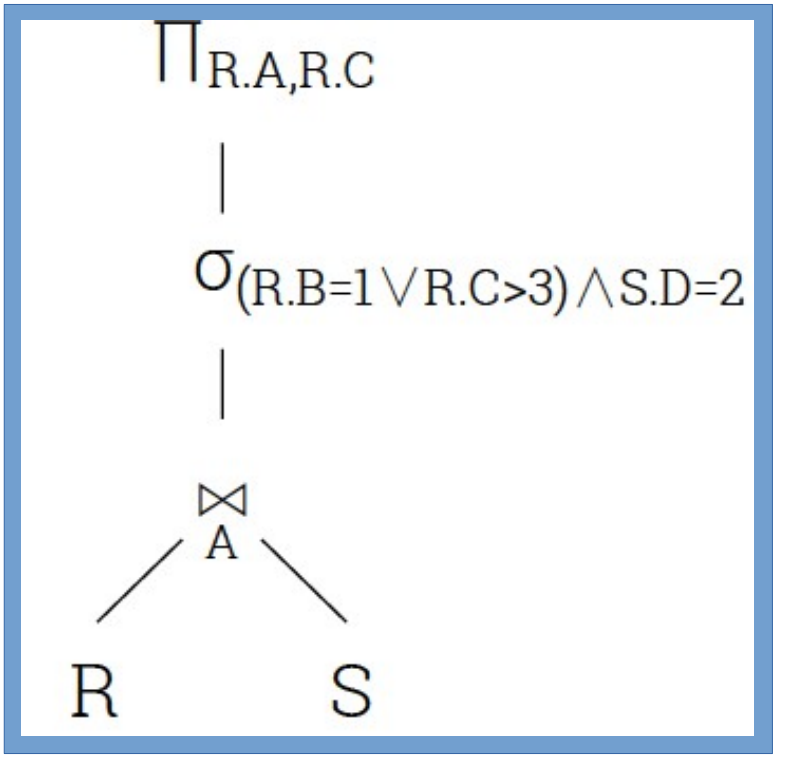
=> Sinh ra một hoặc nhiều biểu thức ĐSQH

=> Sinh ra biểu thức ĐSQH tối ưu.

Chuẩn hóa

- Kiểm tra cú pháp
- Chuyển đổi từ ngôn ngữ truy vấn chung (SQL) sang dạng chuẩn (đại số quan hệ)
- **Ví dụ:**

```
SELECT  R.A, R.C
FROM R, S
WHERE R.A = S.A  AND
    ((R.B = 1 AND S.D = 2) OR
     (R.C > 3 AND S.D = 2))
```



