Chương 3 (tt)

Phục hồi dữ liệu và An toàn dữ liệu

Nội dung

- Các PP điều khiển truy cập (Access control)
  - DAC.
  - MAC.
  - RBAC.

Có 3 kiểu điều khiển truy cập

- DAC (Discretionary Access Control)
  - Cho biết chủ thể nào có thể truy cập kiểu gì đến các đối tượng CSDL.
  - Có những nguyên tắc để 1 chủ thể có thể tùy ý cấp quyền hay lấy lại quyền cho/ từ 1 chủ thể khác.
- MAC (Mandatory Access Control)
  - Định trước các nguyên tắc để chủ thể (thuộc 1 lớp) truy cập trực tiếp hoặc gián tiếp đến các lớp dữ liệu.
- RBAC (Role-based Access Control
  - Vai trò là 1 tập các quyền. Không thực hiện cấp quyền cho từng chủ thể mà gán cho chủ thể 1 vai trò, khi đó chủ thể sẽ có tất cả các quyền thuộc vai trò đó.

- DAC

DAC

- Điều khiển truy cập nhiệm ý là điều khiển việc truy cập của chủ thể vào đối tượng thông qua định danh của chủ thể và các luật định trước.
- Cơ chế này được gọi là nhiệm ý là do nó cho phép chủ thể có thể cấp quyền cho chủ thể khác truy cập các đối tượng của nó.

AC ở các HQT CSDL thương mại

- Tất cả các HQT CSDL thương mại đều cài đặt DAC.
- Các mô hình cấp quyền theo cơ chế DAC hiện tại đều dựa trên mô hình System R.
- System R: do Griffiths và Wade phát triển vào 1976, là một trong những mô hình ra đời đầu tiên cho hệ quản trị cơ sở dữ liệu quan hệ.
- System R: dựa trên nguyên lý ủy quyền quản trị cho người sở hữu.

System R

- Các đối tượng được quản lý trong mô hình là table và view.
- Các phương thức truy cập (privilege) là select, insert, update, delete, drop, index (chỉ cho table), alter (chỉ cho table).
- Hỗ trợ làm việc trên group, không hỗ trợ role.
- Dùng câu lệnh GRANT để cấp quyền, có tùy chọn GRANT OPTION.

System R

- Sự ủy quyền được thể hiện thông qua GRANT OPTION, nếu cấp quyền cho 1 user bằng câu lệnh có GRANT OPTION thì user đó ngoài việc có thể thực hiện quyền được cấp, còn có thể cấp quyền đó cho các user khác.
- 1 user chỉ có thể cấp quyền trên 1 quan hệ cho trước nếu user đó là người sở hữu quan hệ hoặc user đó được người khác cấp quyền với câu lệnh có tuỳ chọn là GRANT OPTION.

Câu lệnh GRANT

GRANT *PrivilegeList*| ALL[PRIVILEGES]
ON *Relation* | *View*
TO *UserList* | PUBLIC
[WITH GRANT OPTION]

- Có thể cấp quyền (privilege) trên table và view.
- Privilege áp dụng trên cả table (hoặc view).
- Đối với quyền cập nhật (update priviledge), cần phải chỉ định quyền này áp dụng trên cột nào.

Câu lệnh GRANT – Ví dụ

A: GRANT select, insert ON NHANVIEN TO B
WITH GRANT OPTION;
A: GRANT select ON NHANVIEN TO C
WITH GRANT OPTION;
B: GRANT select, insert ON NHANVIEN TO C;

- C có quyền select (từ A và B) và quyền insert (từ B).
- C có thể cấp quyền select cho các user khác, nhưng C không thể cấp quyền insert.

Câu lệnh GRANT

- Với từng user, hệ thống ghi nhận lại:
  - A: tập hợp các quyền mà user sở hữu.
  - B: tập hợp các quyền mà user có thể ủy quyền cho người khác.

- Khi 1 user thi hành câu lệnh GRANT, hệ thống sẽ
  - Tìm phần giao của tập B với tập quyền trong câu lệnh.
  - Nếu phần giao là rỗng, thì câu lệnh không được thi hành.

Câu lệnh GRANT – Ví dụ

A: GRANT select, insert ON NHANVIEN TO C WITH GRANT OPTION;

A: GRANT select ON NHANVIEN TO B WITH GRANT OPTION;

A: GRANT insert ON NHANVIEN TO B;

C: GRANT update ON NHANVIEN TO D WITH GRANT OPTION;

B: GRANT select, insert ON NHANVIEN TO D;

Câu lệnh GRANT – Ví dụ

- Trong ví dụ trên, hãy cho biết:
  - Câu lệnh nào được thực thi hoàn toàn?
  - Câu lệnh nào không được thực thi?
  - Câu lệnh nào thực thi một phần?

  TRẢ LỜI:
    - 3 câu lệnh đầu tiên thực thi hoàn toàn vì A là người sở hữu table.
    - Câu lệnh thứ 4 không thực thi vì C không có quyền update trên table.
    - Câu lệnh thứ 5 thực thi một phần, B có quyền select, insert nhưng B không có grant option cho quyền insert nên D chỉ nhận được quyền select.

Câu lệnh Revoke

REVOKE *PrivilegeList*|
ALL[PRIVILEGES]
ON *Relation* | *View*
FROM *UserList* | PUBLIC

- Câu lệnh này dùng để thu hồi quyền đã cấp.
- User chỉ có thể thu hồi quyền mà user đã cấp.
- User không thể chỉ thu hồi grant option.
- Một người chỉ có thể bị thu hồi quyền truy xuất p khi tất cả những người cấp quyền p cho họ đều thu hồi quyền p lại.

Câu lệnh Revoke – Ví dụ

A: GRANT select ON NHANVIEN TO C WITH GRANT OPTION;

A: GRANT select ON NHANVIEN TO B WITH GRANT OPTION;

C: GRANT insert ON NHANVIEN TO D;

B: GRANT select ON NHANVIEN TO D;

