CƠ SỞ DỮ LIỆU

Chương 5,6,7

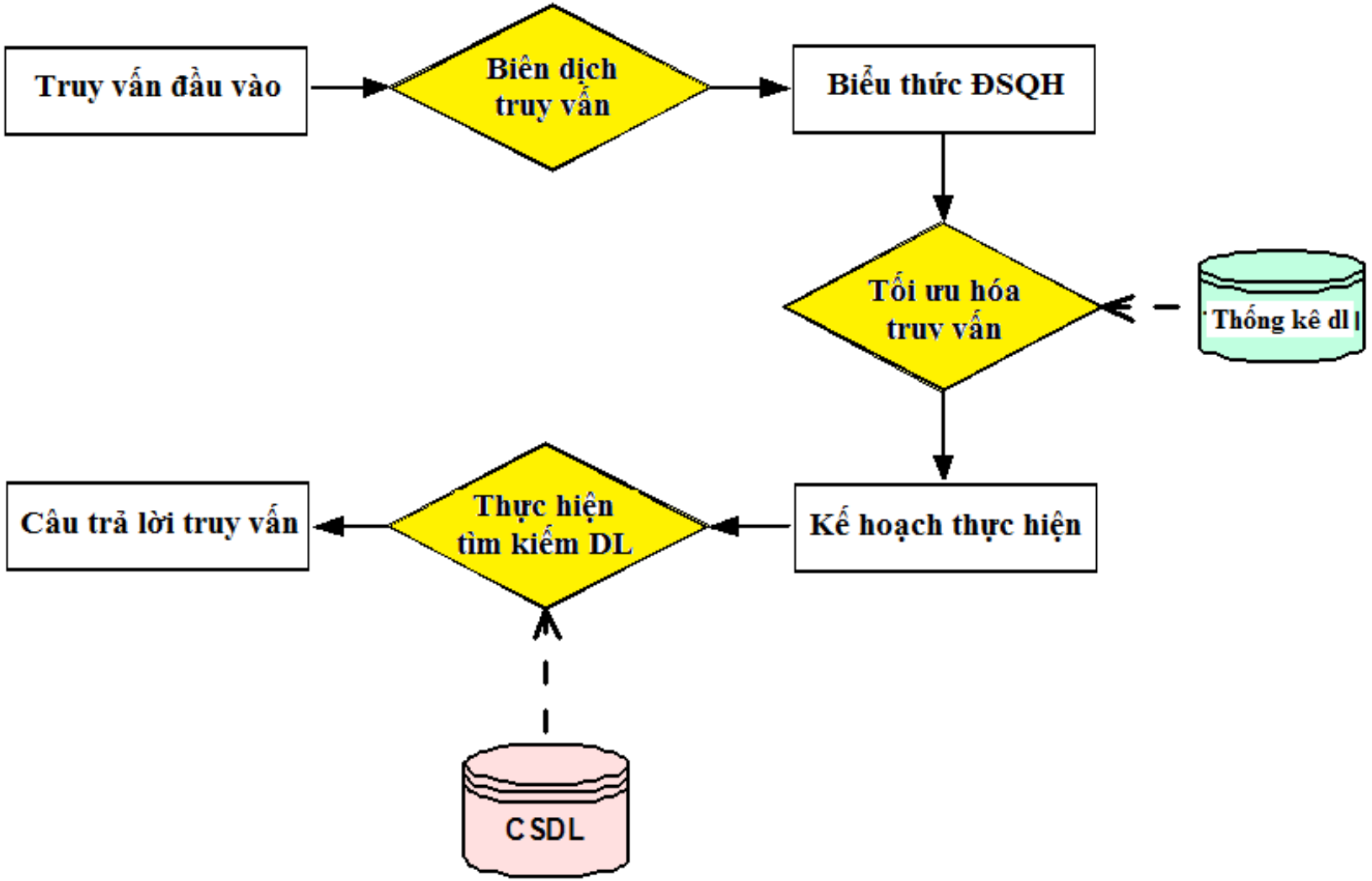
Chương 5 - Tối ưu hóa câu truy vấn

NỘI DUNG:

- Tổng quan về xử lý truy vấn
- Tối ưu hóa các biểu thức đại số quan hệ

5.1. Tổng quan về xử lý truy vấn

- Xử lý một truy vấn bao gồm 3 bước chính:
  - Phân tích và Biên dịch câu truy vấn: dịch câu truy vấn từ dạng ngôn ngữ bậc cao thành một ngôn ngữ biểu diễn dữ liệu bên trong để máy tính có thể thao tác trên đó. Một biểu diễn thích hợp là bằng ngôn ngữ đại số quan hệ
  - Tối ưu hóa câu truy vấn: Mục tiêu của bước tối ưu hóa là chọn ra một kế hoạch thực hiện câu truy vấn có chi phí thấp nhất.
  - Thực hiện đánh giá truy vấn: Từ một kế hoạch thực hiện có được do Trình tối ưu hóa cung cấp, hệ thống sẽ tiến hành thực hiện các thao tác trên dữ liệu trong CSDL và đưa ra câu trả lời cho truy vấn đó.


Truy vấn đầu vào
Biên dịch truy vấn
Biểu thức ĐSQH
Tối ưu hóa truy vấn
Thống kê dl
Kế hoạch thực hiện
Thực hiện tìm kiếm DL
Câu trả lời truy vấn
CSDL

Tối ưu hóa câu truy vấn

Mục tiêu của bước tối ưu hóa là chọn ra một kế hoạch thực hiện câu truy vấn có chi phí thấp nhất.

- Để thực hiện được điều này, trước tiên cần biến đổi một biểu thức ĐSQH đầu vào thành một biểu thức ĐSQH tương đương nhưng có thể xử lý được một cách hiệu quả và ít tốn kém hơn. Bước này gọi là tối ưu hóa đại số.
- Tiếp theo đó, cần phải đặc tả các thuật toán đặc biệt tiến hành thực thi các phép toán , chọn một chỉ dẫn cụ thể nào đó để sử dụng.
- Các dữ liệu thống kê về CSDL sẽ giúp ta trong quá trình xem xét và lựa chọn. Ví dụ như: Số bộ trong quan hệ; Kích thước của một bộ; Số khối (block) chứa các bộ của quan hệ; Số bộ của quan hệ mà một khối có thể chứa; Các thông tin về cơ chế truy nhập, chỉ dẫn trên quan hệ
- Chi phí cho việc thực hiện một truy vấn được đo bởi chi phí sử dụng tài nguyên như: việc truy cập đĩa, thời gian CPU dùng để thực hiện truy vấn.
- Chương này tập trung vào việc đánh giá các biểu thức đại số quan hệ chứ không đi vào chi tiết tính toán chi phí cho việc thực hiện đánh giá truy vấn.

Đánh giá biểu thức ĐSQH

- Sau bước phân tích và biên dịch, ta có một truy vấn được biểu diễn bằng một biểu thức đại số quan hệ bao gồm nhiều phép toán và tác động lên nhiều quan hệ khác nhau, cần tiến hành đánh giá biểu thức này. Có 2 hướng tiếp cận: (i) Vật chất hóa (Materialize), và (ii) Đường ống (Pipeline).
- Vật chất hóa: lần lượt đánh giá các phép toán theo một thứ tự thích hợp. Kết quả của việc đánh giá mỗi phép toán sẽ được lưu trong một quan hệ trung gian tạm thời để sử dụng làm đầu vào cho các phép toán tiếp theo. Điểm bất lợi là cần các quan hệ trung gian (*ghi ra đĩa có chi phí khá lớn*).
- Đường ống: kết hợp một vài phép toán quan hệ vào một đường ống của các phép toán. Trong đường ống thì kết quả của một phép toán được chuyển trực tiếp cho phép toán tiếp theo mà không cần phải lưu lại trong quan hệ trung gian. Cách tiếp cận thứ hai sẽ hạn chế được nhược điểm của cách tiếp cận đầu tiên, nhưng có những trường hợp, ta bắt buộc phải vật chất hóa chứ không dùng đường ống được.

Đánh giá biểu thức ĐSQH (tiếp)

- Ví dụ: Chúng ta có một biểu thức đại số quan hệ gồm 2 phép toán: kết nối và chiếu.

$$\prod_{\{sname,\ pname,\ quantity\}} (S * SP * P)$$

- Trong cách tiếp cận vật chất hóa, xuất phát từ phép toán ở mức thấp nhất là phép kết nối tự nhiên, kết quả của phép kết nối này sẽ được lưu trong một quan hệ trung gian. Sau đó , đọc từ quan hệ trung gian này để tiến hành chiếu lấy kết quả mong muốn.
- Trong cách tiếp cận đường ống, khi một bộ được sinh ra trong phép kết nối 2 quan hệ, bộ này sẽ được chuyển trực tiếp đến phép chiếu để xử lý và kết quả được ghi vào quan hệ đầu ra. Quan hệ kết quả sẽ được tạo lập một cách trực tiếp.

5.2. Tối ưu hóa các biểu thức ĐSQH

- Mục tiêu là tổ chức lại trình tự thực hiện các phép toán trong biểu thức để giảm chi phí thực hiện đánh giá biểu thức đó.
- Trong quá trình tối ưu hóa, ta biểu diễn một biểu thức ĐSQH dưới dạng một cây toán tử. Trong cây thì các nút lá là các quan hệ có mặt trong biểu thức, các nút trong là các phép toán trong biểu thức
- Ví dụ : Đưa ra tên hãng cung ứng mặt hàng có mã là 'P1':

  Select sname From S, SP Where S.sid = SP.sid And pid = 'P1'
- Biểu thức ĐSQH tương ứng là ?
- Cây toán tử tương ứng là ?

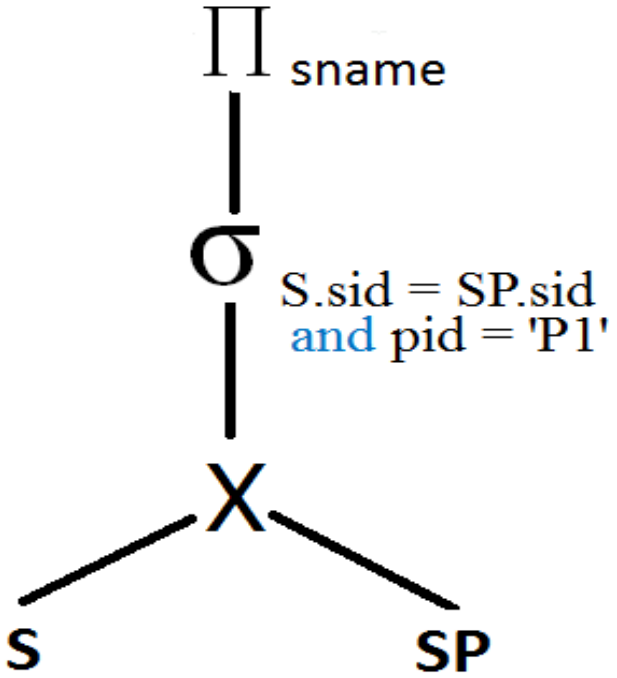
Ví dụ

Cho CSDL gồm các quan hệ:

S (sid, sname, size, city)

P (pid, pname, colour, weight, city)

SP (sid, pid, quantity)



- Đưa ra tên hãng cung ứng mặt hàng có mã là 'P1‘

Select sname From S, SP Where S.sid = SP.sid and pid = 'P1‘

➔ Biểu thức đại số quan hệ và cây toán tử:

$$\prod_{\text{sname}} \left(\sigma_{\substack{\text{S.sid = SP.sid} \\ \text{and pid = 'P1'}}} (\text{S x SP}) \right)$$

Các chiến lược tối ưu tổng quát

1. Đẩy phép chọn và phép chiếu xuống thực hiện sớm nhất có thể: vì hai phép toán này giúp làm giảm kích thước của quan hệ trước khi thực hiện các phép toán 2 ngôi
2. Nhóm dãy các phép chọn và chiếu: Sử dụng chiến lược này nếu như có một dãy các phép chọn hoặc dãy các phép chiếu trên cùng một quan hệ
3. Kết hợp phép chọn và tích Đề các thành phép kết nối: Nếu kết quả của một phép tích Đề các là đối số của 1 phép chọn có điều kiện chọn là phép so sánh giữa các thuộc tính trên 2 quan hệ tham gia tích Đề các thì ta nên kết hợp 2 phép toán thành phép kết nối.
4. Tìm các biểu thức con chung trong biểu thức đại số quan hệ để đánh giá chỉ một lần
5. Xác định các phép toán có thể được đưa vào đường ống và thực hiện đánh giá chúng theo đường ống
6. Xử lý các tệp dữ liệu trước khi tiến hành tính toán: Tạo lập chỉ dẫn hay sắp xếp tệp dữ liệu có thể góp phần làm giảm chi phí của các phép tính trung gian
7. Ước lượng chi phí và lựa chọn thứ tự thực hiện: Do với mỗi câu truy vấn có thể có nhiều cách khác nhau để thực hiện, với việc ước lượng chi phí (số phép tính, tài nguyên sử dụng, dung tích bộ nhớ, thời gian thực hiện ..) ta có thể chọn cách đánh giá biểu thức ĐSQH có chi phí nhỏ nhất.

Các phép biến đổi tương đương biểu thức ĐSQH

- Hai biểu thức ĐSQH E_1 và E_2 là tương đương nếu chúng cho cùng một kết quả khi áp dụng trên cùng một tập các quan hệ
- Trong phần này, ta có các ký hiệu dạng sau: E_1, E_2, E_3, … là các biểu thức đại số quan hệ; F_1, F_2, F_3, … là các điều kiện chọn hoặc là các điều kiện kết nối; X_1, X_2, … Y, Z, U_1, U_2, … là các tập thuộc tính

1. Quy tắc kết hợp của phép tích Đề các và kết nối

$$(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$$

$$(E_1 * E_2) * E_3 \equiv E_1 * (E_2 * E_3)$$

$$(E_1 \underset{F1}{\triangleright\!\triangleleft} E_2) \underset{F2}{\triangleright\!\triangleleft} E_3 \equiv E_1 \underset{F1}{\triangleright\!\triangleleft} (E_2 \underset{F2}{\triangleright\!\triangleleft} E_3)$$

- Qui tắc này sử dụng cho chiến lược số 7. Thứ tự thực hiện các phép kết nối hay tích Đề các là rất quan trọng vì kích thước của quan hệ trung gian có thể rất lớn. Lựa chọn thứ tự tùy thuộc vào kích thước của các quan hệ tham gia phép toán và cả ngữ nghĩa của quan hệ (mối liên hệ)

- Ví dụ: S* SP * P có thể được thực hiện theo 3 thứ tự như sau
  1) (S*SP)*P
  2) (S*P)*SP
  3) S*(SP*P)

  Xét theo ngữ nghĩa S, P không kết nối được nên (1) và (3) là tốt hơn (2). Xét về kích thước thì (3) tốt hơn (1) vì S có 4 thuộc tính còn P có 3 thuộc tính, tuy nhiên, cũng còn tùy thuộc vào lực lượng của 2 quan hệ S và P nữa

Các phép biến đổi tương đương biểu thức ĐSQH

2. Quy tắc giao hoán trong phép tích Đề các và kết nối

$$E_1 \times E_2 \equiv E_2 \times E_1$$

$$E_1 * E_2 \equiv E_2 * E_1$$

$$E_1 \rhd\lhd_F E_2 \equiv E_2 \rhd\lhd_F E_1$$

3. Quy tắc đối với dãy các phép chiếu

$$\Pi_{X_1}(\Pi_{X_2} \ldots \Pi_{X_n}(E)\ldots) \equiv \Pi_{X_1}(E)$$

$$X_1 \subseteq X_2 \subseteq \ldots \subseteq X_n$$

4. Quy tắc đối với dãy các phép chọn

$$\sigma_{F1}(\sigma_{F2} \ldots . \sigma_{Fn}(E)\ldots) \equiv \sigma_{F1 \wedge F2 \wedge \ldots \wedge Fn}(E)$$

5. Quy tắc giao hoán phép chọn và phép chiếu

$$\prod_X(\sigma_F(E)) \equiv \sigma_F(\prod_X(E))$$

Quy tắc này áp dụng khi F là điều kiện xác định được trên tập thuộc tính X. Tổng quát hơn ta có:

$$\prod_X(\sigma_F(E)) \equiv \prod_X(\sigma_F(\prod_{XY}(E)))$$

6. Quy tắc đối với phép chọn và phép tích Đề các: Ký hiệu: $E_1(U_1)$ có nghĩa là biểu thức E_1 xác định trên tập thuộc tính U_1; $F_1(U_1)$ có nghĩa là điều kiện chọn F_1 xác định trên tập thuộc tính U_1. Quy tắc biến đổi liên quan đến phép chọn và tích Đề các được phát biểu như sau:

$\sigma_F(E_1(U_1) \times E_2(U_2))$ tương đương với:

- $\sigma_{F1}(E_1) \times E_2$ trong trường hợp $F = F_1(U_1)$
- $\sigma_{F1}(E_1) \times \sigma_{F2}(E_2)$ trong trường hợp $F = F_1(U_1) \wedge F_2(U_2)$
- $\sigma_{F2}(\sigma_{F1}(E_1) \times E_2)$ trong trường hợp $F = F_1(U_1) \wedge F_2(U_1U_2)$

7. Quy tắc đối với phép chọn và phép hợp:

$$\sigma_F(E_1 \cup E_2) \equiv \sigma_F(E_1) \cup \sigma_F(E_2)$$

8. Quy tắc đối với phép chọn và phép trừ:

$$\sigma_F(E_1 - E_2) \equiv \sigma_F(E_1) - \sigma_F(E_2)$$

9. Quy tắc đối với phép chiếu và tích Đề các:

$$\prod_X(E_1(U_1) \times E_2(U_2)) \equiv \prod_Y(E_1) \times \prod_Z(E_2)$$

$$X = YZ, Y \subset U_1, Z \subset U_2$$

10. Quy tắc đối với phép chiếu và phép hợp:

$$\prod_X(E_1 \cup E_2) \equiv \prod_X(E_1) \cup \prod_X(E_2)$$

Ví dụ

Cho CSDL gồm các quan hệ:

S (sid, sname, size, city)

P (pid, pname, colour, weight, city)

SP (sid, pid, quantity)