Loại bỏ dư thừa

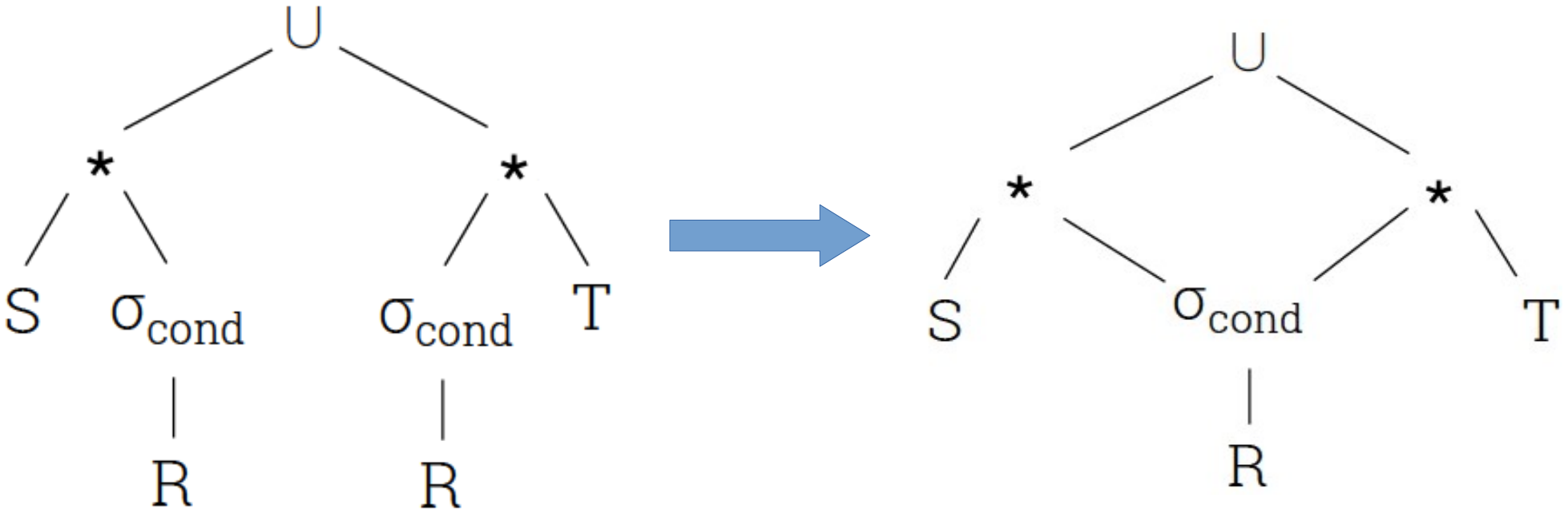
- Loại bỏ dư thừa hoặc các phép toán không cần thiết, ví dụ trong các điều kiện
- Ví dụ:
  - $(S.A = 1) \wedge (S.A > 5) \Longrightarrow$ false
  - $(S.A < 10) \wedge (S.A < 5) \Longrightarrow S.A < 5$

Loại bỏ dư thừa

- Áp dụng các luật chuyển đổi
  - p ∧ p <=> p
  - p ∨ p <=> p
  - p ∧ true <=> p
  - p ∨ f alse <=> p
  - p ∧ f alse <=> f alse
  - p ∨ true <=> true
  - p ∧ ¬p <=> f alse
  - p ∨ ¬p <=> true
  - p1 ∧ (p1 ∨ p2) <=> p1
  - p1 ∨ (p1 ∧ p2) <=> p1