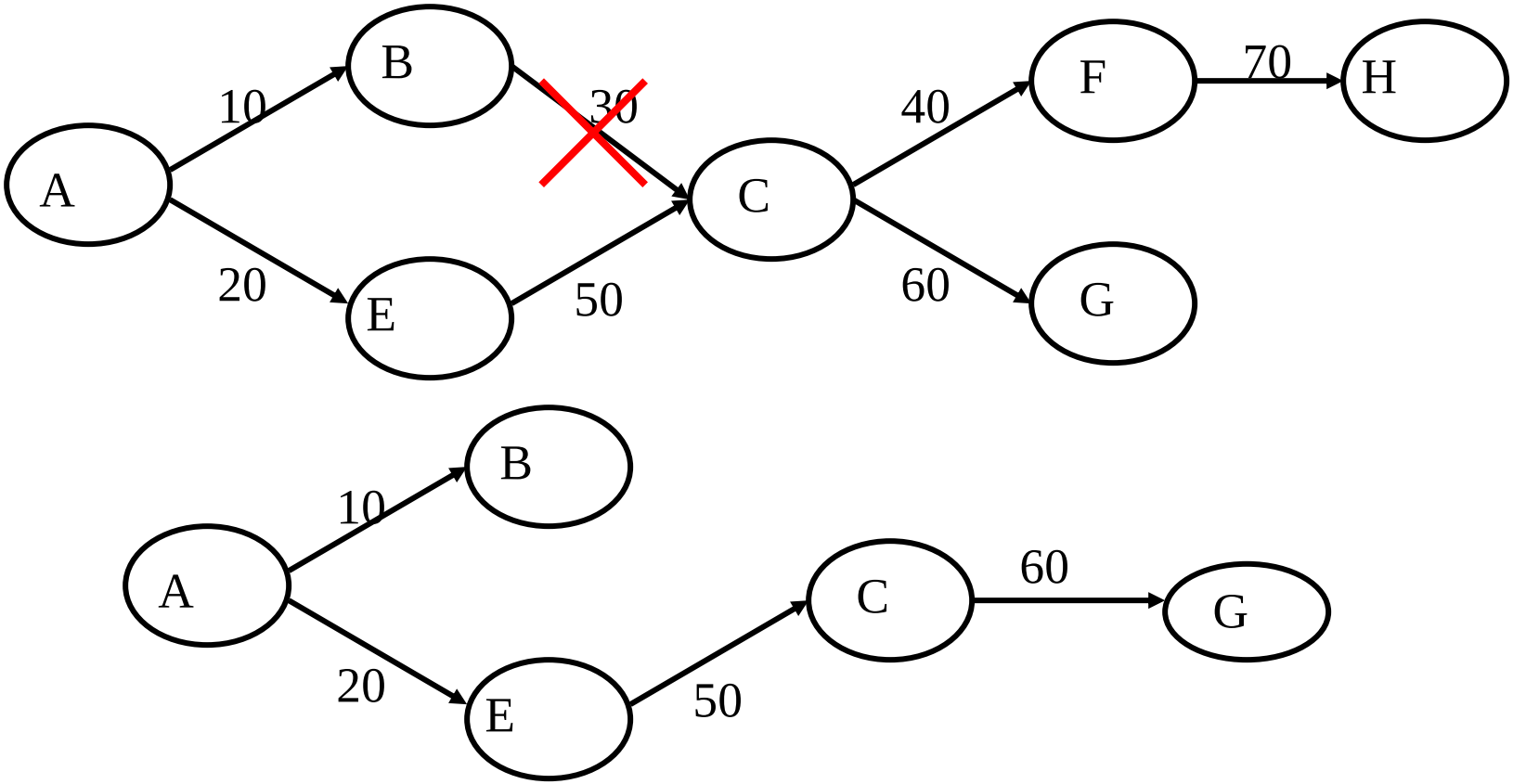
C: REVOKE select ON NHANVIEN FROM D;

- Sau câu lệnh này thì D vẫn có quyền select trên bảng NHANVIEN vì B vẫn chưa thu hồi quyền select của D.

Câu lệnh Revoke

- Thu hồi quyền đệ quy (recursive revocation): khi người dùng A thu hồi quyền truy xuất trên bảng của một người B thì tất cả các quyền mà B đã gán cho người khác đều bị thu hồi.

Thu hồi quyền đệ quy



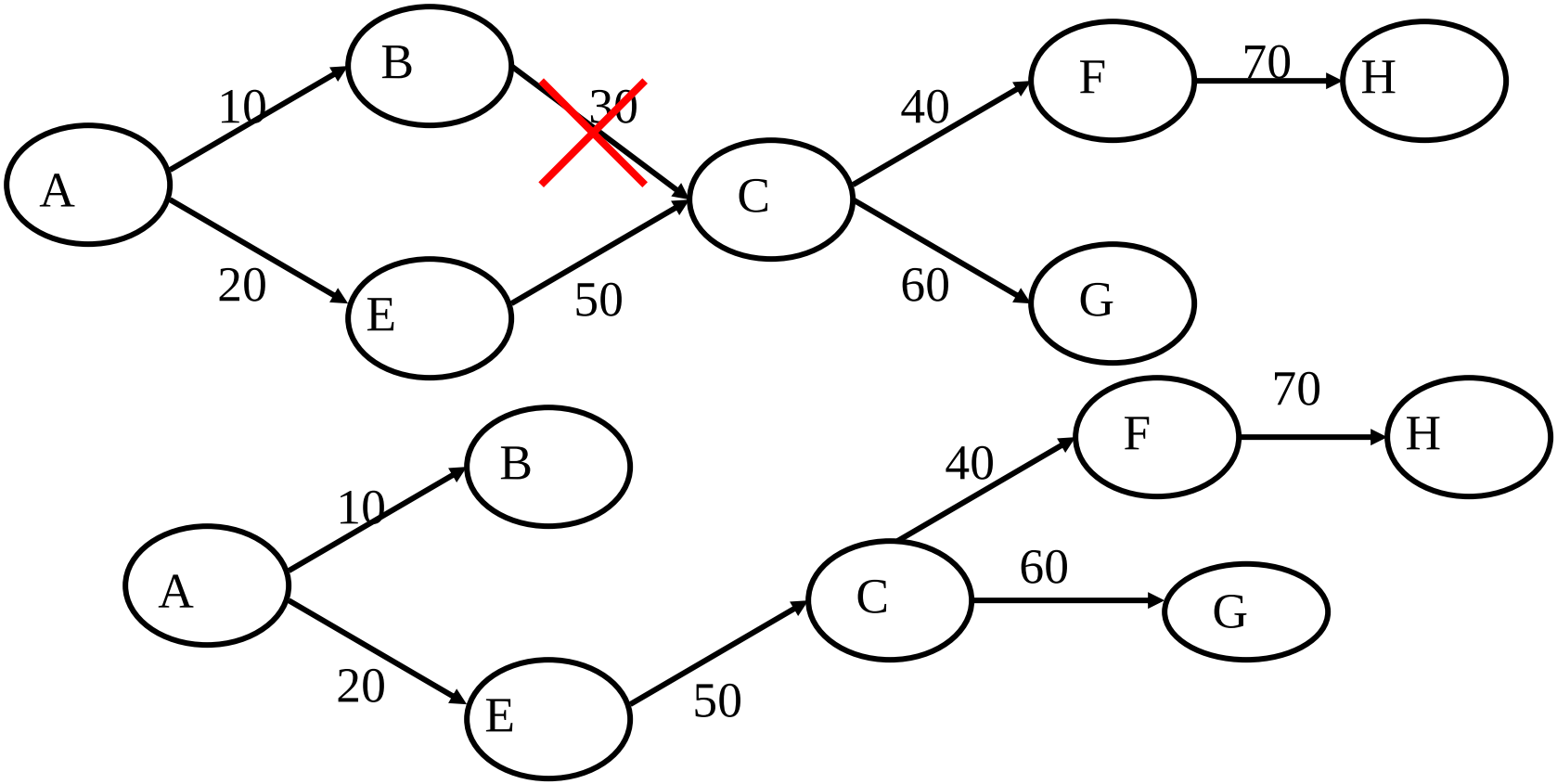
Thu hồi quyền đệ quy

- Sẽ phá vỡ quyền truy xuất mà đối tượng bị thu hồi quyền đã cấp.
- Thực tế khi 1 user A thay đổi công việc hay vị trí thì chỉ muốn lấy lại quyền truy xuất của A mà không muốn lấy lại các quyền truy xuất mà A đã cấp.