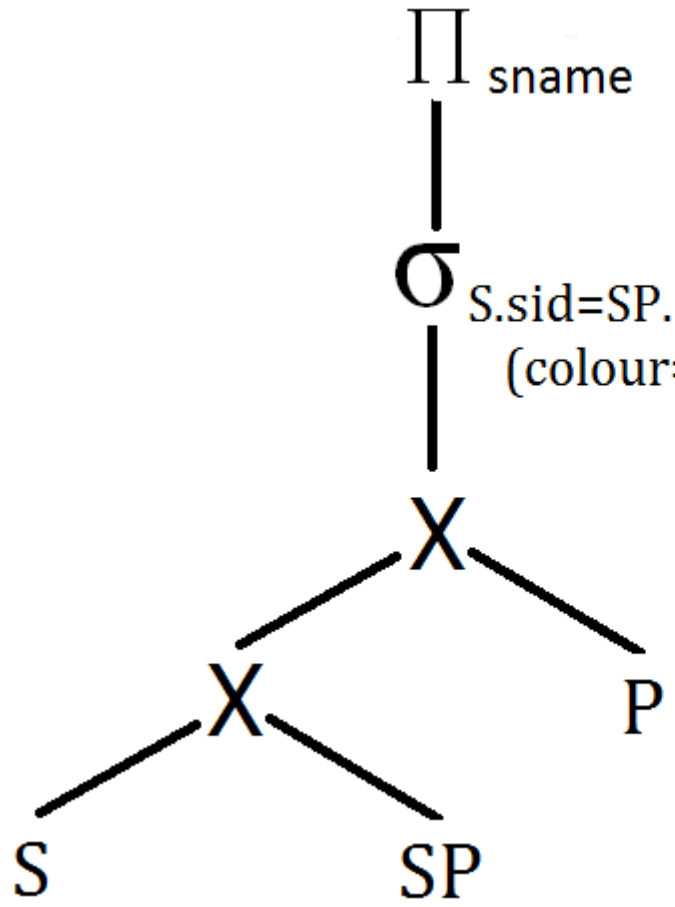
- Tìm tên hãng cung ứng ít nhất một mặt hàng màu đỏ hoặc màu xanh

  SELECT sname FROM S, P, SP

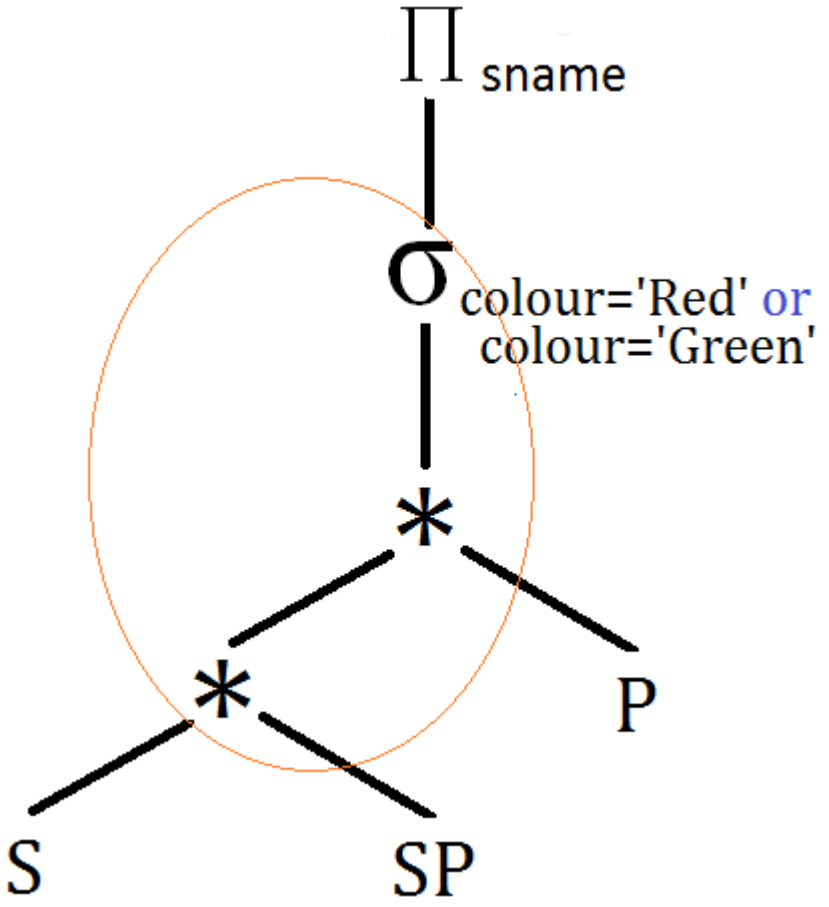
  WHERE S.sid = SP.sid AND P.pid = SP.pid AND (colour = 'Red' OR colour = 'Green');

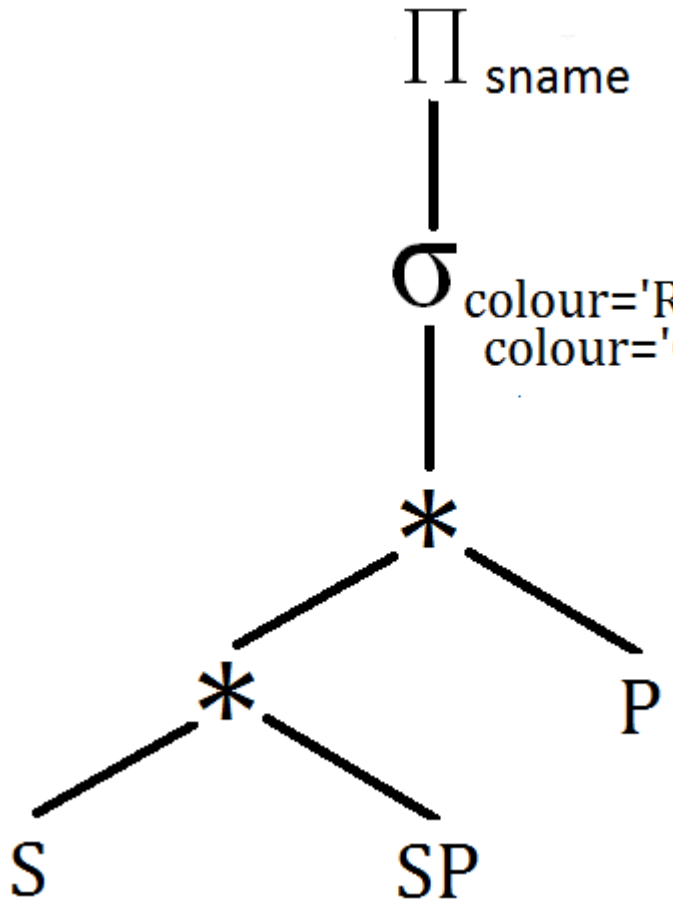
- Biểu thức đại số quan hệ tương đương với câu truy vấn trên là:

$$\Pi_{sname}(\sigma_{S.sid=SP.sid \wedge P.pid=SP.pid \wedge (colour='Red' \vee colour='Green')}(S \times SP \times P))$$

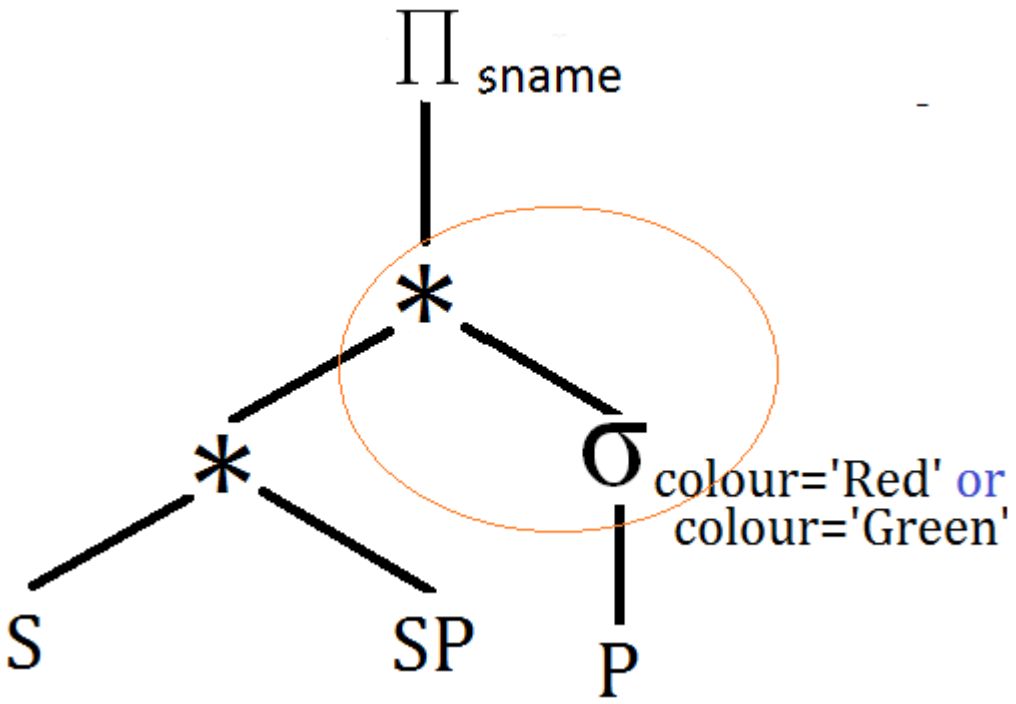


$\Pi_{\text{sname}} \sigma_{\text{S.sid=SP.sid and SP.pid=P.pid and (colour='Red' or colour='Green')}} (S \times SP \times P)$

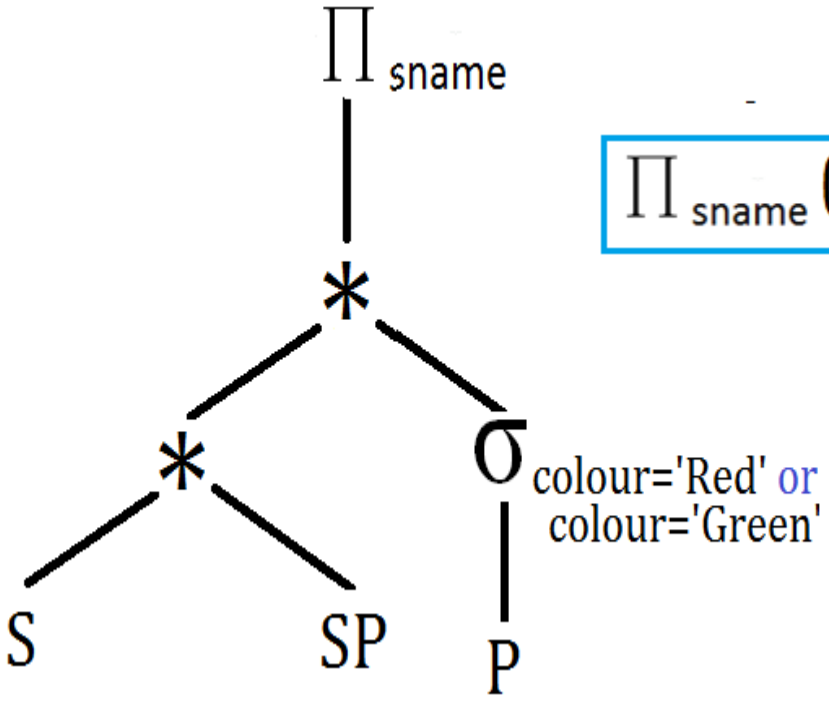




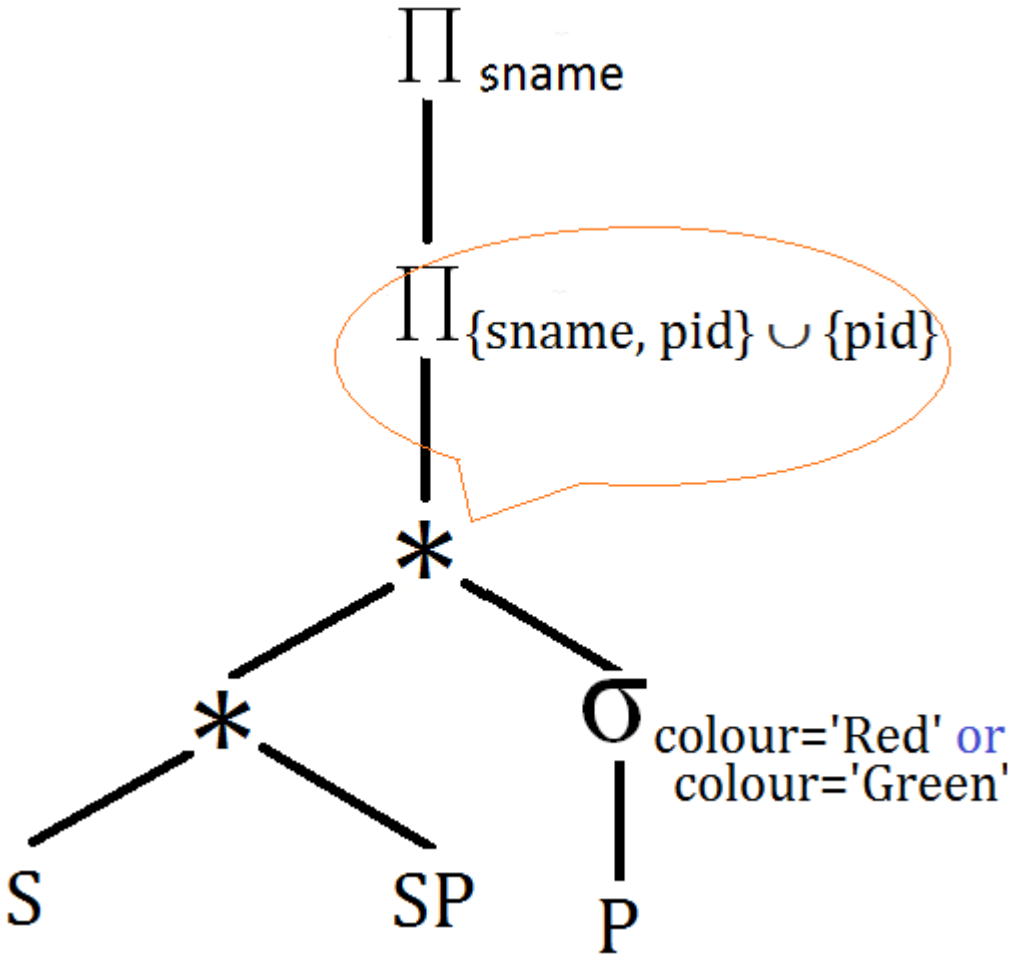
$$\prod_{sname} \sigma_{colour='Red' \text{ or } colour='Green'} (S * SP * P)$$



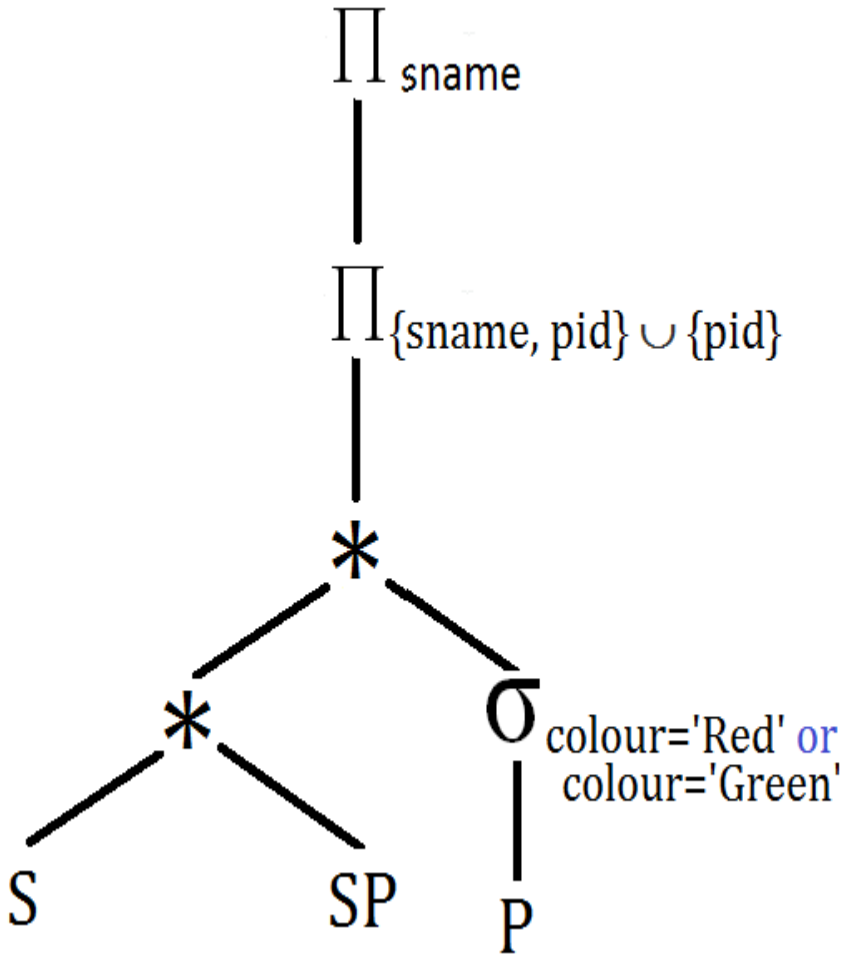
$$\prod_{sname} (S * SP * \sigma_{colour='Red' \text{ or } colour='Green'} (P))$$