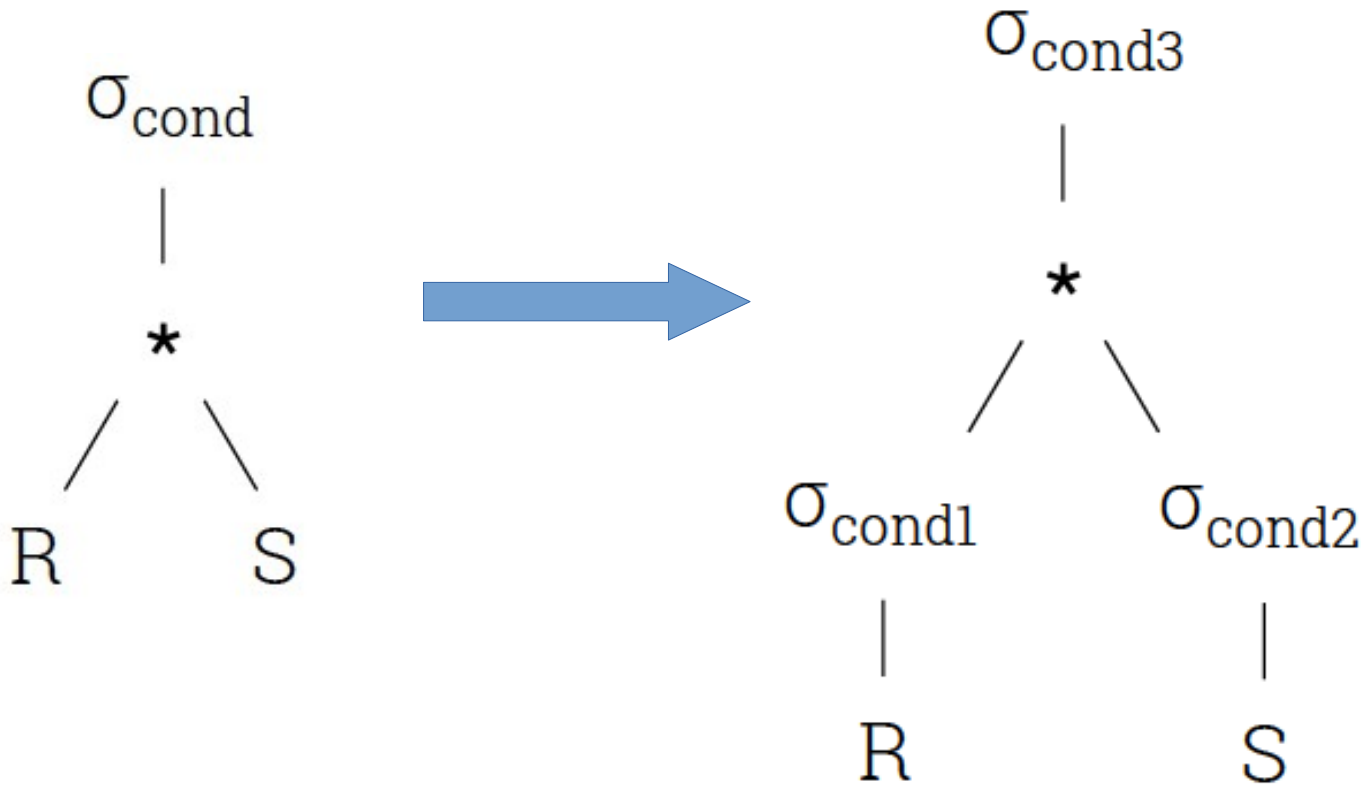
Loại bỏ dư thừa

- Ví dụ:



Viết lại biểu thức ĐSQH

- Áp dụng các toán tử một ngôi càng sớm càng tốt
- Biến đổi tích Cartesian thành kết nối
- Sử dụng chủ yếu các tính chất của đại số quan hệ



Localization

- Thay thế trong cây ĐSQH toàn cục: thay các quan hệ toàn cục bằng các đoạn tương ứng
    - Tối ưu hóa cây mới với cùng nguyên tắc tối ưu
    - Sử dụng sơ đồ thay thế dữ liệu
    - Sinh ra cây ĐSQH cục bộ
- Gồm 4 bước:
    ① Bắt đầu với truy vấn
    ② Thay thế quan hệ bằng các đoạn
    ③ Đẩy phép toán ∪ lên và Π, σ xuống
    ④ Loại bỏ các phép toán không cần thiết

Localization

- Ký hiệu: [R:đk] với:
  - ***R**: đoạn*
  - ***đk**: điều kiện mà các bộ của đoạn R thỏa hay điều kiện để phân đoạn ngang R*

Localization

- Luật 1: Loại bỏ truy xuất đến các đoạn không hợp lệ

σ_{c1} [R: c2] $\Longrightarrow$ σ_{c1} [R: c1$\wedge$c2]

[R: false] $\Longrightarrow$ $\emptyset$

Ví dụ:

$\sigma_{E=3}$ [R2: E $\geq$ 10] $\Longrightarrow$ $\sigma_{E=3}$ [R2: E=3$\wedge$E $\geq$ 10]

$\Longrightarrow$ $\sigma_{E=3}$ [R2: false]

$\Longrightarrow$ $\emptyset$

Localization

- **Luật 2**: A là thuộc tính nối kết giữa R và S

$$[R: c1] * [S: c2] \Longrightarrow [R * S: c1 \wedge c2 \wedge R.A=S.A]$$

$$[R: false] \Longrightarrow \emptyset$$

Ví dụ: A là thuộc tính nối kết giữa R1 và S2

$$[R1: A < 5] * [S2: A \geq 5]$$

$$\Longrightarrow [R1^* S2: R1.A < 5 \wedge S2.A \geq 5 \wedge R1.A=S2.A]$$

$$\Longrightarrow [R1^* S2: false] \Longrightarrow \emptyset$$

Localization

- **Luật 3**: Phân đoạn dọc: Loại bỏ các truy cập vào các đoạn mà không có các thuộc tính có ích cho kết quả cuối cùng

$R_i = \Pi_{Ai}(R), A_i \subseteq U$
Với mọi $B \subseteq U$
$\Pi_B(R) = \Pi_B(* R_i \mid B \cap A_i \neq \emptyset)$