Thu hồi quyền đệ quy

- Thu hồi quyền đệ quy trong System R dựa vào nhãn thời gian mỗi lần cấp quyền truy xuất cho người dùng.
- Một biến thể của cách tiếp cận này là không dựa vào nhãn thời gian, mục đích là để tránh thu hồi quyền dây chuyền.
- Khi đó, nếu C bị B thu hồi quyền và C lại có quyền tương đương do người khác cấp (mặc dù sau đó) thì quyền mà C cấp cho người khác vẫn được giữ.

Một biến thể của thu hồi quyền đệ quy



Thu hồi quyền không dây chuyền (Noncascading revoke)

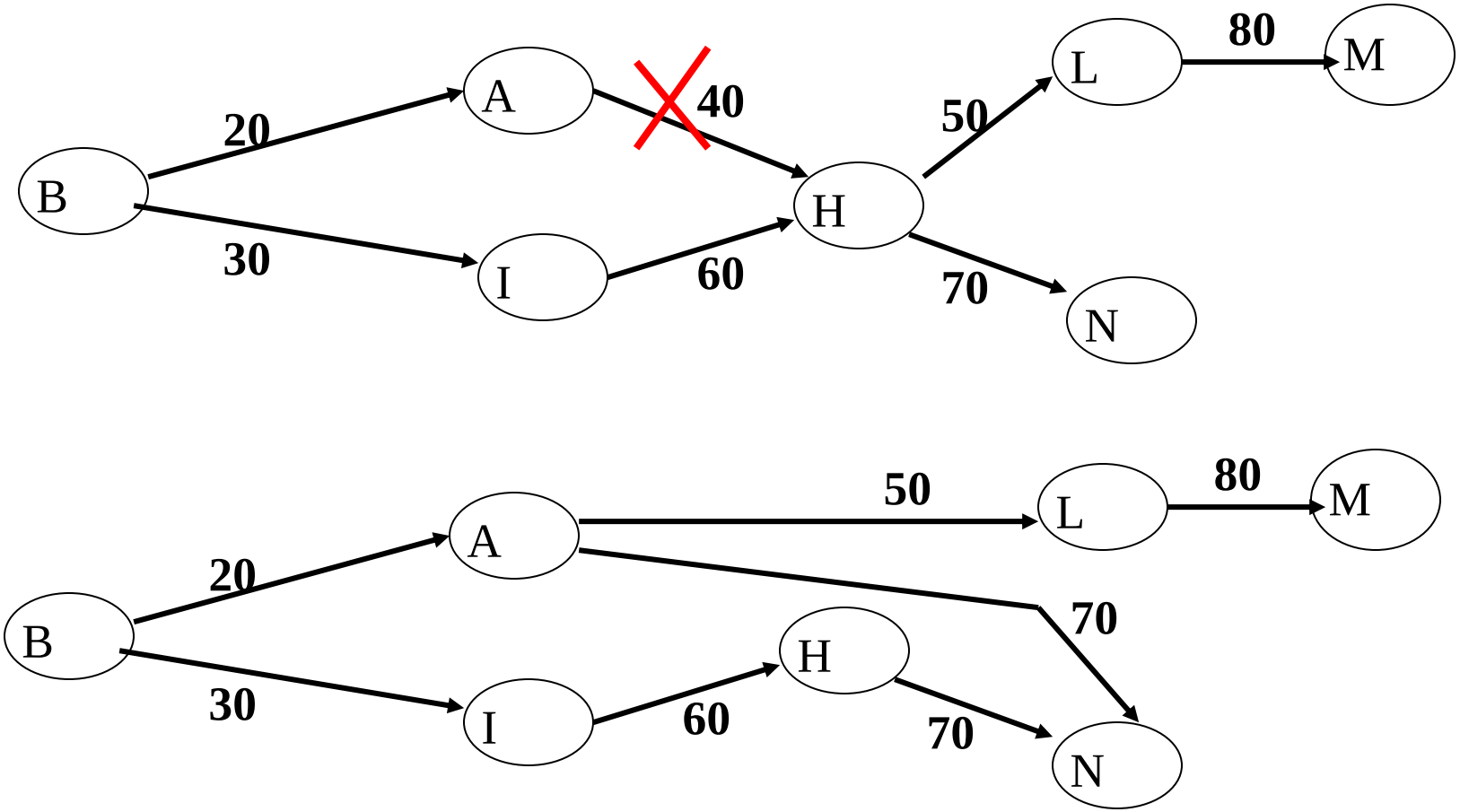
- Khi A thu hồi quyền truy xuất trên B thì tất cả quyền truy xuất mà B đã cấp cho chủ thể khác được thay bằng A đã cấp cho những chủ thể này.