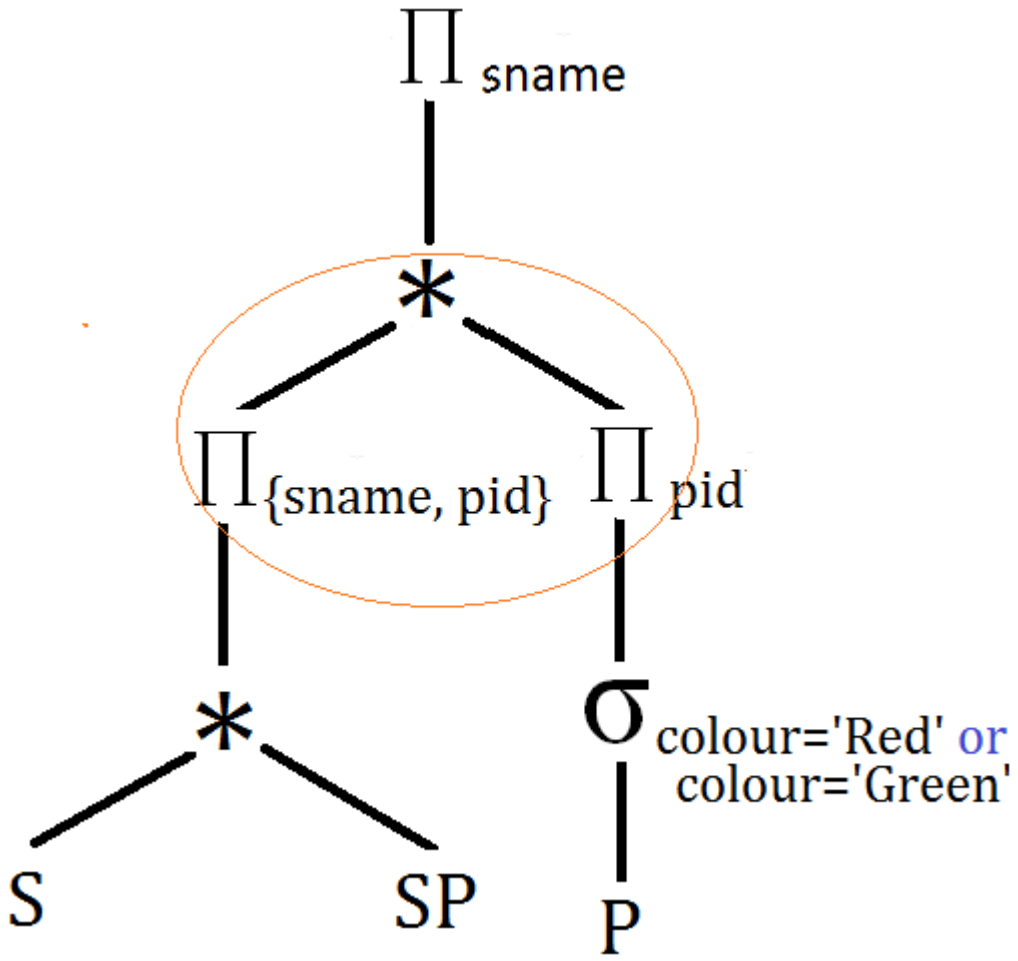


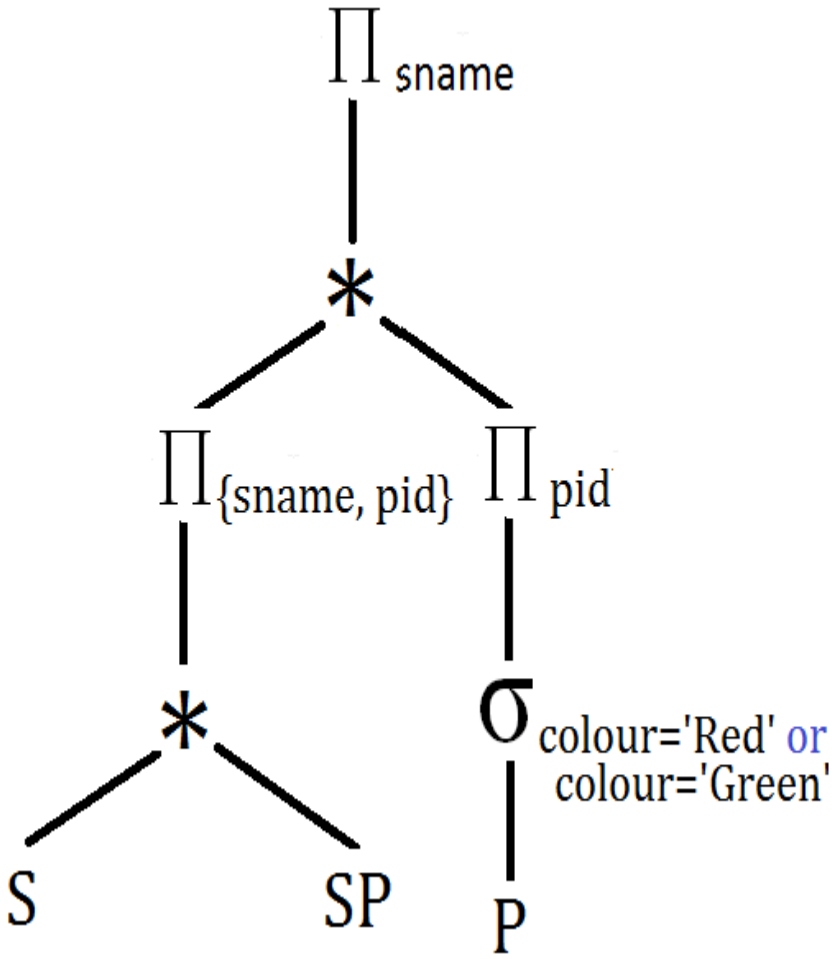
$$\prod_{sname}(S * SP * \sigma_{colour='Red' \text{ or } colour='Green'}(P))$$

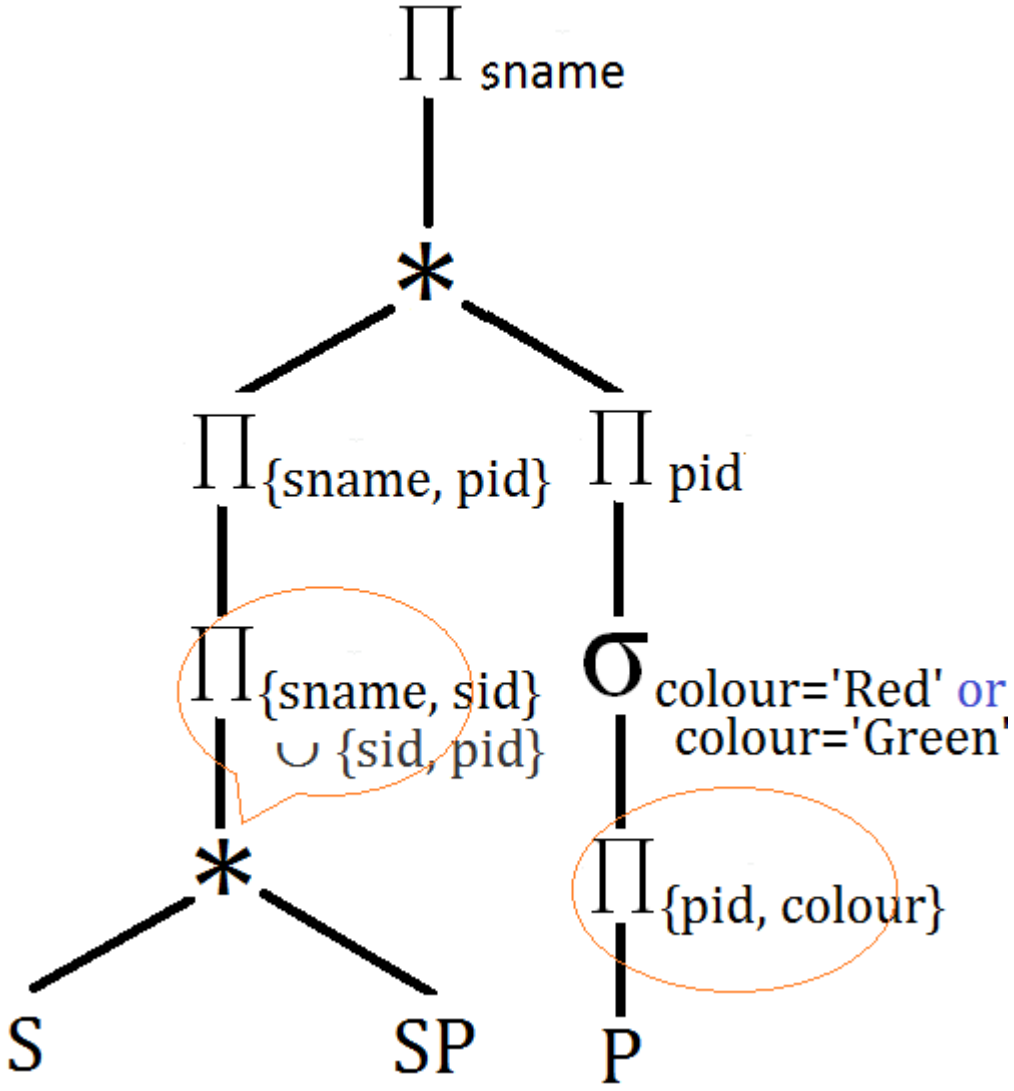


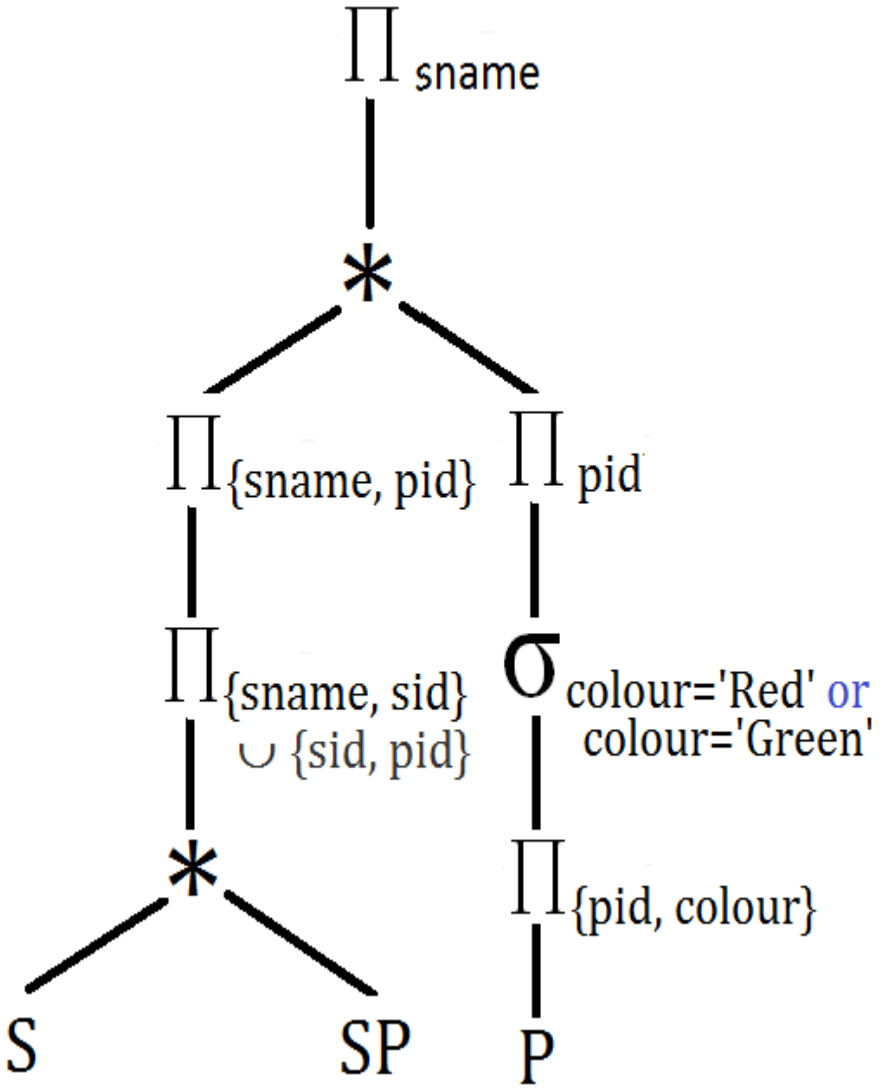
$$\prod_{\text{sname}} \prod_{\{\text{sname, pid}\}} (S * SP * \sigma_{\text{colour='Red' or colour='Green'}} (P))$$

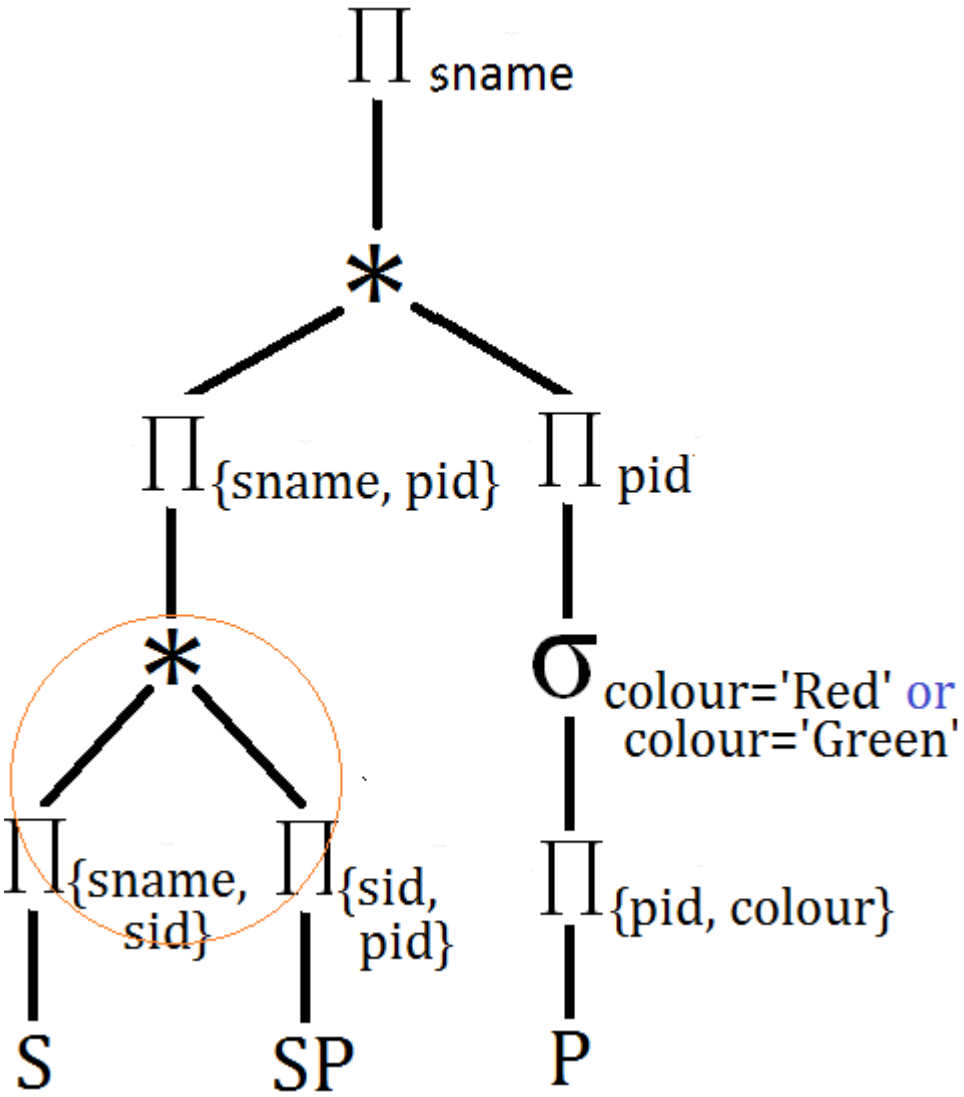





∏ sname
*
∏ {sname, pid}
∏ pid
*
σ colour='Red' or colour='Green'
S
SP
P


∏ sname
*
∏ {sname, pid}
∏ pid
∏ {sname, sid} ∪ {sid, pid}
σ colour='Red' or colour='Green'
*
∏ {pid, colour}
S
SP
P


∏ sname
*
∏ {sname, pid}
∏ {sname, sid} ∪ {sid, pid}
*
S
SP
∏ pid
σ colour='Red' or colour='Green'
∏ {pid, colour}
P

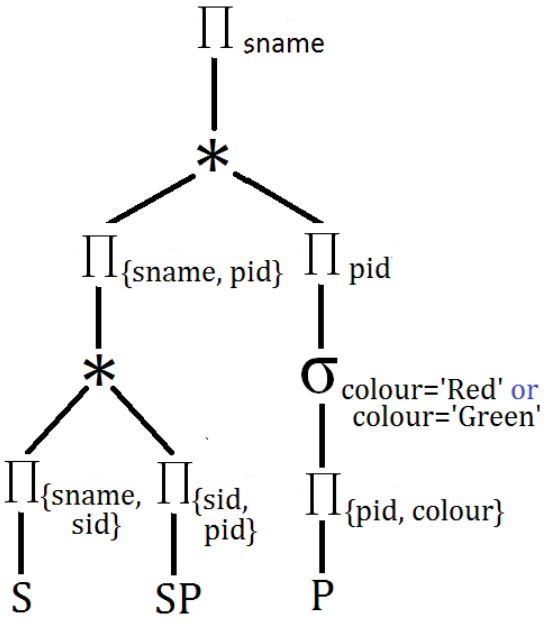

∏ sname
*
∏ {sname, pid}
*
∏ {sname, sid}
∏ {sid, pid}
S
SP
∏ pid
σ colour='Red' or colour='Green'
∏ {pid, colour}
P

Cho CSDL gồm các quan hệ:

S(sid, sname, size, city), P(pid, pname, colour, weight, city), SP (sid, pid, quantity)

Tìm tên hãng cung ứng ít nhất một mặt hàng màu đỏ hoặc màu xanh

$$\prod_{sname}\left(\prod_{\{sname,\ pid\}}\left(\prod_{\{sname,\ sid\}}(S)*\prod_{\{sid,\ pid\}}(SP)\right)*\right.$$
$$\left.*\prod_{pid}\left(\sigma_{colour='Red'\ or\ colour='Green'}\left(\prod_{\{pid,\ colour\}}(P)\right)\right)\right)$$




```
SELECT sname FROM
  (SELECT sname, pid FROM
     (SELECT sname, sid FROM S) AS S1,
     (SELECT sid, pid FROM SP) AS SP1
   WHERE S1.sid = SP1.sid    )   AS   SSP1,
  (SELECT pid FROM
     (SELECT pid, colour FROM P) AS  P1
   WHERE P1.colour = 'Red' or P1.colour = 'Green') AS P2
WHERE  SSP1.pid = P2.pid
```

```
SELECT sname FROM S WHERE sid IN
   (SELECT sid FROM SP WHERE pid IN
      (SELECT pid FROM P WHERE
         colour = ‘Red’ OR colour = ‘Green’ ) )
```

Kết luận

- Xử lý truy vấn:
  - Biên dịch câu truy vấn
  - Tối ưu hóa câu truy vấn
  - Thực hiện tìm kiếm dữ liệu
- Tối ưu hóa câu truy vấn:
  - Biến đổi tương đương
  - Đặc tả các phép toán của biểu thức
  - Đánh giá chi phí, chọn kế hoạch thực hiện