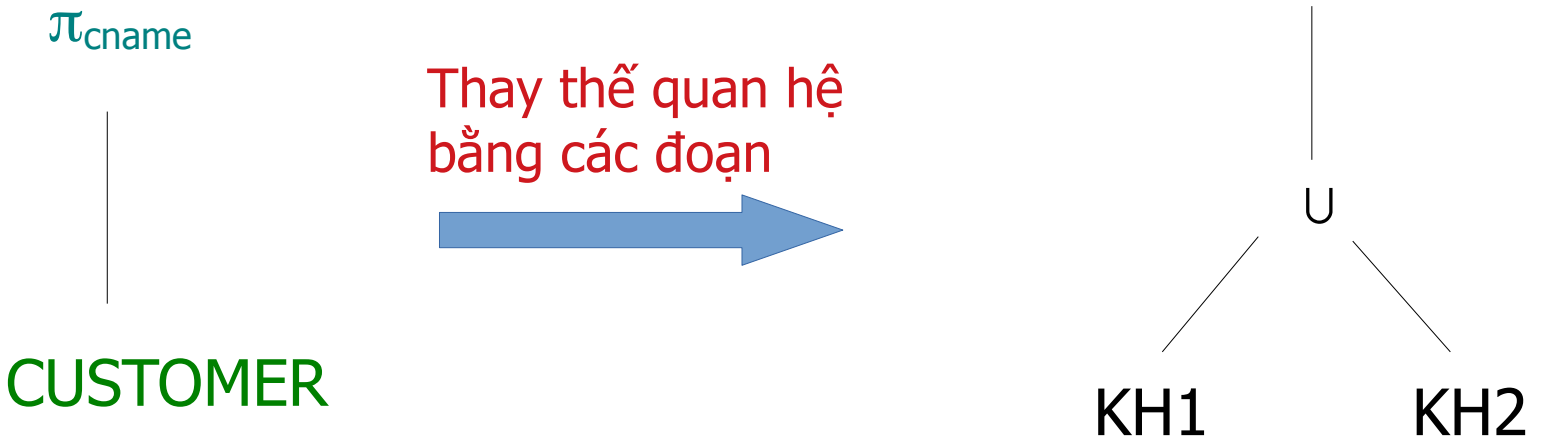
Localization

- Ví dụ: SELECT distinct cname
  FROM CUSTOMER

$KH_1 = \sigma_{ccity = 'Cantho'}(CUSTOMER)$

$KH_2 = \sigma_{ccity \neq 'Cantho'}(CUSTOMER)$

$CUSTOMER = KH_1 \cup KH_2$



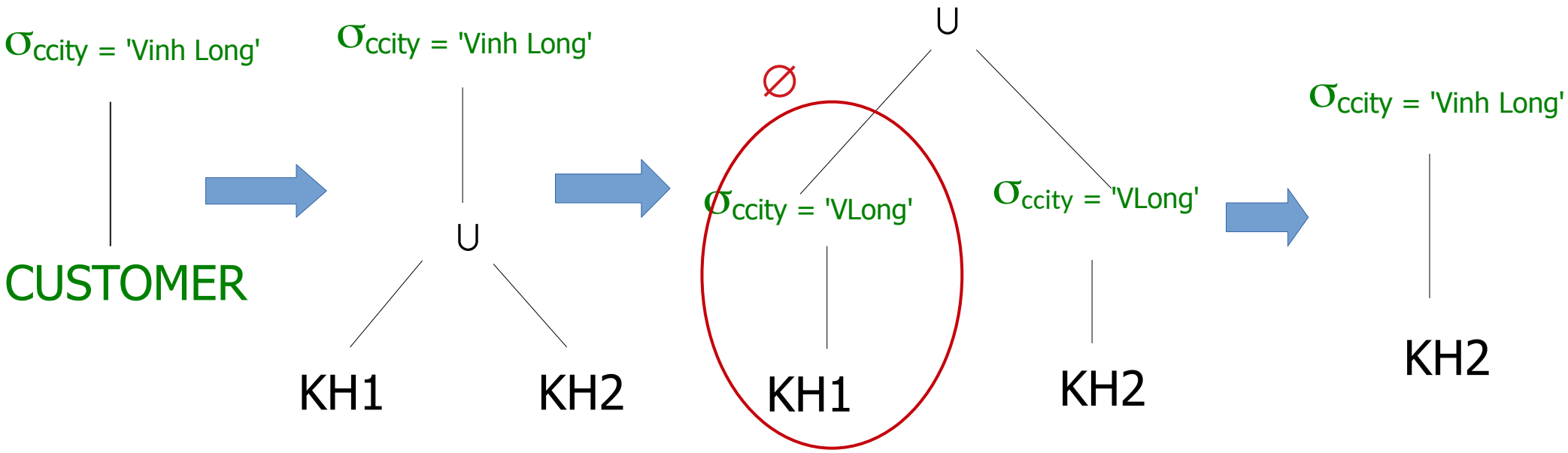