Thu hồi quyền không dây chuyền

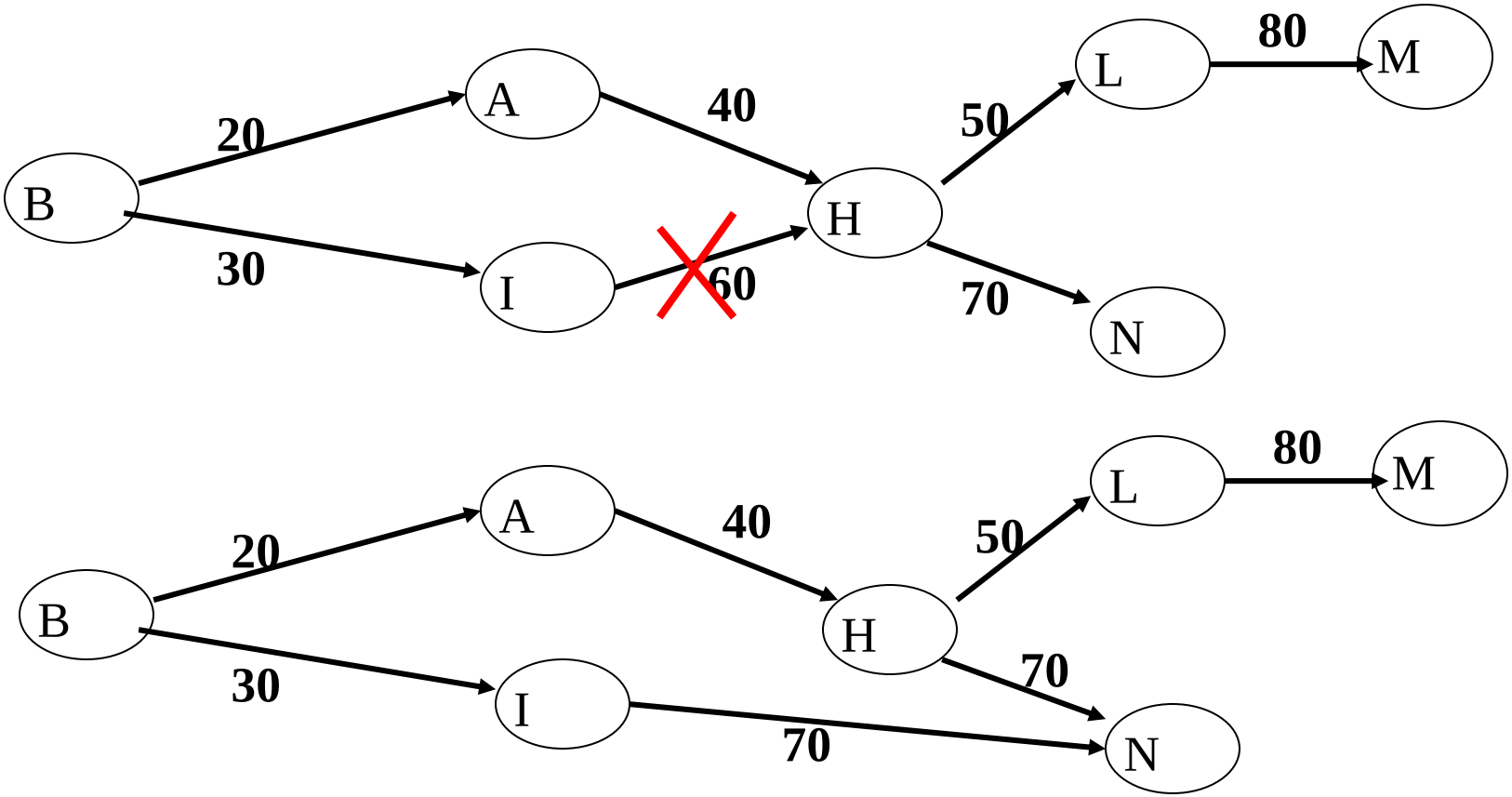
Chú ý:

- Bởi vì B được cấp quyền truy xuất trên đối tượng từ nhiều chủ thể (khác A), nên không phải tất cả các quyền mà B cấp đều thay bằng A cấp. Và A được xem là người cấp thay cho B khi B sử dụng quyền A đã cấp cho mình cấp cho những chủ thể khác.
- A sẽ là người cấp các quyền mà B đã cấp sau khi nhận quyền đó từ A có chỉ định WITH GRANT OPTION. Với những quyền B đã được cấp bởi C ≠ A, đến lượt B cấp cho người khác thì B vẫn là người cấp các quyền này.

Thu hồi quyền không dây chuyền



Thu hồi quyền không dây chuyền



Thu hồi quyền không dây chuyền

- Lưu ý rằng với quyền mà H cấp cho L, sau khi thu hồi quyền, không được thay I như là người cấp vì quyền này được cấp trước khi I cấp quyền cho H.

View và sự phân quyền dựa trên nội dung

- Trong các RDBMS, view là một cơ chế thường được dùng để hỗ trợ việc điều khiển truy cập dựa trên nội dung
- Dùng các vị từ (predicate) để giới hạn nội dung dữ liệu cần cấp quyền.
- Chỉ những bộ của quan hệ thỏa mãn vị từ được xem là các đối tượng để cấp quyền.

View và sự phân quyền dựa trên nội dung

- Việc điều khiển truy cập dựa trên nội dung trong các RDBMS được thực hiện như sau:
  - Định nghĩa 1 view dùng các vị từ để chọn ra các dòng dữ liệu muốn cấp cho chủ thể S.
  - Cấp cho S các quyền trên view.

View và sự phân quyền dựa trên nội dung

- Ví dụ: giả sử ta muốn cấp quyền cho user B truy cập chỉ những nhân viên có lương ít hơn 20000:
  - CREATE VIEW V_NHANVIEN AS
    SELECT * FROM NHANVIEN
    WHERE LUONG < 20000;
  - GRANT Select ON V_NHANVIEN TO B;

Các bước xử lý truy vấn

- Parsing
- Catalog lookup
- Authorization checking
- View Composition
  - Truy vấn trên view sẽ được chuyển thành truy vấn trên bảng cơ sở thông qua bước này.
  - Kết quả sẽ dựa trên vị từ của câu truy vấn và vị từ định nghĩa nên view.

```
B:   SELECT * FROM V_NHANVIEN
                WHERE CONGVIEC = 'Lap trinh vien';
```

Câu truy vấn sau bước view composition:

```
SELECT * FROM NHANVIEN
                WHERE LUONG < 20000 AND
                 CONGVIEC = 'Lap trinh vien';
```

- Query optimization

Nhận xét

- Vì việc kiểm tra quyền được thực hiện trước bước view composition nên quyền được kiểm tra sẽ dựa trên view chứ không dựa trên các bảng cơ sở dùng để định nghĩa view.
- View hữu ích khi cấp quyền trên các cột – chỉ cần tạo view gồm các cột mà ta muốn cấp quyền.
- View còn hữu ích trong việc cấp quyền trên dữ liệu thống kê (dữ liệu sinh ra từ các hàm AVG, SUM, …)
- Chủ thể truy cập có thể cấp quyền truy xuất hay thu hồi trên view tương tự như trên bảng dữ liệu.
- Người dùng muốn tạo View thì phải có quyền Select trên bảng dữ liệu.
- Nếu người tạo View bị thu hồi quyền (hay cấp quyền) trên bảng thì cũng bị thu hồi quyền(hay cấp quyền) trên View, và thu hồi những người dùng khác được người này cấp quyền.

View và cấp quyền dựa trên nội dung

- Người tạo view: view definer.
- Quyền mà view definer có trên view phụ thuộc vào:
  - Ngữ nghĩa của view hay các quan hệ cơ sở dùng để tạo nên view.
  - Quyền mà view definer có trên các bảng cơ sở.
- Quyền alter và index không thể áp dụng cho view, nên view definer không bao giờ có quyền này trên view.