Chương 6 – An toàn và toàn vẹn dữ liệu

NỘI DUNG:

- Đặt vấn đề
- An toàn dữ liệu
- Toàn vẹn dữ liệu
- Giao dịch, điều khiển đồng thời

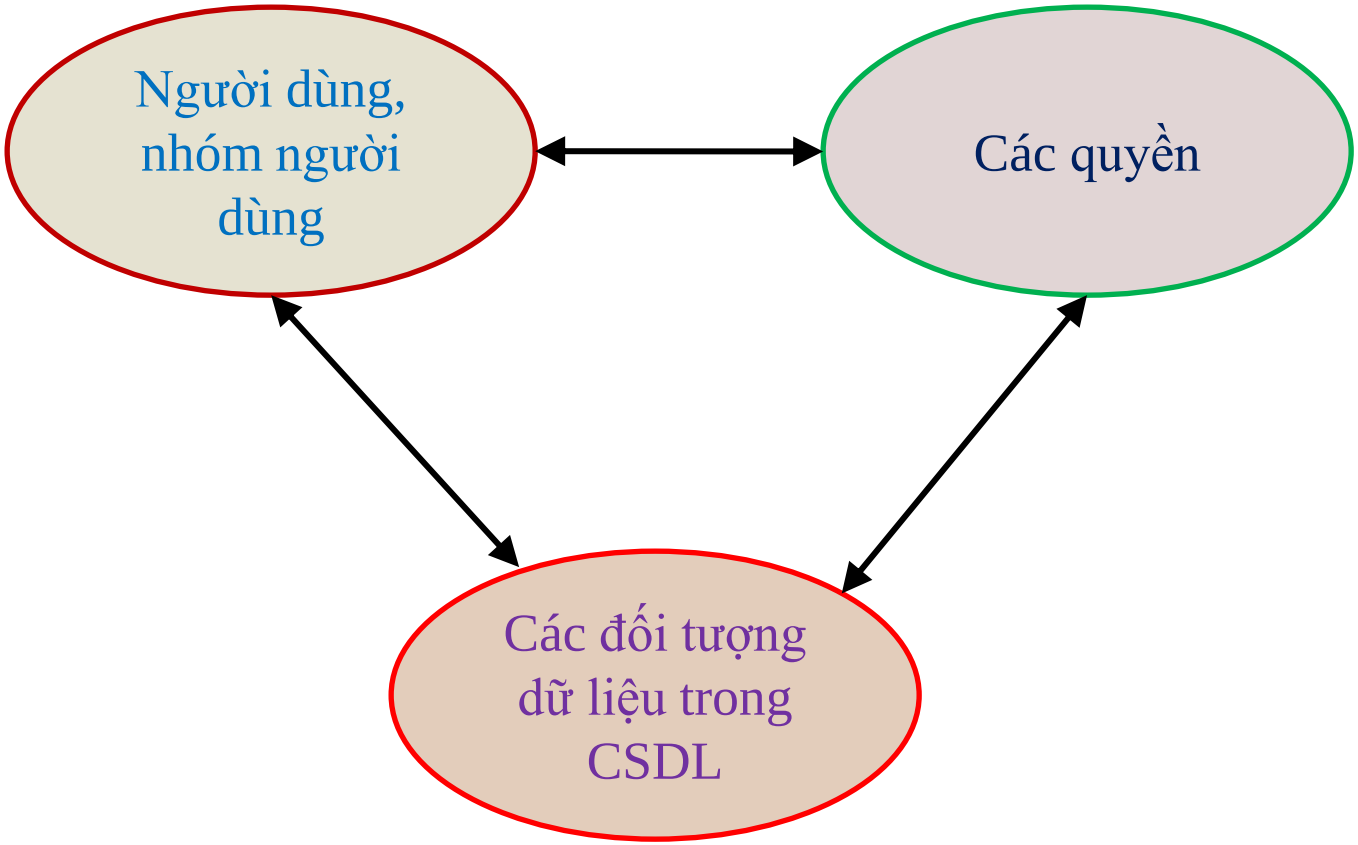
6.1. Đặt vấn đề

- Một số yêu cầu đối với thiết kế, cài đặt và quản trị CSDL:
  - Đảm bảo tính an toàn của dữ liệu
    - Tránh truy nhập không hợp lệ từ phía người dùng: phân quyền, xác minh và kiểm tra quyền hạn người sử dụng.
  - Đảm bảo tính đúng đắn của dữ liệu
    - Tránh sai sót khi cập nhật dữ liệu: định nghĩa và kiểm tra các ràng buộc dữ liệu.
    - Tránh sai sót trong quá trình thao tác với dữ liệu: kiểm tra tính toàn vẹn của các thao tác với dữ liệu.

6.2. An toàn dữ liệu

- **Định nghĩa:** Tính an toàn dữ liệu là sự bảo vệ dữ liệu trong CSDL chống lại những truy nhập, sửa đổi hay phá hủy bất hợp pháp.
- Người sử dụng hợp pháp là những người sử dụng được cấp phép, ủy quyền.
- Để đảm bảo tính an toàn cho CSDL, cần có một cơ chế để quản lý người dùng hợp lý.
- Những nhóm người dùng khác nhau trong hệ CSDL có quyền sử dụng khác nhau đối với các đối tượng dữ liệu trong CSDL.

Trục tam giác



6.2.1. Các quyền truy nhập của người sử dụng

- Đối với người khai thác CSDL:
  - Quyền đọc dữ liệu: được phép đọc một phần hay toàn bộ dữ liệu trong CSDL.
  - Quyền cập nhật dữ liệu: được phép sửa đổi một số giá trị nhưng không được xóa dữ liệu trong CSDL.
  - Quyền xóa dữ liệu: được phép xóa dữ liệu trong CSDL.
  - Quyền bổ sung dữ liệu: được phép thêm dữ liệu mới vào trong CSDL nhưng không được phép thay đổi dữ liệu
- Đối với người quản trị CSDL:
  - Quyền tạo chỉ dẫn trên các quan hệ trong CSDL
  - Quyền thay đổi sơ đồ cơ sở dữ liệu: thêm hay xóa các thuộc tính của các quan hệ trong CSDL
  - Quyền loại bỏ quan hệ trong CSDL
  - Quyền quản lý tài nguyên: được phép thêm các quan hệ mới vào CSDL

6.2.2. Người dùng

- Một người dùng cụ thể (user)
- Một nhóm người dùng (group)
- Một số hệ quản trị không phân biệt hai khái niệm trên, mà gọi chung là vai trò (role)

6.2.3. Các đối tượng dữ liệu

- Tables
- Views

Trách nhiệm của người quản trị hệ thống

- Để có thể phân biệt được người sử dụng trong hệ CSDL, người quản trị hệ thống phải có trách nhiệm:
  - Xác định các quyền cụ thể mà mỗi người sử dụng hay một nhóm người sử dụng được phép thực hiện, xác định vai trò và trách nhiệm của mỗi người sử dụng. Điều này được gọi chung là Phân quyền người sử dụng.
  - Cung cấp một phương tiện cho người sử dụng để hệ thống có thể nhận biết được người sử dụng đó hay còn gọi là Xác minh người sử dụng.

Xác minh người sử dụng

- Để xác minh được người sử dụng, người ta có thể dùng các kỹ thuật sau:
  - Kỹ thuật dùng tài khoản có tên và mật khẩu, mật khẩu cũng được bảo vệ bởi hệ thống.
  - Kỹ thuật sử dụng các hàm kiểm tra người sử dụng: Hệ thống đưa cho người sử dụng một số ngẫu nhiên x, người sử dụng dùng một hàm F tính nhẩm kết quả và đưa kết quả y = F(x) vào hệ thống. Trong lúc đó, hệ thống cũng tính toán và so sánh kết quả với y. Người sử dụng hợp pháp là người biết hàm biến đổi F và đưa vào giá trị y đúng.
  - Kỹ thuật dùng thẻ điện tử, thẻ thông minh.
  - Kỹ thuật sử dụng nhận dạng tiếng nói, vân tay v..v.

Kiểm tra quyền truy nhập của người sử dụng

- Mỗi người sử dụng sẽ có một bộ hồ sơ do người quản trị thiết lập và được hệ thống quản lý, trong hồ sơ đó sẽ có chi tiết về các thao tác người sử dụng được phép thực hiện:
  - Phân quyền người sử dụng: Người quản trị hệ thống phải có trách nhiệm xác định khung nhìn để kiểm soát xem mỗi người sử dụng chỉ được truy nhập phần dữ liệu nào trong CSDL và có được các quyền nào trong số các quyền đọc, thêm, xóa, sửa đổi.
  - Xác định và kiểm soát sự lưu chuyển dữ liệu: Hệ thống phải bảo trì danh sách các quyền một cách chặt chẽ vì người sử dụng có thể được quyền lan truyền các quyền cho người sử dụng khác.

Các câu lệnh an toàn dữ liệu trong SQL

- Câu lệnh tạo khung nhìn
- Câu lệnh phân quyền cho người sử dụng
- Câu lệnh thu hồi quyền của người sử dụng

Câu lệnh tạo khung nhìn

- CREATE VIEW <Tên khung nhìn> [(d/s cột)] AS <Câu truy vấn>
- Danh sách các cột trong khung nhìn là phần không bắt buộc. Trong trường hợp người sử dụng muốn đặt tên khác cho các cột xuất hiện trong khung nhìn thì người sử dụng có thể chỉ ra tên các cột, dữ liệu trên cột thì tương ứng với các cột trong mệnh đề SELECT của câu truy vấn.

VÍ DỤ

- Cho cơ sở dữ liệu gồm 2 quan hệ:

Nhânviên(Id, Họtên, Địachỉ, Lương, NămBD, Đánhgiá, PhòngCT)

Phòng(PId, Tên, ĐC, Điệnthoại, Trưởngphòng)

- Câu lệnh tạo khung nhìn cho một nhân viên của phòng 'Khoa học' có thể được định nghĩa như sau:

```
CREATE VIEW NVKH(HọtênNhânviên, Địachỉliênlạc) AS
SELECT Họtên, Địachỉ  FROM Nhânviên WHERE PhòngCT IN
  (SELECT PId FROM Phòng WHERE Tên ='Khoa học')
```

Câu lệnh phân quyền cho NSD

- GRANT <D/s thao tác> ON <Đối tượng>
  TO <D/s người dùng> [WITH GRANT OPTION]
- <D/s thao tác>: có thể bao gồm một hay nhiều thao tác được liệt kê dưới đây:
  - Insert: chèn dữ liệu vào trong CSDL có sẵn nhưng không được thay đổi bất kỳ mục dữ liệu nào trong CSDL
  - Update: sửa đổi dữ liệu nhưng không được xóa dữ liệu
  - Delete: xóa dữ liệu trong CSDL
  - Select : tìm kiếm
  - Create: tạo lập các quan hệ mới
  - Alter: Thay đổi cấu trúc của quan hệ
  - Drop: Loại bỏ quan hệ
  - Read/Write: Đọc và Ghi
- <Đối tượng>: bảng hoặc khung nhìn

- <D/s người dùng>: Một người hay một nhóm hay một danh sách người sử dụng. Từ khóa public được dùng thay thế cho mọi người sử dụng
- [With Grant Option] Nếu dùng từ khóa này trong câu lệnh phân quyền thì người dùng xuất hiện trong <D/s người dùng> có quyền được lan truyền các quyền vừa được tuyên bố cho những người dùng khác

VÍ DỤ

- Trao quyền đọc, ghi, tìm kiếm, sửa đổi dữ liệu cho nhân viên tên 'Hoa' của phòng 'Khoa học' trên khung nhìn vừa tạo lập trong phần trước

```
GRANT read, write, select, update ON NVKH TO Hoa;
```

- Trao quyền cho trưởng phòng Khoa học – ông HungNC

```
GRANT read, write, select, update, delete ON NVKH TO HungNC
WITH GRANT OPTION;
```

Câu lệnh thu hồi quyền của NSD

- REVOKE <D/s thao tác> ON <Đối tượng> FROM <D/s người dùng> [RESTRICT/CASCADE]
- <D/s thao tác>, <Đối tượng>, <D/s người dùng> giống như đối với câu lệnh GRANT.
- Phần [RESTRICT/CASCADE] là chỉ ra cơ chế thu hồi với các quyền đã được người dùng trong <D/s người dùng> lan truyền
- Nếu Restrict thì có nghĩa là chỉ hủy bỏ quyền của những người có trong danh sách, quyền đã được lan truyền cho người khác không bị thu hồi.
- Nếu dùng Cascade thì hủy bỏ quyền của người trong <D/s người dùng>, đồng thời kéo theo hủy bỏ quyền mà người dùng đó đã luân chuyển cho những người khác.
- Ví dụ:

REVOKE update, delete ON NVKH FROM HungNC CASCADE

6.3. Toàn vẹn dữ liệu

- Định nghĩa: Tính toàn vẹn dữ liệu là sự bảo vệ dữ liệu trong CSDL chống lại những sự sửa đổi, phá hủy vô căn cứ để đảm bảo tính đúng đắn và chính xác của dữ liệu.
- Các thao tác có thể ảnh hưởng đến tính đúng đắn của CSDL là thêm, xóa, sửa đổi.
- Để đảm bảo tính toàn vẹn dữ liệu, cần phải chỉ ra và duy trì những ràng buộc toàn vẹn liên kết với mỗi quan hệ. Các ràng buộc toàn vẹn cung cấp một phương tiện để đảm bảo rằng các thao tác được thực hiện bởi những người sử dụng hợp pháp không làm mất đi tính đúng đắn của CSDL.
- Trong hệ thống đa người dùng, để đảm bảo được toàn vẹn dữ liệu, hệ thống còn phải có được một trình điều khiển tương tranh để tránh đụng độ giữa các thao tác được đưa ra bởi những người sử dụng khác nhau tại cùng một thời điểm

Các ràng buộc toàn vẹn trong SQL

- Các ràng buộc về khóa chính, khóa ngoài, kiểm tra miền giá trị sử dụng Check đã được đề cập đến khi nói về câu lệnh tạo bảng trong CSDL.
- Các khẳng định (assertion): Là một vị từ biểu thị một điều kiện mà CSDL phải luôn luôn thỏa mãn.

  Các khẳng định được tạo ra bằng câu lệnh:

  CREATE ASSERTION <Tên khẳng định> CHECK <Vị từ>
- Các kích hoạt (trigger): Là một thủ tục lưu trữ hệ thống (stored procedure) đặc biệt, được thực thi một cách tự động khi có sự kiện gây biến đổi dữ liệu như Update, Insert hay Delete

  Được dùng để đảm bảo toàn vẹn dữ liệu hay thực hiện các quy tắc nghiệp vụ nào đó.

Ví dụ về khẳng định

- Số lượng mặt hàng được cung cấp bởi các hãng có số nhân viên < 50 phải nhỏ hơn 100:

```
CREATE ASSERTION KĐSốlượng CHECK NOT EXISTS
(SELECT * FROM S WHERE numofemps < 50 AND sid
IN (SELECT sid FROM SP WHERE quantity >= 100))
```

- Lương của nhân viên không được cao hơn lương người quản lý phòng ban của nhân viên đó.

```
CREATE ASSERTION Salary_Constraint
CHECK (NOT EXISTS
        (SELECT * FROM Employee E,
        Manager M, Department D
        WHERE E.Salary>M.Salary AND
E.Dno=D.Number AND D.MgrSSN=M.SSN))
```

Sử dụng trigger

- Khi nào sử dụng trigger?
  - khi các biện pháp đảm bảo toàn vẹn dữ liệu khác như Constraint không thể thỏa mãn yêu cầu của ứng dụng
    - Constraint thuộc loại toàn vẹn dữ liệu khai báo: kiểm tra dữ liệu trước khi cho phép nhận vào bảng
    - Trigger thuộc loại toàn vẹn dữ liệu thủ tục nên việc Insert, Update, Delete xảy ra rồi mới kích hoạt trigger.
  - Đôi khi, do nhu cầu thay đổi dây chuyền, có thể sử dụng trigger

Đặc điểm của trigger

- một trigger có thể làm nhiều công việc, có thể được kích hoạt bởi nhiều sự kiện
- trigger không thể được tạo ra trên bảng tạm hoặc bảng hệ thống
- trigger chỉ có thể được kích hoạt tự động bởi các sự kiện mà không thể chạy thủ công được.
- có thể áp dụng trigger cho view
- khi trigger được kích hoạt
  - dữ liệu mới được insert sẽ được chứa trong bảng "inserted"
  - dữ liệu mới được delete sẽ được chứa trong bảng "deleted"
  - đây là hai bảng tạm nằm trên bộ nhớ, và chỉ có giá trị bên trong trigger

Trigger

- ECA: Sự kiện – Điều kiện – Hành động
- Sự kiện (event): có sự cập nhật dữ liệu (insert, update, delete)
- Điều kiện (condition): biểu thức SQL có giá trị Boolean
- Hành động (Action): câu lệnh SQL

Ví dụ về trigger

- Nhânviên(ID, Họtên, Lương, Địachỉ, Ngườiquảnlý)
- Một nhân viên bao giờ cũng có lương ít hơn lương người trưởng phòng, điều kiện này phải được kiểm tra khi thêm bộ dữ liệu.

```
CREATE TRIGGER ThemNV INSERT ON Nhânviên
IF Nhânviên.Lương > (SELECT E.Lương FROM
   Nhânviên AS E WHERE E.ID =
   Nhânviên.Ngườiquảnlý)
THEN ABORT;
```

Ví dụ

- **Constraint:** Khi thêm một sinh viên, thì số sinh viên trong lớp sẽ tăng lên

student(**student_id**, first_name,last_name, dob, gender, address, note, email, *clazz_id*)

clazz(**clazz_id**, name, *lecturer_id*, *monitor_id*, number_students)

```
CREATE TRIGGER clazz_changes_tg
AFTER INSERT ON student
REFERENCING NEW ROW AS nnn
FOR EACH ROW
WHEN (nnn.clazz_id IS NOT NULL)
BEGIN
        update clazz
        set number_students = number_students + 1
        where clazz_id = nnn.clazz_id;
END;
```

Event

Condition

Action

Cú pháp

- Tạo một Trigger

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
      {BEFORE | AFTER | INSTEAD OF }
      {INSERT | DELETE | UPDATE [OF <attribute_name>]}
      ON <table_name>
      REFERENCING {NEW | OLD} {ROW | TABLE} AS <name>
      [FOR EACH ROW ]
      [WHEN (<condition>) ]
      BEGIN
            <trigger body goes here >
      END;
```

- Xóa Trigger