Localization

- Ví dụ: **Loại bỏ các đoạn ngang không hợp lệ**

$$\sigma_{ccity = 'Vinh\ Long'}(CUSTOMER)$$

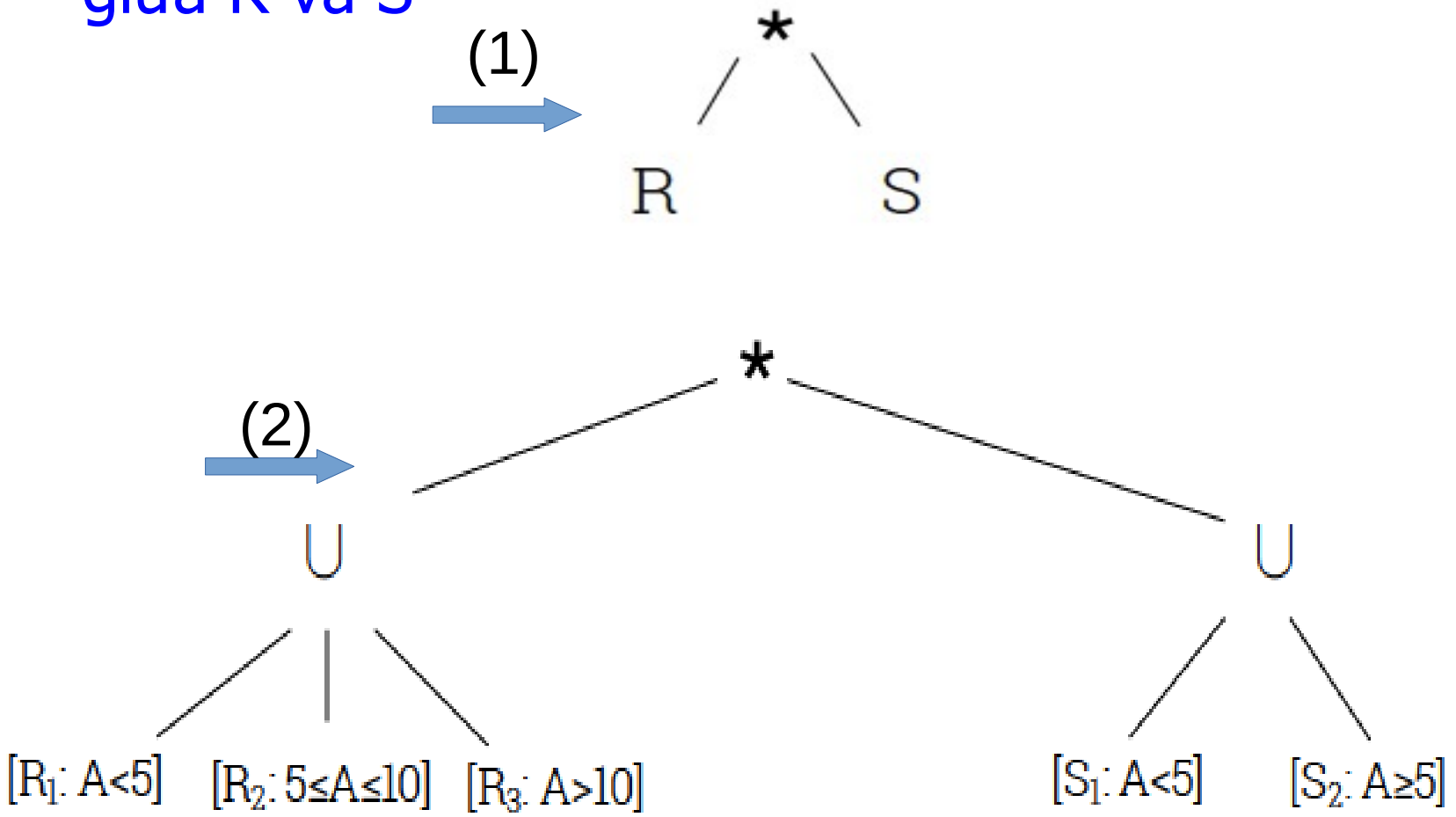
$$KH_1 = \sigma_{ccity = 'Cantho'}(CUSTOMER)$$
$$KH_2 = \sigma_{ccity \neq 'Cantho'}(CUSTOMER)$$