A: CREATE VIEW V1 (MANV, TONGTIEN)
AS SELECT MANV, LUONG+THUONG
FROM NHANVIEN WHERE CONGVIEC = 'Lap trinh vien'

Thao tác update không được định nghĩa trên trường TONGTIEN của view nên A sẽ không thể có quyền update trên field này.

Phân quyền trên view

- Để xác định quyền mà view definer có trên view, hệ thống phải:
  - Tìm giao tập quyền mà view definer có trên các quan hệ cơ sở với tập quyền ứng với các thao tác có thể thực hiện trên view.

Quyền trên view – ví dụ

- Xét quan hệ NHANVIEN và giả sử A là người tạo nên quan hệ NHANVIEN

  A: GRANT Select, Insert, Update ON NHANVIEN to D;

  D: CREATE VIEW V1 AS SELECT MANV, LUONG FROM NHANVIEN;

  D: CREATE VIEW V2 (MANV, LUONG_NAM) AS SELECT MANV, LUONG*12 FROM NHANVIEN;
- D có thể thực hiện tất cả các quyền trên V1 như là các quyền mà D có trên quan hệ NHANVIEN, đó là Select, Insert, Update.
- Tuy nhiên, D chỉ có thể thực hiện trên V2 quyền Select và Update trên cột MANV.

Quyền trên view – ví dụ

- Hoàn toàn có thể cấp quyền trên view.
  - Quyền mà user có thể cấp là những quyền mà user có with grant option trên các quan hệ cơ sở.
  - Ví dụ: user D không thể cấp bất cứ quyền gì trên view V1 và view V2 mà D đã định nghĩa vì D không được chỉ định with grant option khi D được cấp quyền.

Quyền trên view – ví dụ

- Xét các câu lệnh sau:
  - A: GRANT Select ON NHANVIEN TO D WITH GRANT OPTION;
  - A: GRANT Update, Insert ON NHANVIEN TO D;
  - D: CREATE VIEW V4 AS SELECT MANV, LUONG FROM NHANVIEN;

Quyền của D trên V4 sẽ là:

- Select with Grant Option;
- Update, Insert without Grant Option;

DAC – Quyền khẳng định & Phủ định

- System R và hầu hết các HQT dùng chính sách đóng.
  - Với chính sách đóng, thiếu quyền truy xuất đồng nghĩa với việc không có quyền truy xuất.
- Khi chủ thể truy xuất đến 1 đối tượng dữ liệu, HT kiểm tra trong danh sách quyền mà chủ thể được truy xuất, nếu không có thì truy xuất bị từ chối.
  - Hạn chế: Việc thiếu quyền truy xuất không ngăn cấm chủ thể sẽ nhận quyền này từ chủ thể khác.
  - Ví dụ: ***x*** không được quyền truy xuất trên đối tượng ***o***, nhưng trong trường hợp hệ thống sử dụng chính sách phân chia quyền quản trị thì chủ thể có quyền cấp quyền truy xuất trên ***o*** vô tình cấp quyền cho ***x***.
- Người ta đã đưa ra *quyền phủ định* để giải quyết vấn đề ràng buộc này.

DAC – Quyền khẳng định & Phủ định

- Quyền khẳng định: danh sách quyền truy xuất được sử dụng.
- Quyền phủ định: danh sách quyền truy xuất không được sử dụng.
- Tuy nhiên sử dụng Quyền khẳng định và Quyền phủ định thì gây nên xung đột.

Ví dụ: A có quyền WRITE trên bảng NHANVIEN.

A không được READ trên PHONGBAN.

A không được WRITE trên thuộc tính LUONG của NHANVIEN.

- Thường người ta giải quyết xung đột bằng cách ưu tiên quyền phủ định.

DAC – Quyền khẳng định & Phủ định

- Quyền phủ định được thực hiện như là chặn quyền.
- Khi chủ thể bị gán quyền phủ định trên đối tượng thì quyền khẳng định trên đối tượng mà họ có trước đó bị chặn lại.
- Nếu sau này chủ thể được rút quyền phủ định thì họ có thể sử dụng lại quyền khẳng định của mình trước đó.
  - Ưu điểm:
    - Thứ nhất nếu vô tình gán quyền phủ định cho người dùng thì có thể thu hồi lại.
    - Thứ hai là có thể chặn quyền truy cập của chủ thể trong một thời gian bằng cách gán quyền phủ định và sau đó thu hồi lại.

Câu lệnh DENY

- DENY {ALL [PRIVILEGES] | *permission*[,…*n*]}
  { [(*column*[,…*n*])] ON { *table* | *view*} |
  ON {*table* | *view*} [(*column*[,…*n*])] |
  {*procedure* | *extended_procedure*} }
  TO *security_account*

Ví dụ:

DENY SELECT, INSERT, UPDATE

ON NHANVIEN

TO A, B

Thu hồi quyền đã cấp hoặc quyền đã cấm
(Revoking Granted and Denied Permissions)

- **Dùng câu lệnh REVOKE:**
  - **Ta có thể thu hồi lại quyền khẳng định (quyền đã cấp dùng lệnh GRANT)**
  - **Ta có thể thu hồi lại quyền phủ định (quyền đã cấm dùng câu lệnh DENY)**
- Câu lệnh REVOKE giống lệnh DENY ở chỗ không cho thực hiện điều gì đó.
- Câu lệnh REVOKE khác lệnh DENY ở chỗ REVOKE sẽ thu lại quyền đã cấp, còn DENY sẽ cấm một chủ thể (hoặc 1 vai trò) thực hiện một quyền gì đó trong tương lai.

DAC - Ràng buộc ngữ cảnh

- Thực tế, người dùng chỉ được phép truy cập dữ liệu trong 1 khoảng thời gian nhất định .
- Cần phải có một cơ chế hỗ trợ việc truy xuất trong khoảng thời gian cho trước.
  - Ví dụ: cơ chế chỉ cho phép những người làm việc bán thời gian chỉ được phép truy cập dữ liệu vào khoảng từ 9am đến 1pm từ ngày 1/1/98.
- Trong hầu hết các hệ quản trị cơ sở dữ liệu chính sách này thường triển khai ở chương trình ứng dụng.
  - Hạn chế: khi xác nhận và thay đổi chính sách điều khiển truy cập, không bảo đảm rằng chính sách này thực thi.
- Mô hình điều khiển truy cập dựa vào thời gian được đề xuất để giải quyết vấn đề này.