```
DROP TRIGGER <trigger_name>;
```

- AFTER, BEFORE: used for tables / views
- INSTEAD OF: used only for views
- UPDATE OF <colums>: update on a particular column
- Row-level trigger: Indicated by option FOR EACH ROW

6.4. Giao dịch và điều khiển đồng thời

- Trong hệ CSDL đa người dùng, hệ thống cần đưa ra giải pháp chống đụng độ giữa các giao dịch (một dãy các thao tác) được đưa ra bởi những người dùng khác nhau để tránh việc một đối tượng dữ liệu nào đó bị làm mất tính đúng đắn trong quá trình cập nhật.

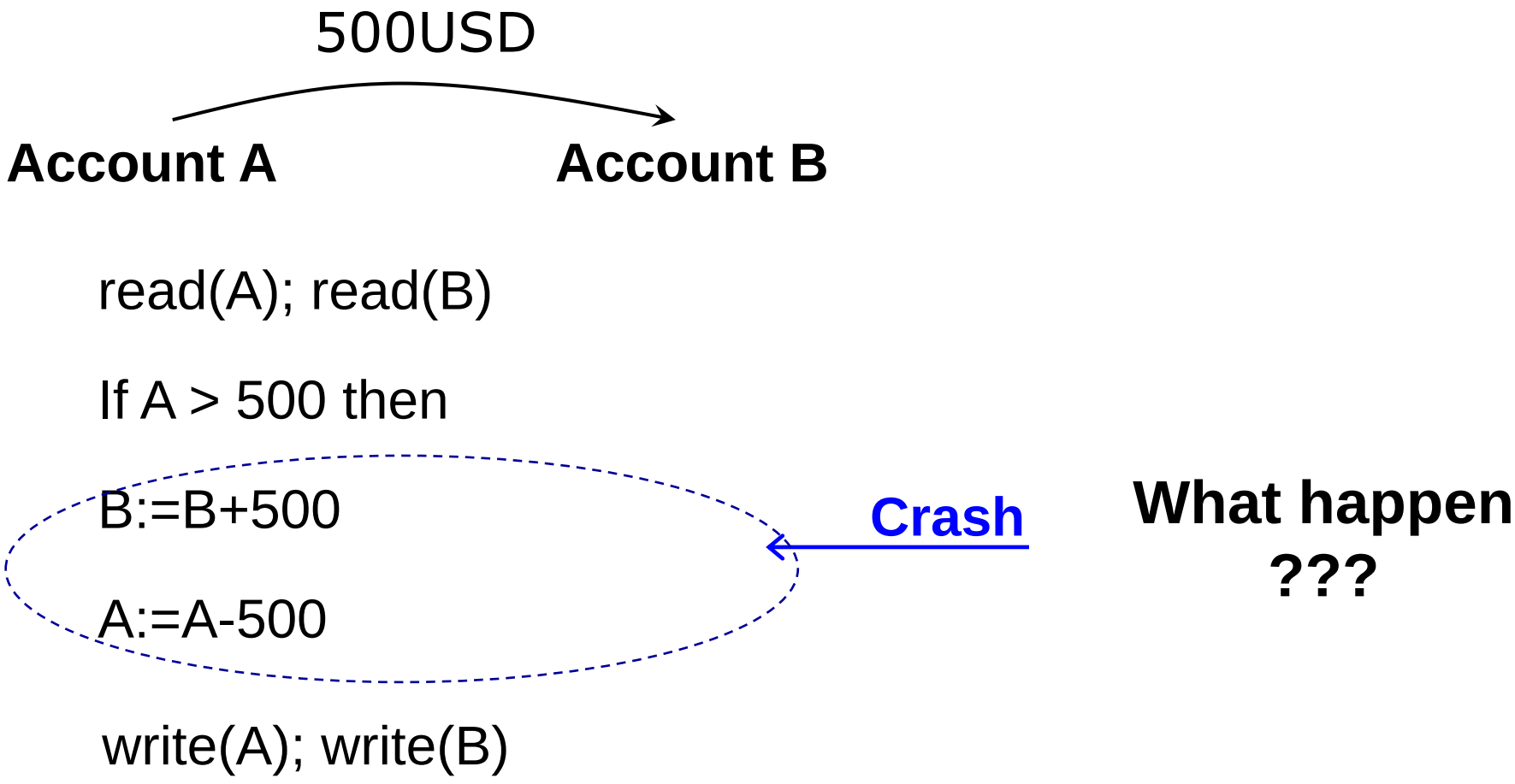
6.4.1. Giao dịch

- Một giao dịch hình thành nên một đơn vị công việc trong một DBMS đối với một CSDL, được coi là cố kết và tin cậy độc lập với các giao dịch khác.
- Giao dịch có nhiều bước, và phải được thực hiện một cách trọn vẹn.
- Trạng thái trung gian giữa các bước là ẩn đối với các giao dịch khác.
- Nếu có sự cố mà giao dịch không thể hoàn thành, thì tất cả các bước không ảnh hưởng lên CSDL.

Ví dụ về giao dịch

- Một CSDL ngân hàng chứa thông tin tài khoản của các khách hàng và tài khoản quỹ của các chi nhánh.
- Giả sử, thực hiện giao dịch chuyển 500 đô-la từ tài khoản của Alice sang tài khoản của Bob:

```
UPDATE accounts SET balance = balance - 500.00
    WHERE name = 'Alice';
UPDATE branches SET balance = balance - 500.00
    WHERE name = (SELECT branch_name FROM accounts
                  WHERE name = 'Alice');
UPDATE accounts SET balance = balance + 500.00
    WHERE name = 'Bob';
UPDATE branches SET balance = balance + 500.00
    WHERE name = (SELECT branch_name FROM accounts
                  WHERE name = 'Bob');
```

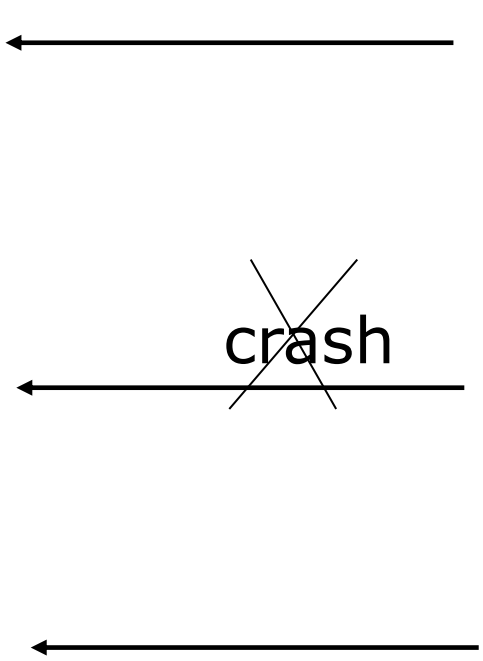

500USD
Account A
Account B
read(A); read(B)
If A > 500 then
B:=B+500
A:=A-500
write(A); write(B)
Crash
What happen ???

6.4.2. Các tính chất của một giao dịch

- Tính nguyên tố (Atomicy):
  - Hoặc là tất cả các thao tác được thực hiện hoặc là không thao tác nào được thực hiện
  - Đảm bảo tính không thể chia cắt, không thể rút gọn
- Tính nhất quán (Consistency): Dữ liệu nhất quán sau khi giao dịch thực hiện.
- Tính cách ly (Isolation): xác định cách mà những thay đổi bởi một thao tác là ẩn với các thao tác đồng thời khác.
- Tính bền vững (Durability):
  - Đảm bảo giao dịch đã được xác nhận sẽ tồn tại vĩnh cửu.
  - ví dụ, một vị trí chỗ ngồi trên máy bay đã được đặt chỗ thì vị trí đó sẽ vẫn trong trạng thái đã đặt chỗ cho dù hệ thống có vấn đề.
- Tất cả 4 tính chất trên, gọi tắt là ACID

Tính nguyên tố

```
T: Read(A,t1);
   If t1 > 500 {
        Read(B,t2);
        t2:=t2+500;
        Write(B,t2);
        t1:=t1-500;
        Write(A,t1);
   }
```



Tính nhất quán

```
T: Read(A,t1);
   If t1 > 500 {
        Read(B,t2);
        t2:=t2+500;
        Write(B,t2);
        t1:=t1-500;
        Write(A,t1);
   }
```

⟵ A+B = C

⟵ A+B = C

Tính cách ly

A= 5000, B= 3000

```
T: Read(A,t1);
   If t1 > 500 {
        Read(B,t2);
        t2:=t2+500;
        Write(B,t2);
        t1:=t1-500;
        Write(A,t1);
   }
```

← *T': A+B (= 5000+3500)*

← *(A+B = 4500+3500)*

Tính bền vững

A= 5000, B= 3000