$$CUSTOMER = KH_1 \cup KH_2$$



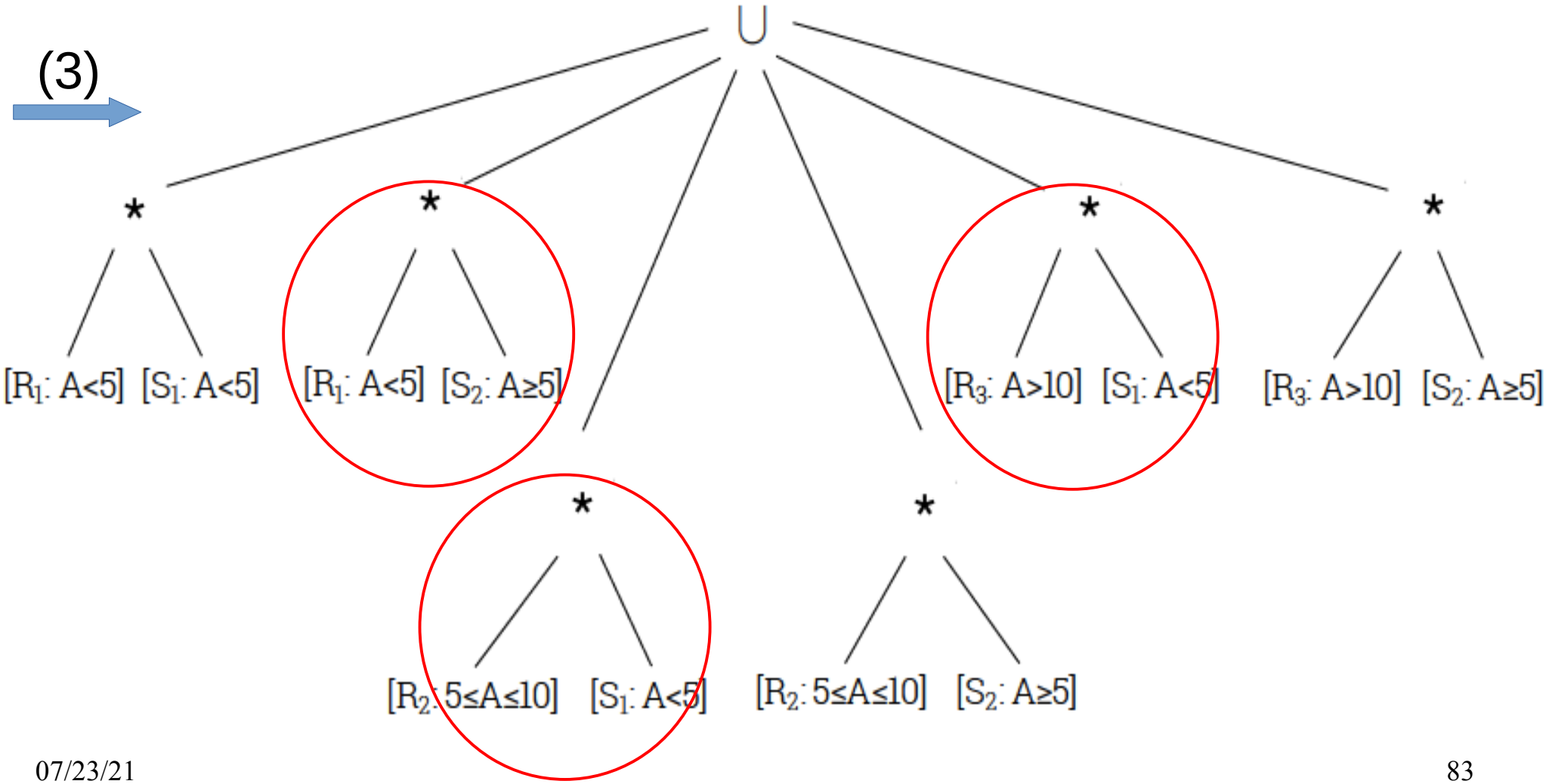
Localization

- **Ví dụ 1- Phân đoạn ngang**: A là thuộc tính nối kết giữa R và S



Localization

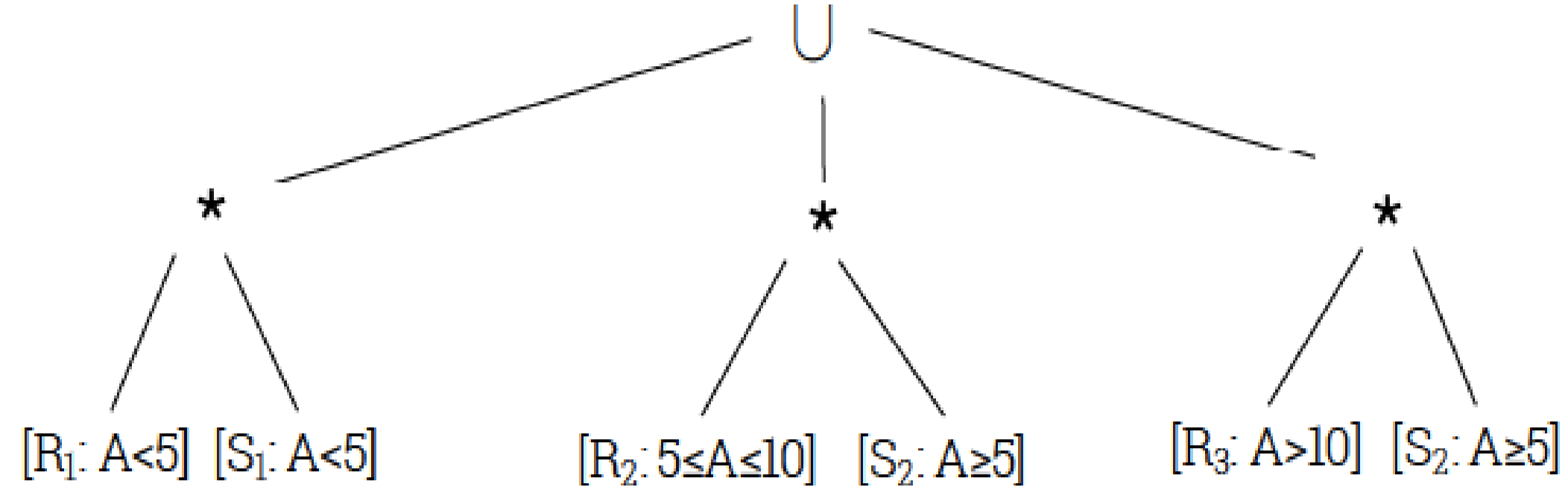
- **Ví dụ 1**: A là thuộc tính nối kết giữa R và S



Localization

- **Ví dụ 1**: A là thuộc tính nối kết giữa R và S

(4)



Localization

- **Ví dụ 2- Phân đoạn dọc** $\Pi_{cname}(CUSTOMER)$

  CUSTOMER phân thành 2 đoạn

  K_1 (cid, cname)

  K_2 (cid, street, ccity)