DAC - Ràng buộc ngữ cảnh

- Thời gian hiệu lực:
  - Mỗi quyền truy xuất đều có khoảng thời gian hiệu lực
  - Khi hết thời gian hiệu lực thì quyền truy xuất tự động bị thu hồi mà không cần người quản trị thu hồi.
- Chu kỳ sử dụng quyền truy xuất:
  - Quyền truy xuất theo chu kỳ có thể là quyền khẳng định hay quyền phủ định. Nếu trong cùng một khoảng thời gian mà người dùng vừa có quyền khẳng định vừa có quyền phủ định trên cùng một đối tượng và cùng phương thức truy cập thì ưu tiên cho quyền phủ định.
- Cơ chế suy diễn dựa vào quy tắc suy diễn
  - Quy tắc suy diễn biểu thị ràng buộc của các quyền truy xuất theo thời gian.
  - Quy tắc cho phép suy ra quyền truy xuất mới dựa sự tồn tại hay không tồn tại của quyền truy xuất khác trong khoảng thời gian xác định.
  - Bằng cách sử dụng quy tắc suy diễn đã đáp ứng được yêu cầu bảo vệ dữ liệu một cách ngắn gọn và rõ ràng.
  - Ví dụ: Nếu hai người dùng làm chung một dự án thì phải cùng quyền truy xuất trên các đối tượng.

DAC - Ràng buộc ngữ cảnh

- Quyền truy xuất được định nghĩa là một bộ gồm 5 thuộc tính auth = (s, o, m, pn, g).

Trong đó:

  - **s** (chủ thể), **g** (người gán) U (danh sách người dùng).
  - **m** ∈ M (phương thức truy cập).
  - **o** ∈ O(đối tượng).
  - **pn** ∈ {+ , -}(quyền khẳng định, quyền phủ định).

- *Ví dụ:*

(B,o1, read, +, C) : C gán quyền cho B có thể read đối tượng o1.

(B, o1, write, -, C) : C không cho phép B được quyền write trên đối tượng o1.

DAC - Ràng buộc ngữ cảnh

- Quyền truy xuất theo chu kỳ là bộ ba ([begin,end], P, auth).

Trong đó:

- **begin** là ngày bắt đầu.
- **end** là ngày kết thúc và lớn hơn ngày bắt đầu hay có thể là .
- **P** là biểu thức chu kỳ.
- **auth** là quyền truy xuất.

- Quyền truy xuất theo chu kỳ thể hiện quyền truy xuất có hiệu lực trong chu kỳ P với ngày sử dụng quyền lớn hơn hay bằng t_b (ngày bắt đầu) và nhỏ hơn hay bằng t_e (ngày kết thúc).

Ví dụ 2: A1= ([1/1/94,], Mondays, (A, o1, read, +, B)) quyền truy xuất này được B gán, thể hiện A có quyền read trên đối tượng o1 vào các ngày thứ hai bắt đầu từ ngày 1/1/94.

DAC - Ràng buộc ngữ cảnh

- Khi sử dụng quyền phủ định thì có thể dẫn đến hiện tượng xung đột .
- Ví dụ:
  - Giả sử rằng có thêm quyền truy xuất A2 = ([1/1/95,], Working-days, (A, o1, read, -, B)) được gán cùng với quyền truy xuất A1= ([1/1/94,], Mondays, (A, o1, read, +, B)).
  - Lúc này bắt đầu từ ngày 1/1/95 thì A vừa có quyền phủ định vừa có quyền khẳng định trên cùng đối tượng o1 và cùng phương thức read.
  - Hiện tượng xung đột được giải quyết bằng cách ưu tiên quyền phủ định.
  - Do đó trong khoảng thời gian [1/1/94, 12/31/94] thì A vẫn có quyền read trên đối tượng o1 vào ngày thứ hai, tuy nhiên từ ngày 1/1/95 thì A không được quyền read trên đối tượng này vào các ngày làm việc kể cả thứ hai.

DAC - Ràng buộc ngữ cảnh

- Quy tắc suy diễn được định nghĩa là bộ ba ([begin, end], P, A <OP> $\mathscr{A}$)

Trong đó:

- **begin** là ngày bắt đầu.
- **end** là ngày kết thúc và lớn hơn ngày bắt đầu hay có thể là .
- **P** là biểu thức chu kỳ,
- **A** là quyền truy xuất.
- $\mathscr{A}$ là biểu thức Bool của các quyền truy xuất.
- **OP** là một trong các toán tử WHENEVER, ASLONGAS, UPON.

- Ngữ nghĩa của từng toán tử trong quy tắc suy diễn:

([begin, end], P, A WHENEVER $\mathscr{A}$) : quyền truy xuất A có hiệu lực vào thời điểm t $\in$ chu kỳ P và t $\in$ [t_b, t_e] khi $\mathscr{A}$ có hiệu lực.

- Ví dụ:

A1 = ([1/1/95,1/1/96], Working-days, (M, o1, read, B))

R1= ([1/1/95,], Summer-time, (S, o1, read, +, Bob) WHENEVER (M, o1, read, +, B)).

→S chỉ được read trên đối tượng o1 vào thời điểm mùa hè, từ ngày 1/1/95 khi M được read trên đối tượng này.

DAC - Nhận xét

- Ưu điểm:
  - DAC linh động trong chính sách nên được hầu hết các HQT CSDL ứng dụng.
- Khuyết điểm:
  - Không thể điều khiển được dòng thông tin (information flow control) để có thể chống lại tấn công dạng Trojan Horse.

- RBAC (Role based Access Control)
  - Hầu hết các HQT CSDL đều hỗ trợ RBAC.
  - RBAC có thể dùng kết hợp với DAC hoặc MAC hoặc được dùng độc lập.
  - Đa số các HQT CSDL chỉ hỗ trợ RBAC đơn giản (flat RBAC).

Vai trò và nhóm

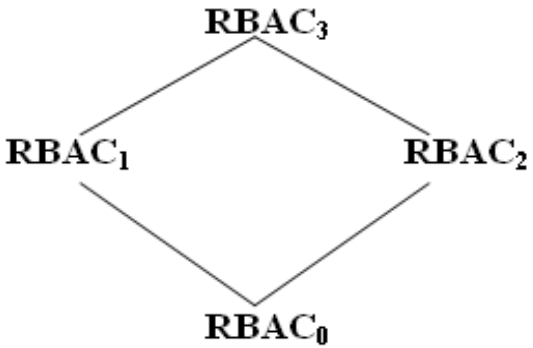
- Mức cơ bản, vai trò có thể được xem tương đương như nhóm.
  - Một đặc quyền có thể được gán cho một hay nhiều nhóm hoặc một hay nhiều vai trò, và một nhóm hay vai trò thì được kết hợp với một hay nhiều đặc quyền.
  - Việc gán một người dùng cho một nhóm hay một vai trò cho phép người dùng thực thi những đặc quyền của nhóm hay vai trò đó.
  - Điểm khác nhau chính giữa nhóm và vai trò đó là nhóm thì coi như đặc trưng một tập hợp người dùng và không là tập hợp quyền hạn. Một vai trò thì một mặt là tập hợp người dùng và một mặt là tập hợp quyền hạn. Vai trò là đối tượng trung gian để mang hai tập hợp này lại với nhau.