```
T: Read(A,t1);
   If t1 > 500 {
        Read(B,t2);
        t2:=t2+500;
        Write(B,t2);
        t1:=t1-500;
        Write(A,t1);
   }
```

crash

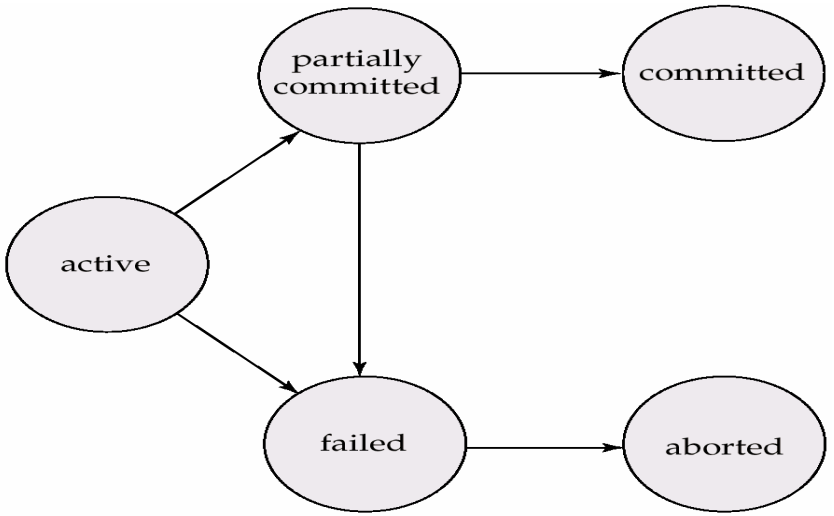
A= 4500, B=3500

- Các thao tác hình thành nên một giao dịch có thể được nhúng trong một chương trình ứng dụng hoặc được xác định bởi ngôn ngữ cấp cao như SQL.
- Để biết một giao dịch diễn ra, cần xác định tường minh câu lệnh bắt đầu (begin transaction) và kết thúc (end transaction) của giao dịch trong chương trình ứng dụng.
- Giao dịch chỉ đọc (read-only)
  - không cập nhật dữ liệu, mà chỉ lấy ra dữ liệu.
- Để đơn giản, giả sử các thao tác truy cập CSDL trong một giao dịch gồm có:
  - read_item(X): đọc 1 khoản mục CSDL tên là X vào một biến trong chương trình cũng tên là X.
  - write_item(X): ghi giá trị của biến X trong chương trình vào khoản mục CSDL cũng tên là X.

- Đơn vị cơ bản của việc chuyển dữ liệu từ đĩa vào bộ nhớ chính là khối (block).
- Thực hiện lệnh read_item(X) bao gồm các bước sau:
  - Tìm địa chỉ của khối trên đĩa mà chứa khoản mục X.
  - Sao chép khối đó vào bộ đệm trên bộ nhớ chính (nếu trên bộ đệm chưa có).
  - Sao chép khoản mục X từ bộ đệm vào biến chương trình X.
- Thực hiện lệnh write_item(X) bao gồm các bước sau:
  - Tìm địa chỉ khối trên đĩa mà nó chứa khoản mục X.
  - Sao chép khối đó vào bộ đệm trong bộ nhớ chính (nếu bộ đệm chưa có).
  - Sao chép khoản mục X từ biến chương trình tên là X vào vị trí chính xác trên bộ đệm.
  - Lưu trữ khối đã cập nhật này từ bộ đệm lên đĩa (có thể ngay tức thì hoặc lưu trữ sau).

Quản lý giao dịch

- Các trạng thái: Active, Failed, Aborted, Committed, Partially committed



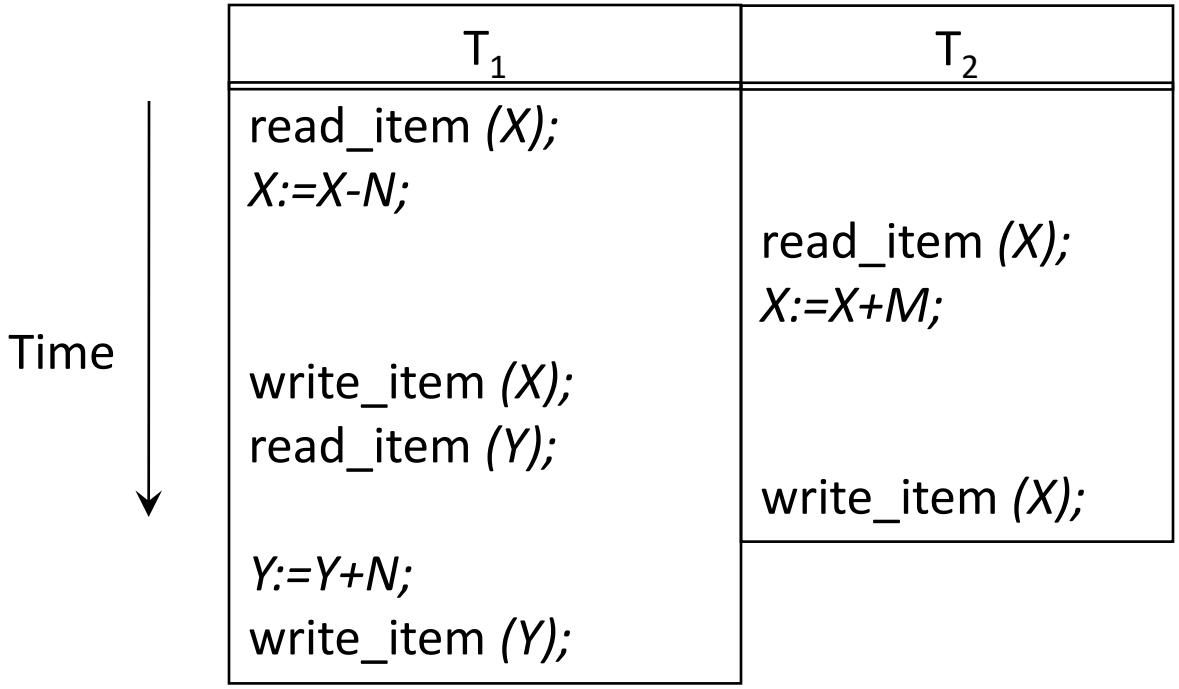
- Lệnh (SQL): Active, Begin Trans, Commit(), Abort(), Savepoint Save(), Rollback (savepoint) (savepoint = 0 ==> Abort)
  (commit để lưu, rollback quay lại trạng thái trước)

6.4.3. Tại sao lại phải điều khiển đồng thời?

- Các vấn đề
  - The Lost Update
  - The Temporary Update (Dirty Read)
  - The Incorrect Summary
  - The Unrepeatable Read

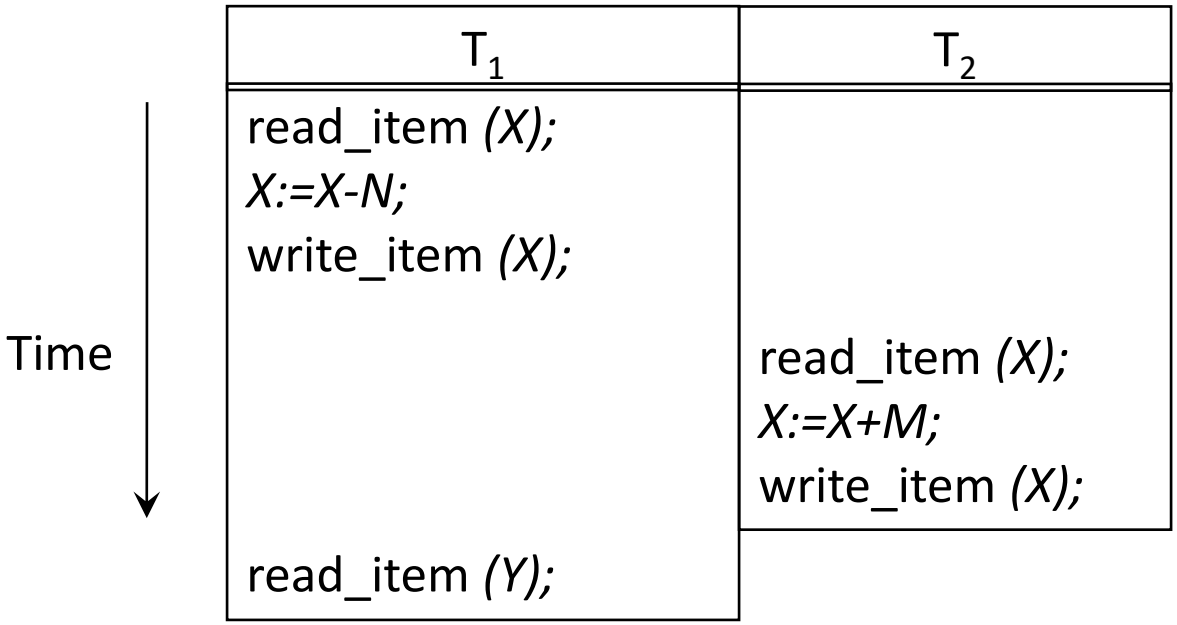
The lost update

- Hiện tượng này xuất hiện khi hai giao dịch truy cập vào cùng các khoản mục dữ liệu nhưng thao tác của hai giao dịch lại xen kẽ, làm cho giá trị của các khoản mục dữ liệu không còn đúng nữa.



The Temporary Update

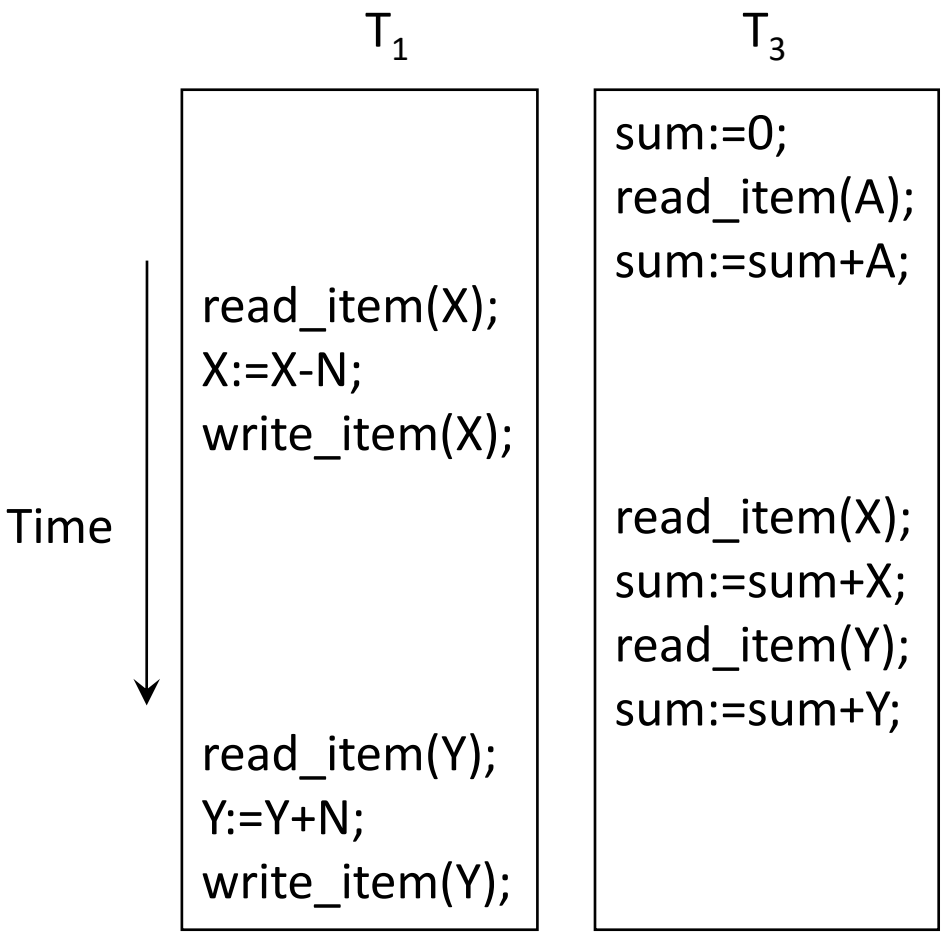
- Xuất hiện khi một giao dịch cập nhật một khoản mục dữ liệu, nhưng sau đó không hoàn thành các bước tiếp theo => giao dịch không trọn vẹn nên phải roll-back giá trị đã cập nhật về giá trị cũ.
- Khoản mục dữ liệu đã được cập nhật kia lại được sử dụng bởi một giao dịch khác trước khi nó được roll-back về giá trị cũ



Giao dịch *T1 fails và phải thay đổi giá trị của X về giá trị cũ của nó; trong khi đó T2* lại đọc giá trị không đúng tạm thời của X.

The Incorrect Summary

- Một giao dịch tính hàm tích lũy trên các bản ghi đang bị cập nhật bởi một giao dịch khác => hàm tích lũy này có thể tính toán dựa trên một số giá trị đã cập nhật và một số giá trị chưa cập nhật.



T3 đọc X sau khi X đã trừ N và đọc Y trước khi Y cộng N

The Unrepeatable Read

- Một giao dịch T đọc một khoản mục dữ liệu hai lần. Khoản mục dữ liệu này bị một giao dịch khác thay đổi giữa hai lần đọc đó.
- Do đó, T nhận các giá trị khác nhau cho hai lần đọc cùng một khoản mục.

Điều khiển đồng thời

- Mục tiêu: Các giao dịch được thực hiện đồng thời mà không vi phạm tính toàn vẹn dữ liệu; Các giao dịch thành công được lưu lại, các giao dịch bị hủy bỏ không còn trong CSDL
- Lịch trình (schedule) của một tập giao dịch là một thứ tự tuyến tính các hành động, ví dụ, R1(X) R2(X) W1(X) W2(X)
- Lịch trình tuần tự có các bước của mỗi giao dịch diễn ra nối tiếp

T_0	T_1
read(A)	
A := A - 50	
write(A)	
read(B)	
B := B + 50	
write(B)	
	read(A)
	temp := A * 0.1
	A := A -temp
	write(A)
	read(B)
	B := B + temp
	write(B)

(1)

T_0	T_1
	read(A)
	temp := A * 0.1
	A := A -temp
	write(A)
	read(B)
	B := B + temp
	write(B)
read(A)	
A := A - 50	
write(A)	
read(B)	
B := B + 50	
write(B)	

(2)

T_0	T_1
read(A)	
A := A - 50	
	read(A)
	temp := A * 0.1
	A := A -temp
	write(A)
	read(B)
write(A)	
read(B)	
B := B + 50	
write(B)	
	B := B + temp
	write(B)

(3)

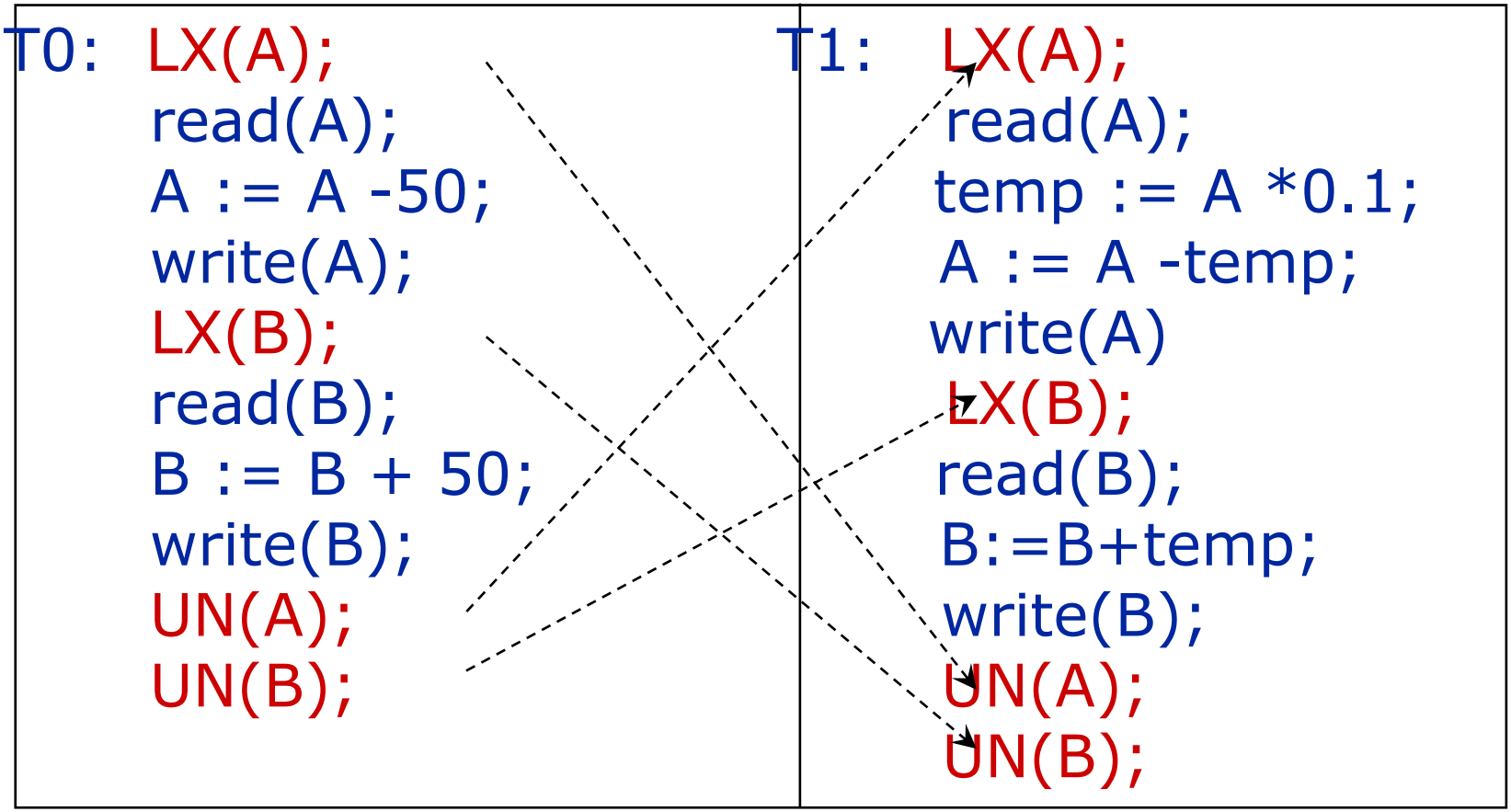
6.4.4. Các kỹ thuật điều khiển đồng thời

- **Kỹ thuật dùng khóa**: Khi một giao dịch cần dữ liệu nào thì xin hệ điều hành một khóa trên phần dữ liệu đó, các giao dịch khác phải đợi đến khi giải phóng khóa mới được sử dụng phần dữ liệu đó. Có thể người ta sử dụng các loại khóa khác nhau ví dụ như khóa đọc – cho phép nhiều giao dịch đọc cùng một lúc, khóa ghi – chỉ một giao dịch có được tại một thời điểm.
- **Kỹ thuật gán nhãn thời gian**: Mỗi giao dịch được gán một nhãn T theo thời gian, giao dịch nào cần được ưu tiên thì được gán nhãn thời gian nhỏ hơn và được thực hiện trước. Kỹ thuật này giúp đưa yêu cầu đồng thời về thực hiện tuần tự.

Khóa

- Shared lock (LS): khóa đọc, chỉ đọc, không ghi
- Exclusive lock (LX): khóa ghi, có thể đọc, ghi
- UN(D): unlock

	LS	LX
LS	true	false
LX	false	false


T0: LX(A);
read(A);
A := A -50;
write(A);
LX(B);
read(B);
B := B + 50;
write(B);
UN(A);
UN(B);
T1: LX(A);
read(A);
temp := A *0.1;
A := A -temp;
write(A)
LX(B);
read(B);
B:=B+temp;
write(B);
UN(A);
UN(B);

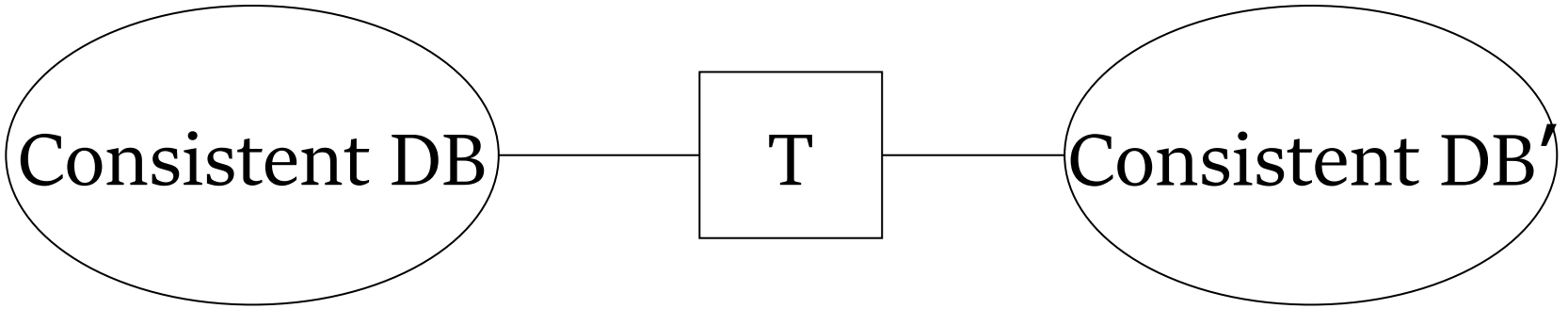
Mức độ cô lập (mức cách ly)