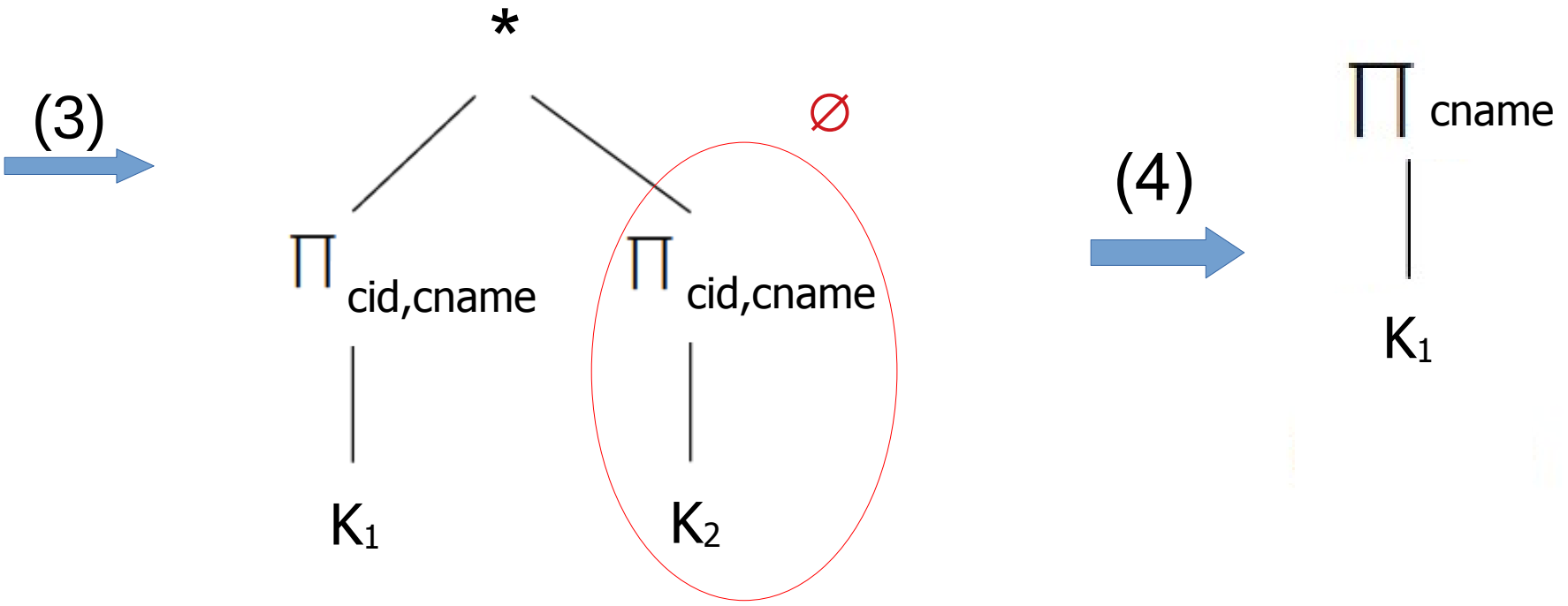
  } CUSTOMER= $K_1 * K_2$

(1) → Π cname | CUSTOMER

(2) → Π cname | * | K_1 K_2

Localization

- **Ví dụ 2- Phân đoạn dọc**



Localization

- **Bài tập. Cho các đoạn sau và câu truy vấn**

  $KH_1 = \sigma_{ccity = 'Cantho'}(Customer)$

  $KH_2 = \sigma_{ccity <> 'Cantho'}(Customer)$

  $AC_1 = Account \ltimes KH_1$

  $AC_2 = Account \ltimes KH_2$

- **Câu truy vấn**

```
Select *
From Customer c, Account a
Where c.cid = a.cid
        And bal > 10.000.000
        and ccity='ca mau'
```

Câu hỏi:

1. Viết BT ĐSQH tối ưu tương câu lệnh SQL
2. Vẽ cây bt ĐSQH tương đương.
3. Thay thế các quan hệ toàn cục bằng các đoạn và đơn giản hoá cây bằng cách loại bỏ các phép toán dư thừa