RBAC

- Được áp dụng vào đầu những năm 1970s.
- Khái niệm chính của RBAC là những quyền hạn được liên kết với những vai trò.
- Khi số lượng chủ thể và đối tượng lớn → số lượng quyền hạn có thể trở nên vô cùng lớn.
- Nếu người dùng có nhu cầu cao, số lượng cấp và thu hồi quyền diễn ra thường xuyên.
- Với RBAC thì có thể giới hạn trước các mối quan hệ vai trò – quyền hạn, làm cho việc phân công người dùng đến các vai trò được xác định trước dễ dàng hơn.
- Không có RBAC sẽ khó khăn cho việc xác định quyền hạn nào được quy định đến người dùng nào.
- Những người dùng được chỉ định những vai trò thích hợp. Điều này làm đơn giản cho việc quản lí quyền hạn.
- Trong một tổ chức, những chức năng công việc khác nhau được phân thành những vai trò và người dùng được chỉ định vai trò dựa vào trách nhiệm và năng lực của họ.

RBAC – Các mô hình

- Mô hình RBAC gồm 4 mô hình: RBAC0 , RBAC1 , RBAC2 , RBAC3.
  - RBAC0 là nền tảng.
  - RBAC1 thêm vào khái niệm kế thừa của hệ thống phân cấp vai trò. Vai trò có thể kế thừa quyền hạn từ vai trò khác.
  - RBAC2 thêm vào các ràng buộc.
  - RBAC3 là tổng hợp của 3 mô hình.

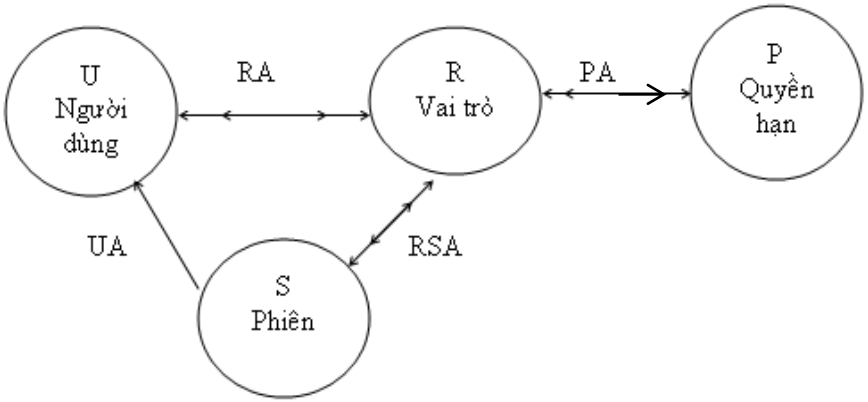


(a) Mối quan hệ giữa các mô hình RBAC

RBAC – Các mô hình

- Mô hình nền tảng *RBAC0* thì ở dưới cùng, nó là yêu cầu tối thiểu cho bất kỳ hệ thống nào có hỗ trợ RBAC.
- Mô hình RBAC1 , RBAC2 được phát triển từ mô hình RBAC0 nhưng có thêm các điểm đặc trưng cho từng mô hình.
  - *RBAC1* thêm vào khái niệm của hệ thống phân cấp vai trò (các trạng thái trong đó vai trò có thể thừa kế quyền hạn từ vai trò khác).
  - *RBAC2* thêm vào các ràng buộc (áp dụng ràng buộc để có thể thừa nhận cấu hình của các thành phần khác nhau của RBAC). RBAC1 , RBAC2 không liên quan nhau.
- RBAC3 là mô hình tổng hợp của ba mô hình RBAC0 , RBAC1 và RBAC2.

RBAC0



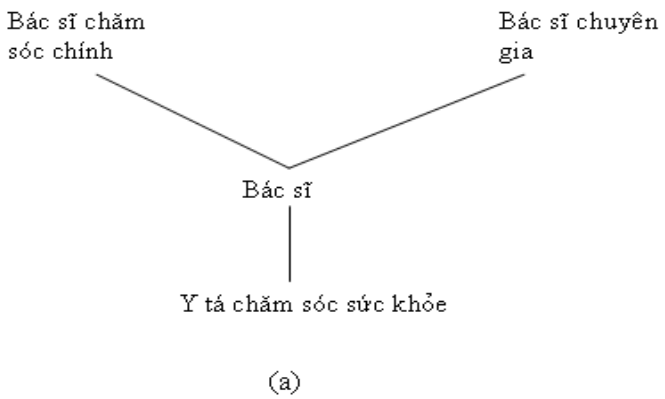
(b) Mô hình $RBAC_0$

- **Người dùng (U)**
- **Vai trò (R)**
- **Quyền hạn (P)**
- **Phiên làm việc (S)**

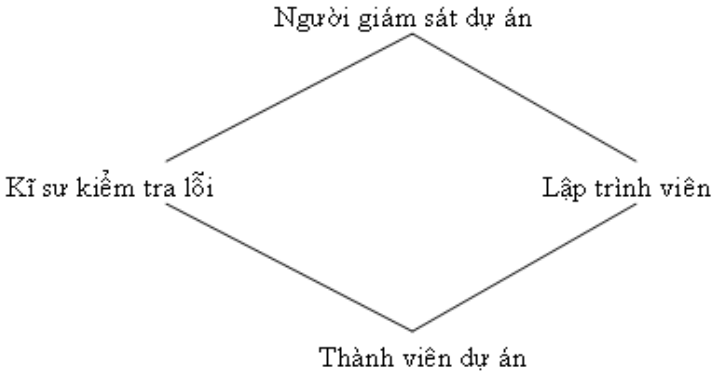
- **Mô hình này cho thấy mối quan hệ giữa người dùng và vai trò là quan hệ nhiều-nhiều. Một người dùng có thể là một thành viên của nhiều vai trò, và một vai trò có thể có nhiều người dùng.**
- **Tương tự, một vai trò có thể có nhiều quyền hạn và cùng một quyền hạn có thể được gán cho nhiều vai trò.**
- **Đặc điểm chính của RBAC nằm trong hai quan hệ này.**
- **Một người dùng sử dụng những quyền hạn. Như vậy vai trò như một đối tượng trung gian.**
- **Mỗi *phiên* là một phép ánh xạ của một người dùng đến nhiều vai trò. Một người dùng thiết lập một phiên trong suốt quá trình người dùng đó kích hoạt vài tập hợp con của các vai trò mà họ là một thành viên.**