```
Set isolation level  <level>
```

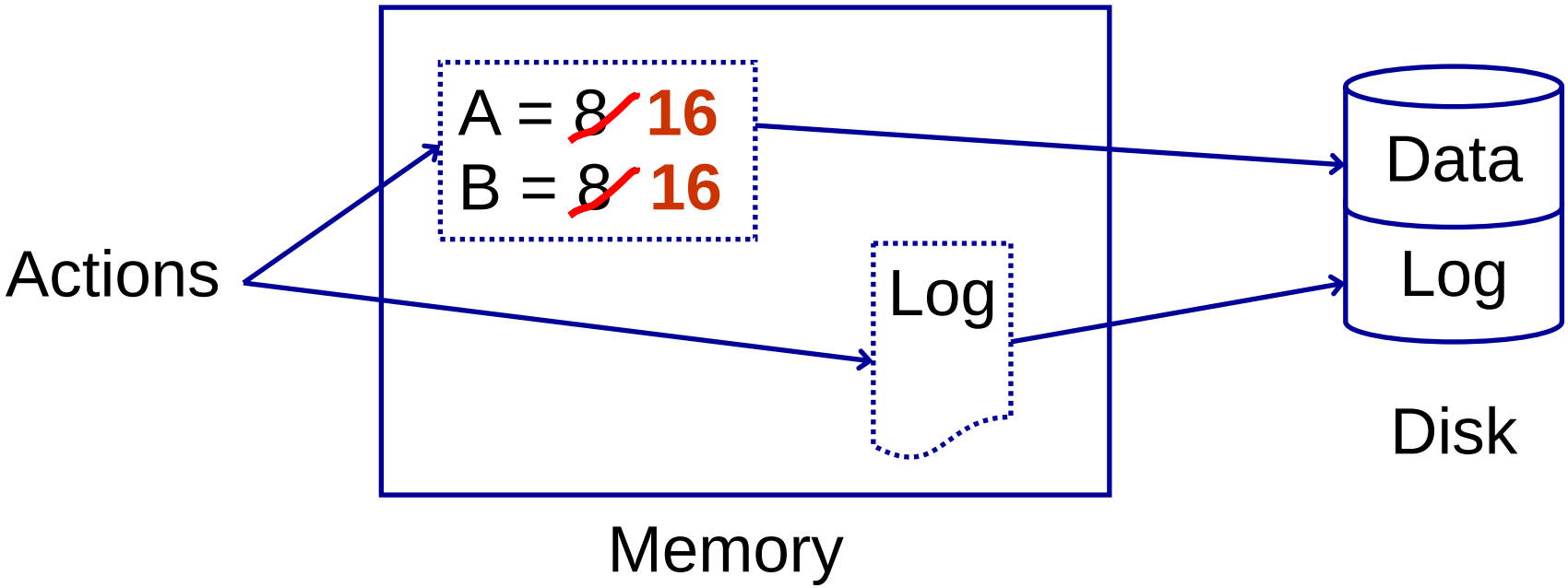
- Read Uncommited: Khóa ghi trong suốt giao dịch, không khóa đọc
- Read Committed: Khóa đọc được giải phóng ngay khi đọc xong, khóa ghi
- Repeatable Read: Khóa đọc và ghi trên khối SQL được chọn
- Serializable: khóa đọc và ghi trên bảng đầy đủ

6.4.5. Khôi phục dữ liệu

- Nếu giao dịch T bắt đầu với Consistent DB, T thỏa tính cách ly, thì T kết thúc với Consistent DB



- Vấn đề - sự cố: lỗi giao dịch, lỗi DBMS, lỗi phần cứng (sự cố đĩa), chia sẻ dữ liệu (T1, T2 song song) ==> Cần phục hồi dữ liệu (ROLLBACK về trạng thái nhất quán gần nhất): sử dụng LOG


A = 8 16
B = 8 16
Log
Actions
Data
Log
Disk
Memory

Undo logging

- Với mọi hành động, tạo bản ghi undo log (giá trị cũ); khi X thay đổi thì bản ghi log liên quan X phải có trên đĩa

Step	Action	t	Mem A	Mem B	Disk A	Disk B	Mem Log
1							<start T>
2	Read(A,t)	8	8		8	8	
3	t:=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8>
5	Read(B,t)	8	16	8	8	8	
6	t:=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8>
8	**Flush log**						
9	Output(A)	16	16	16	16	8	
10	Output(B)	16	16	16	16	16	
11							<commit T>
12	**Flush log**						

Redo logging

- Sử dụng memory lưu các bản ghi log, ghi vào đĩa khi giao dịch hoàn thành

Step	Action	t	Mem A	Mem B	Disk A	Disk B	Mem Log
1							<start T>
2	Read(A,t)	8	8		8	8	
3	t:=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 16>
5	Read(B,t)	8	16	8	8	8	
6	t:=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 16>
8							<commit T>
9	**Flush log**						
10	Output(A)	16	16	16	16	8	
11	Output(B)	16	16	16	16	16	<T, end>

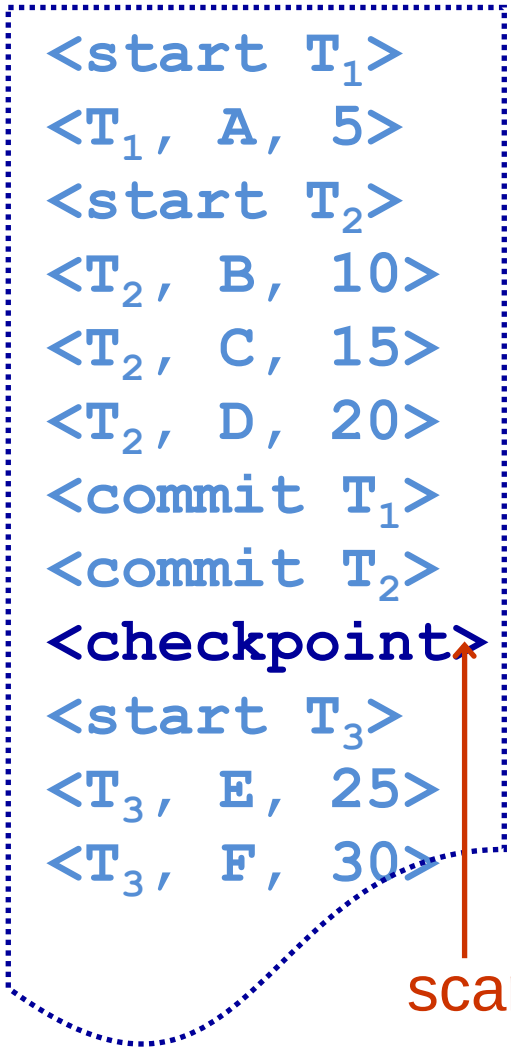
Undo/Redo logging

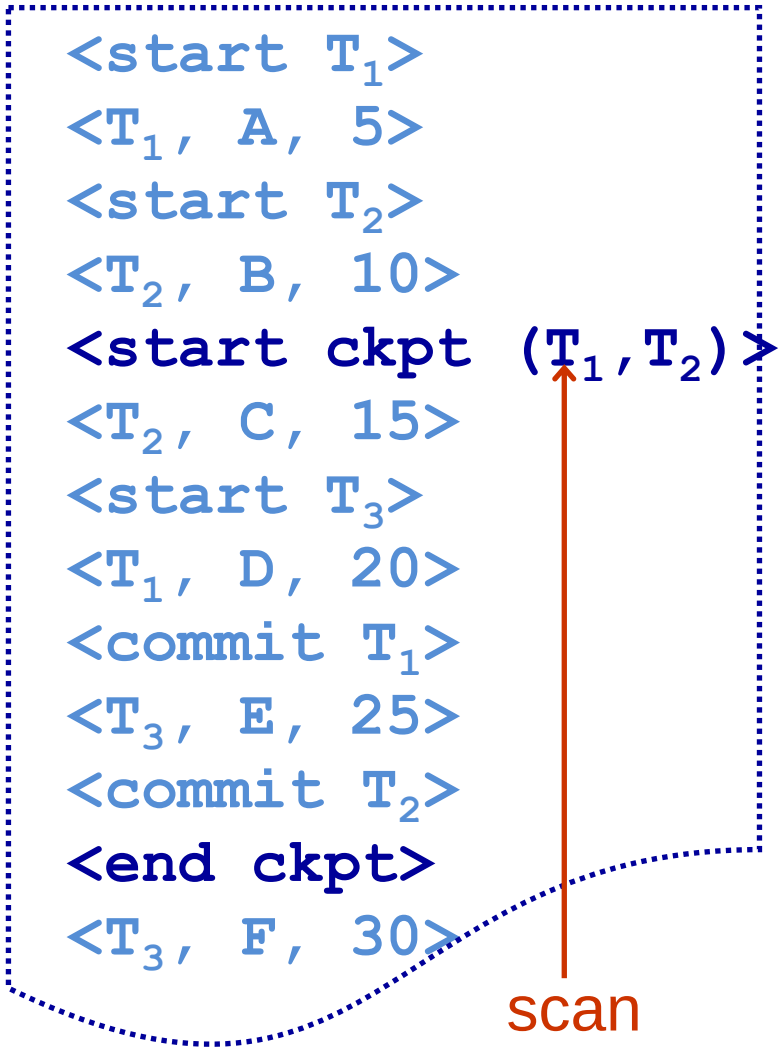
Step	Action	t	Mem A	Mem B	Disk A	Disk B	Mem Log
1							<start T>
2	Read(A,t)	8	8		8	8	
3	t:=t*2	16	8		8	8	
4	Write(A,t)	16	16		8	8	<T, A, 8, 16>
5	Read(B,t)	8	16	8	8	8	
6	t:=t*2	16	16	8	8	8	
7	Write(B,t)	16	16	16	8	8	<T, B, 8, 16>
8	**Flush log**						
9	Output(A)	16	16	16	16	8	
10							<commit T>
11	Output(B)	16	16	16	16	16	

Checkpoint

- Nhật ký giao dịch có thêm <checkpoint> là thời điểm ghi những thay đổi CSDL từ vùng đệm xuống đĩa; Khi đến điểm kiểm tra, DBMS tạm dừng tiếp nhận giao dịch mới, đợi các giao dịch đang thực hiện commit hoặc abort để ghi xuống đĩa ==> Giảm bớt các nhật ký giao dịch không còn cần thiết cho việc khôi phục dữ liệu trong tương lai ==> Giảm thời lượng cần cho phục hồi dữ liệu (chỉ cần duyệt đến điểm kiểm tra gần nhất)
- Điểm kiểm tra linh động: <start ckpt (T1,…)> - <end ckpt> cho phép tiếp nhận các giao dịch mới trong quá trình checkpoint

Điểm kiểm tra cho Undo logging





Chương 7 – Tổ chức dữ liệu vật lý

NỘI DUNG

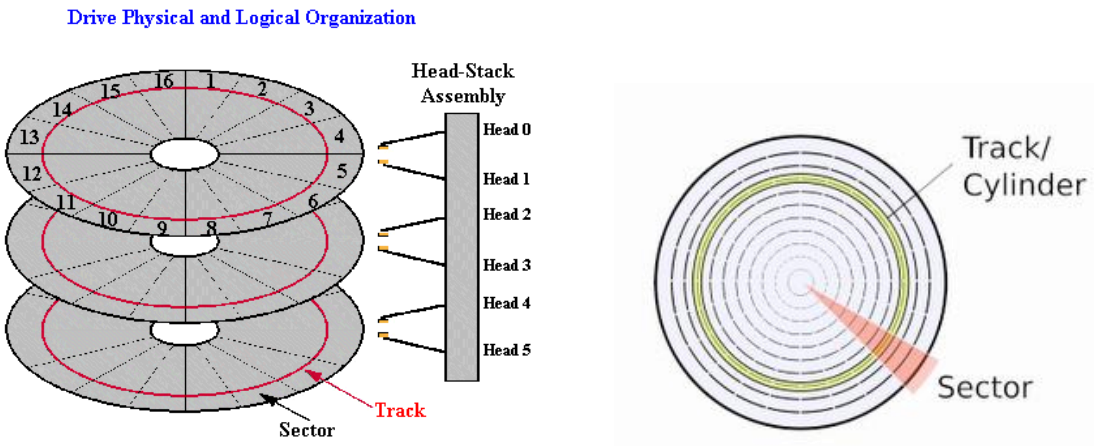
1. Mô hình tổ chức bộ nhớ
2. Tổ chức tệp đống
3. Tổ chức tệp băm
4. Tổ chức tệp chỉ dẫn
5. Cây cân bằng

7.1. Mô hình tổ chức bộ nhớ

- Phương tiện nhớ của máy tính hình thành một phân cấp bộ nhớ bao gồm 2 loại chính:
  - Bộ nhớ sơ cấp:
    - Bao gồm các thiết bị nhớ mà CPU của máy tính có thể thao tác trực tiếp trên đó, như bộ nhớ chính, bộ nhớ đệm cache.
    - Cung cấp cơ thế truy cập dữ liệu nhanh nhưng lại bị giới hạn về dung lượng.
  - Bộ nhớ thứ cấp:
    - Bao gồm các đĩa từ, đĩa quang, băng từ.
    - Dung lượng lớn hơn, chi phí rẻ hơn.
    - Cung cấp cơ chế truy cập dữ liệu chậm hơn.
    - CPU không xử lý dữ liệu trực tiếp trên bộ nhớ thứ cấp mà dữ liệu được sao chép sang bộ nhớ sơ cấp để CPU xử lý.

Bộ nhớ ngoài

- Bộ nhớ ngoài (bộ nhớ thứ cấp): đĩa từ, băng từ,...

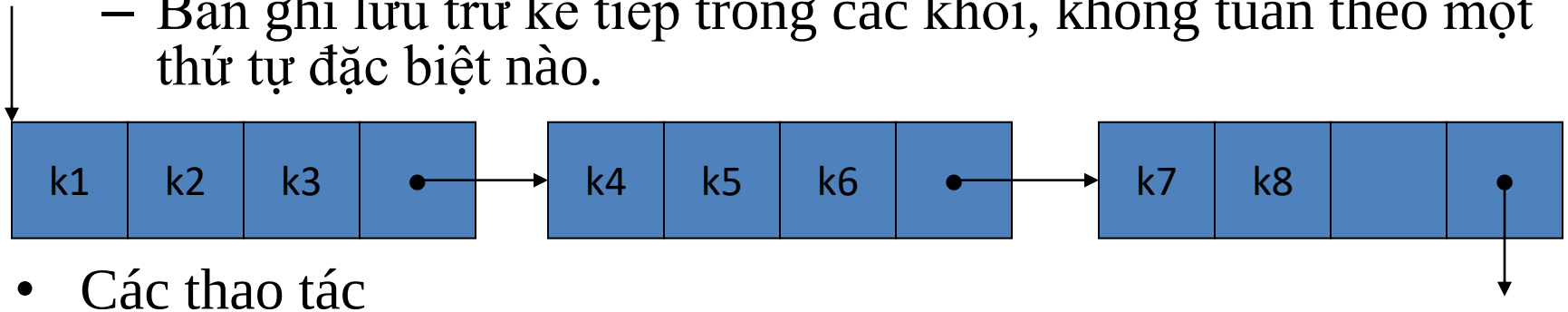


- Đĩa được chia thành các khối vật lý (sector) - 512 byte đến 4096 byte được đánh địa chỉ khối gọi là địa chỉ tuyệt đối
- Mỗi tệp dữ liệu chiếm một hoặc nhiều khối
- Mỗi khối chứa một hoặc nhiều bản ghi

- Thao tác với dữ liệu của tệp thông qua địa chỉ tuyệt đối của các khối.
- Các bản ghi đều có địa chỉ:
  - địa chỉ tuyệt đối của byte đầu tiên
  - địa chỉ khối và số byte tính từ đầu khối đến vị trí đầu bản ghi
- Địa chỉ của các bản ghi/khối được lưu ở một tệp => sử dụng con trỏ (pointer) để truy cập dữ liệu của tệp.

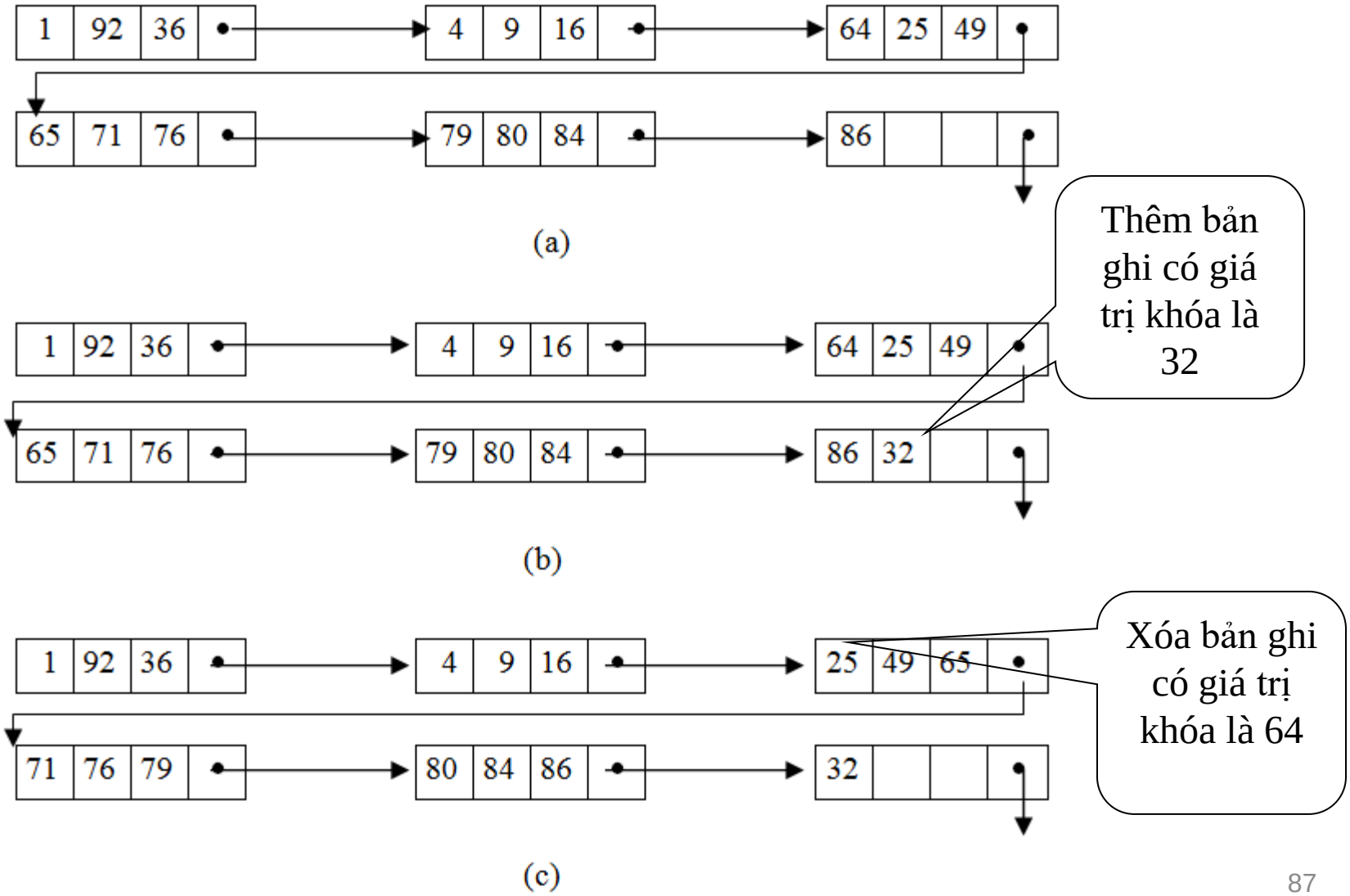
7.2. Tổ chức tệp đống (Heap file)

- Tổ chức dữ liệu
  - Bản ghi lưu trữ kế tiếp trong các khối, không tuân theo một thứ tự đặc biệt nào.



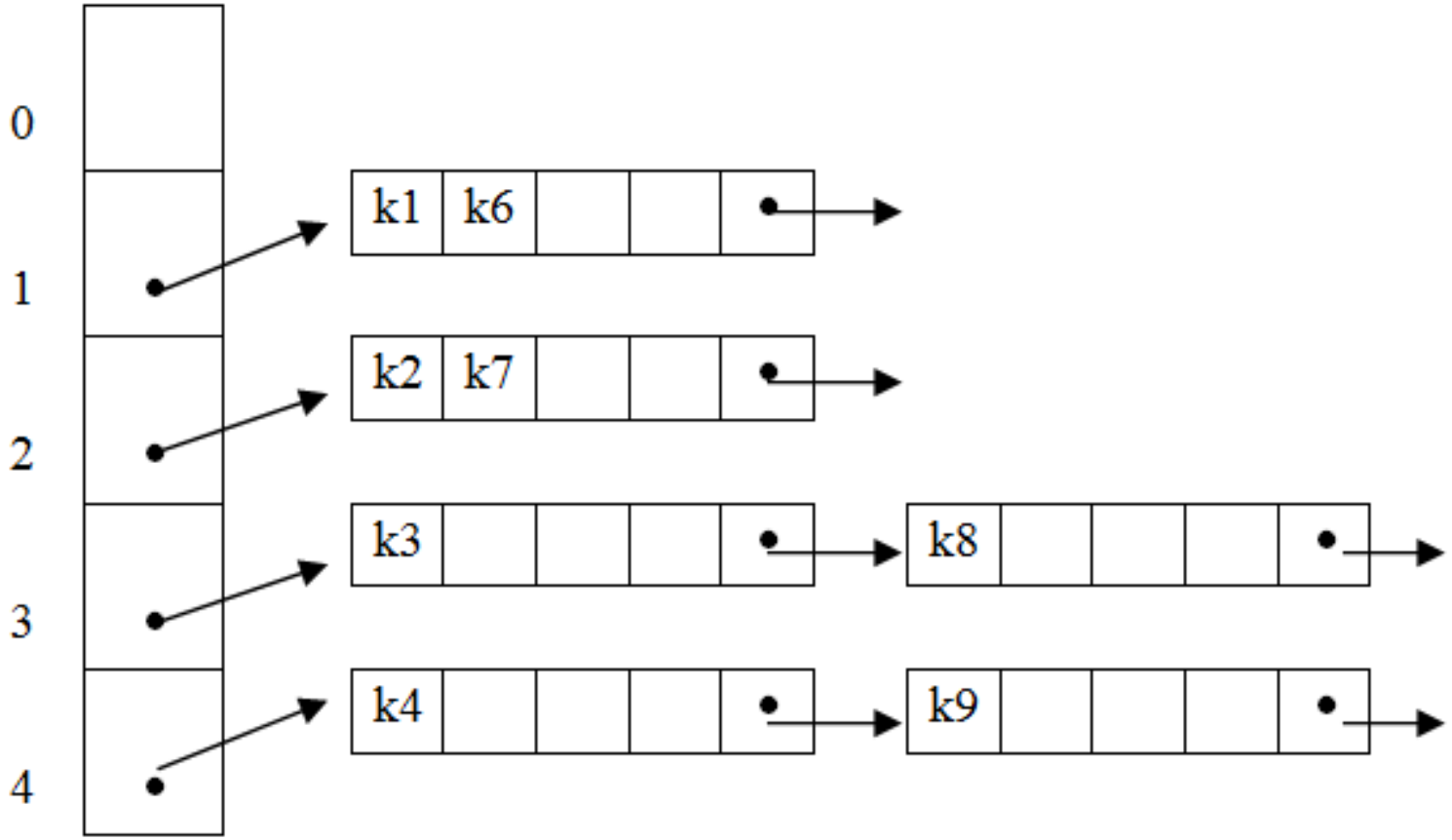
- Các thao tác
  - Tìm kiếm một bản ghi: tìm kiếm một bản ghi có giá trị khóa cho trước => quét toàn bộ tệp.
  - Thêm một bản ghi: thêm bản ghi mới vào sau bản ghi cuối cùng
  - Xóa một bản ghi: thao tác xóa bao hàm thao tác tìm kiếm. Nếu có bản ghi cần xóa thì nó sẽ được đánh dấu là xóa => hệ thống cần tổ chức lại đĩa định kỳ.
  - Sửa một bản ghi: tìm bản ghi rồi sửa một hay nhiều trường.

- **Ví dụ:**

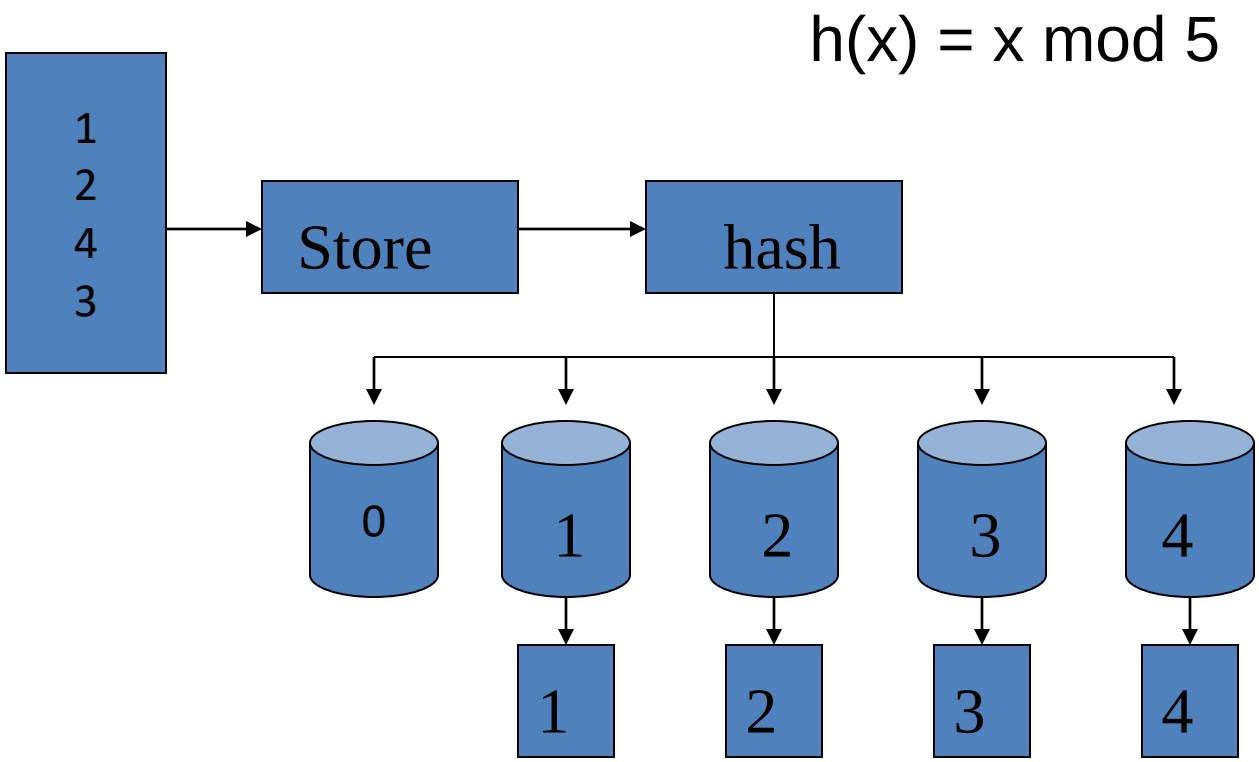

1 92 36
4 9 16
64 25 49
65 71 76
79 80 84
86
(a)
Thêm bản ghi có giá trị khóa là 32
1 92 36
4 9 16
64 25 49
65 71 76
79 80 84
86 32
(b)
Xóa bản ghi có giá trị khóa là 64
1 92 36
4 9 16
25 49 65
71 76 79
80 84 86
32
(c)

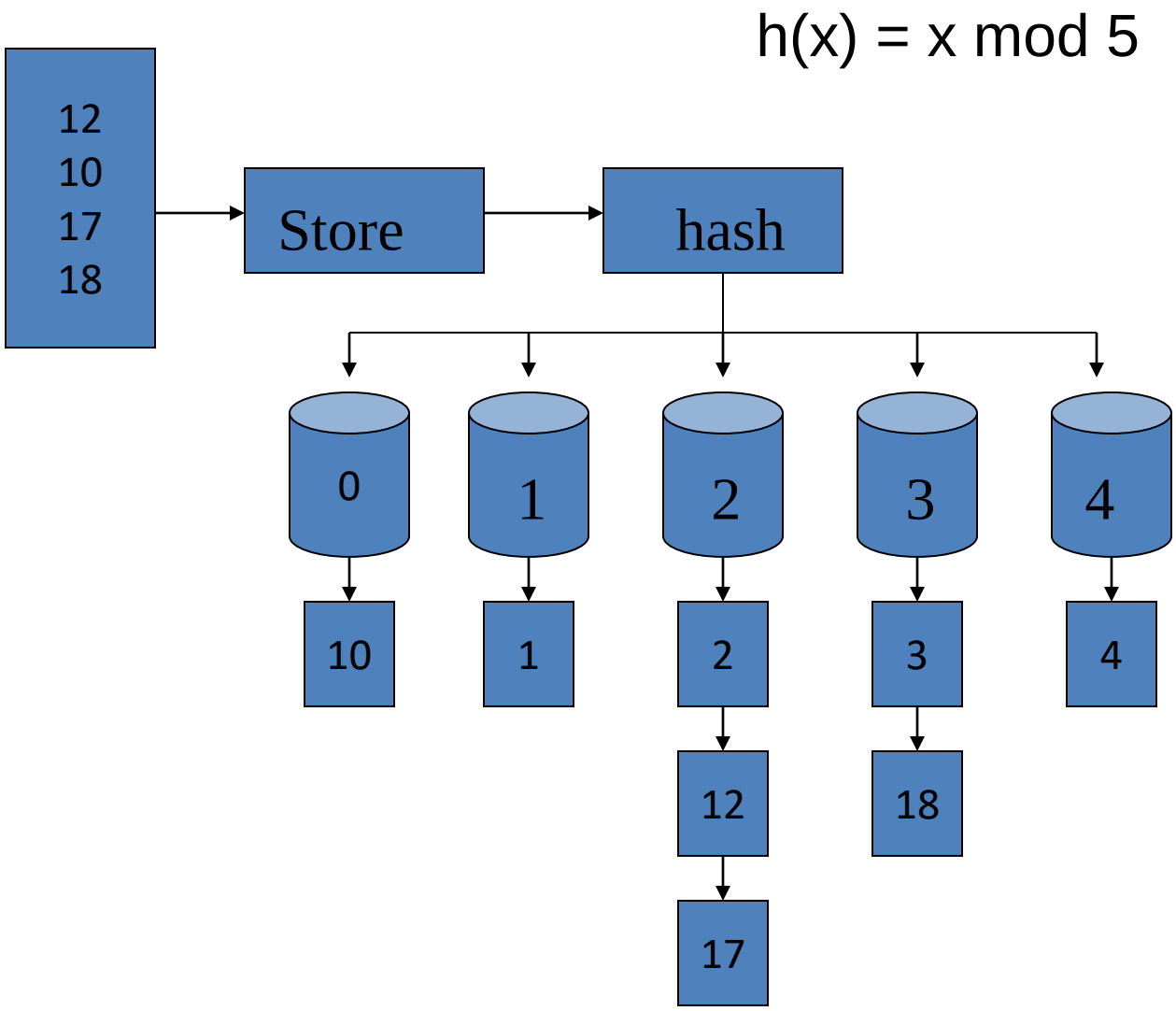
7.3. Tổ chức tệp băm (Hashed files)

- Hàm băm: h(x) nhận một giá trị trong đoạn [0,k], ví dụ: h(x) = x mod k
- Tổ chức tệp dữ liệu
  - Phân chia các bản ghi vào các cụm.
  - Mỗi cụm gồm một hoặc nhiều khối.
  - Mỗi khối chứa số lượng bản ghi cố định.
  - Tổ chức lữu trữ dữ liệu trong mỗi cụm áp dụng theo tổ chức đống
- Tiêu chí chọn hàm băm: phân bố các bản ghi tương đối đồng đều theo các cụm.



Bảng chỉ dẫn cụm

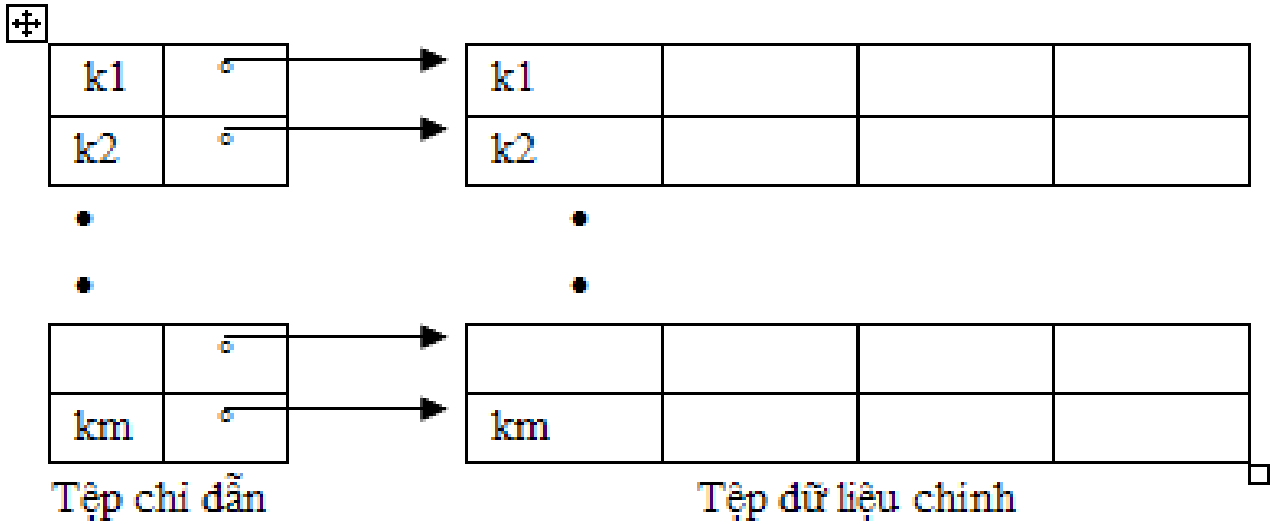

h(x) = x mod 5
1
2
4
3
Store
hash
0
1
2
3
4
1
2
3
4


h(x) = x mod 5
12
10
17
18
Store
hash
0
1
2
3
4
10
1
2
3
4
12
18
17

- Các thao tác:
  - Tìm kiếm một bản ghi: để tìm bản ghi có khóa x, tính h(x) sẽ được cụm chứa bản ghi, sau đó tìm kiếm theo tổ chức đống.
  - Thêm một bản ghi: thêm một bản ghi có giá trị khóa là x.
    - nếu trong tệp đã có một bản ghi có trùng khóa x =>bản ghi mới sai (vì khóa là duy nhất!)
    - nếu không có bản ghi trùng khóa, bản ghi được thêm vào khối còn chỗ trống đầu tiên trong cụm, nếu hết chỗ thì tạo khối mới.
  - Xóa một bản ghi: tìm kiếm bản ghi rồi xóa
  - Sửa đổi một bản ghi:
    - nếu trường cần sửa có tham gia vào trong khóa thì việc sửa sẽ là loại bỏ bản ghi này và thêm mới một bản ghi (bản ghi có thể thuộc vào một cụm khác)
    - nếu trường cần sửa không thuộc khóa: tìm kiếm rồi sửa. Nếu bản ghi không tồn tại thì xem như có lỗi.

7.4. Tổ chức tệp chỉ dẫn(Indexed Files)

- Giả sử giá trị các khóa của các bản ghi được sắp xếp tăng dần.
- Tệp chỉ dẫn được tạo bằng cách chọn các giá trị khóa trong các bản ghi
- Tệp chỉ dẫn bao gồm các cặp (k,d), trong đó k là giá trị khoá của bản ghi đầu tiên, d là địa chỉ của khối (hay con trỏ khối).

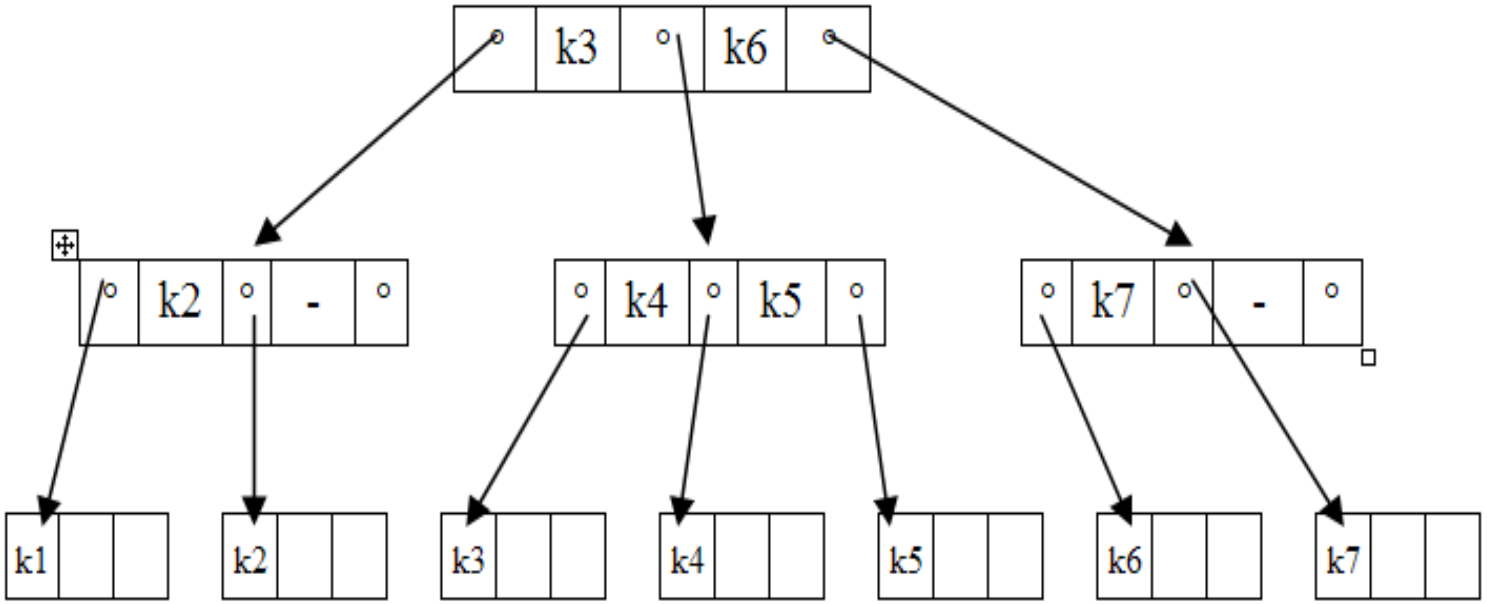


- Tìm kiếm trên tệp chỉ dẫn
  - Cho một giá trị khóa k_i, tìm một bản ghi (k_m,d) trong tệp chỉ dẫn sao cho $k_m <= k_i$ và:
    - hoặc (k_m,d) là bản ghi cuối cùng trong tệp chỉ dẫn
    - hoặc bản ghi tiếp theo (k_{m+1},d') thỏa mãn $k_i<k_{m+1}$
  - Khi đó, chúng ta nói k_m phủ k_i
  - Tìm kiếm này có thể là:
    - tuần tự
    - nhị phân

- Các thao tác:
  - Tìm kiếm một bản ghi
  - Thêm một bản ghi: xác định khối i sẽ chứa bản ghi đó
    - nếu trong khối i còn chỗ thì đặt bản ghi này vào đúng chỗ theo thứ tự sắp xếp của khóa, dồn toa các bản ghi đằng sau nó.
    - nếu khối i hết chỗ thì việc thêm này sẽ đẩy bản ghi cuối cùng trong khối sang làm bản ghi đầu tiên của khối tiếp theo i+1 => sửa bản ghi chỉ dẫn tương ứng
    - nếu bản ghi mới này có giá trị khóa lớn hơn tất cả mọi khóa trong tệp dữ liệu chính và không còn chỗ thì tạo thêm một khối mới.
  - Xóa một bản ghi: giống như thêm một bản ghi, nếu xóa mà tạo thành một khối rỗng, khi đó có thể loại bỏ cả khối đó.
  - Sửa một bản ghi:
    - Sử dụng thủ tục tìm kiếm để xác định bản ghi cần sửa
    - nếu các trường cần sửa không phải là khóa thì sửa bình thường
    - nếu các trường cần sửa tham gia vào khóa thì quá trình sửa sẽ là quá trình thêm và xóa một bản ghi.

7.5. Cây cân bằng(Balanced-trees)

- B-tree được tổ chức theo cấp m, có các tính chất sau đây:
  - Gốc của cây hoặc là một nút lá hoặc ít nhất có hai con.
  - Mỗi nút (trừ nút gốc và nút lá) có từ [m/2] đến m con.
  - Mỗi đường đi từ nút gốc đến bất kỳ nút lá nào đều có độ dài như nhau.



- Cấu trúc của mỗi nút trong B-cây có dạng (p_0, k_1, p_1, k_2,...,k_n, p_n) với p_i (i=1..n) là con trỏ trỏ tới khối i của nút có k_i là khoá đầu tiên của khối đó. Các khoá k trong một nút được sắp xếp theo thứ tự tăng dần.
- Mọi khoá trong cây con, trỏ bởi con trỏ p_0 đều $\leq k_1$;
- Mọi khoá trong cây con, trỏ bởi con trỏ p_i đều nhỏ hơn k_{i+1}.
- Mọi khoá trong cây con, trỏ bởi con trỏ p_n đều $\geq k_n$.

- Các thao tác:
  - Tìm kiếm một bản ghi: xác định đường dẫn từ nút gốc tới nút lá chứa bản ghi này
  - Thêm một bản ghi:
    - Xác định vị trí nút lá sẽ chứa bản ghi này (như tìm kiếm)
    - Nếu còn chỗ thì thêm bình thường
    - Nếu hết chỗ thì phải tạo thêm nút lá mới, chuyển nửa dữ liệu cuối của nút lá hiện tại sang nút mới, sau đó thêm bản ghi mới này vào vị trí phù hợp nút lá hiện tại hoặc nút mới tạo
    - Rất có khả năng "động chạm" đến nút cha,….nút gốc.
  - Loại bỏ một bản ghi:
    - Dùng thủ tục tìm kiếm một bản ghi để xác định nút L có thể chứa bản ghi đó.
    - Rất có khả năng "động chạm" đến nút cha,…,nút gốc.

Kết luận

- Tổ chức tệp chỉ dẫn:
  - được áp dụng phổ biến
  - Với các ứng dụng yêu cầu cả xử lý tuần tự và truy nhập trực tiếp đến các bản ghi
  - Hiệu năng sẽ giảm khi kích thước tệp tăng =>chỉ dẫn B-cây
- Tổ chức băm:
  - Dựa trên một hàm băm, cho phép tìm thấy địa chỉ khoản mục dữ liệu một cách trực tiếp
  - Hàm băm tốt? Phân bố các bản ghi đồng đều trong các cụm