RBAC1

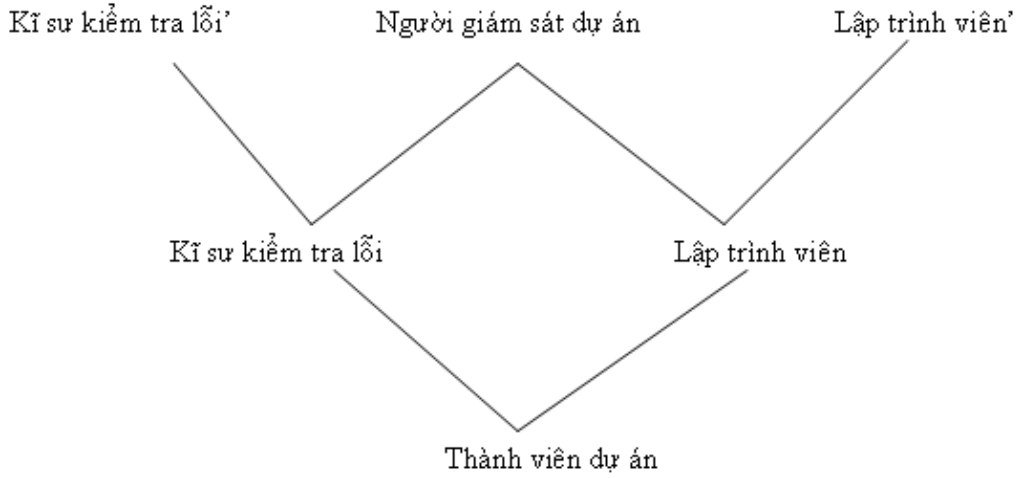
- Quan tâm đến mối quan hệ kế thừa quyền hạn giữa các role.
- Có ý nghĩa về mặt quản lý quyền trong một hệ thống lớn.



(a)



(b)



(c)

RBAC2

- Ràng buộc về sự tách biệt nhiệm vụ (Separation of duties - SoD)
  - Hoạt động quan trọng được chia ra cho hơn hai người nắm giữ nhằm tránh vi phạm đến tính bảo mật dữ liệu.
  - SoD nhằm thực thi chính sách xung đột về lợi ích. Việc xung đột về lợi ích trong RBAC là do người dùng có nhiều quyền hạn liên quan đến các vai trò xung đột nhau.
  - SoD tĩnh thi hành các ràng buộc khi người dùng được gán cho một vai trò.
    - SoD tĩnh: Khi người dùng đã được gán cho một trong các vai trò xung đột nhau thì họ không được gán cho các vai trò còn lại.
    - SoD tĩnh khi có quan hệ phân cấp vai trò: giống như SoD tĩnh nhưng xét thêm ràng buộc về vai trò thừa kế và vai trò được gán trực tiếp.
  - SoD động thì người dùng có thể được gán cho các vai trò xung đột nhưng giới hạn truy cập sẽ được áp đặt khi người dùng truy cập vào hệ thống.

RBAC2 (tt)

- SoD tĩnh và SoD động:
  - Giống nhau là đều giới hạn quyền hạn của người dùng.
  - Khác nhau ở chỗ ngữ cảnh áp dụng ràng buộc. SoD động giới hạn quyền hạn của người dùng bằng cách đặt các ràng buộc trên vai trò và các ràng buộc này bắt đầu kích hoạt trong các phiên của người dùng.
- Khi một vai trò thừa kế từ một vai trò khác thì phải đảm bảo cấu trúc thừa kế không gây nên xung đột các ràng buộc SoD.
- Quy tắc SoD và phân cấp vai trò:
  - *Tính chất 1*: hai vai trò Ri và Rj loại trừ lẫn nhau nếu không có vai trò này thừa kế từ vai trò kia một cách trực tiếp hay gián tiếp.
  - *Tính chất 2*: nếu 2 vai trò Ri và Rj loại trừ lẫn nhau thì không có vai trò thứ ba thừa kế từ hai vai trò này.
  - *Tính chất 3*: nếu đã có quy tắc SoD tĩnh thì không cần dùng quy tắc SoD động. Vì người dùng khi chỉ được gán trên một trong hai vai trò thì họ không thể truy cập cùng lúc hai vai trò.
  - *Tính chất 4*: nếu có bất kỳ hai vai trò Ri và Rj loại trừ lẫn nhau thì sẽ không có vai trò "gốc" hoạt động trong hệ thống. Điều này là do không có vai trò thừa kế từ hai vai trò loại trừ.

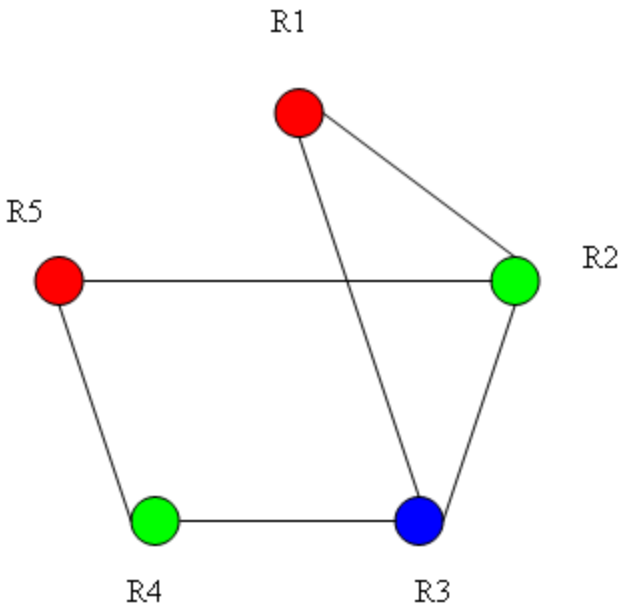
RBAC2 (tt)

- Gán quyền hạn cho các vai trò SoD rất phức tạp. Người quản trị phải đảm bảo không có vai trò nào có tất cả quyền hạn. Và các vai trò được gán cho cá nhân sao cho không có cá nhân nào được tất cả các quyền hạn nhờ vào kết hợp các vai trò.
- Input: n vai trò và mối quan hệ loại trừ nhau.
- Output: Số người tối thiểu để khi gán n vai trò không vi phạm SoD.

RBAC2

	R1	R2	R3	R4	R5
R1	–	X	X	–	–
R2	X	–	X	–	X
R3	X	X	–	X	–
R4	–	–	X	–	X
R5	–	X	–	X	–

R1 và R2 loại trừ nhau



Cần 3 màu để cho không có cạnh trong đồ thị nối 2 đỉnh cùng màu. Nghĩa là để cho việc gán 5 vai trò không vi phạm SoD thì phải gán chúng cho 3 người khác nhau.

RBAC2

- Ràng buộc thời gian:
  - Ràng buộc thời gian trên vai trò.
    - Thời gian hoạt động và bị vô hiệu hóa.
  - Ràng buộc thời gian trên quan hệ người dùng-vai trò.
    - Ràng buộc này có thể dựa vào khoảng thời gian mà một người dùng được gán một vai trò.
  - Ràng buộc thời gian trên quan hệ vai trò-quyền hạn.
    - Ràng buộc này có thể dựa vào khoảng thời gian hoặc thời hạn mà một quyền hạn được gán cho một vai trò.

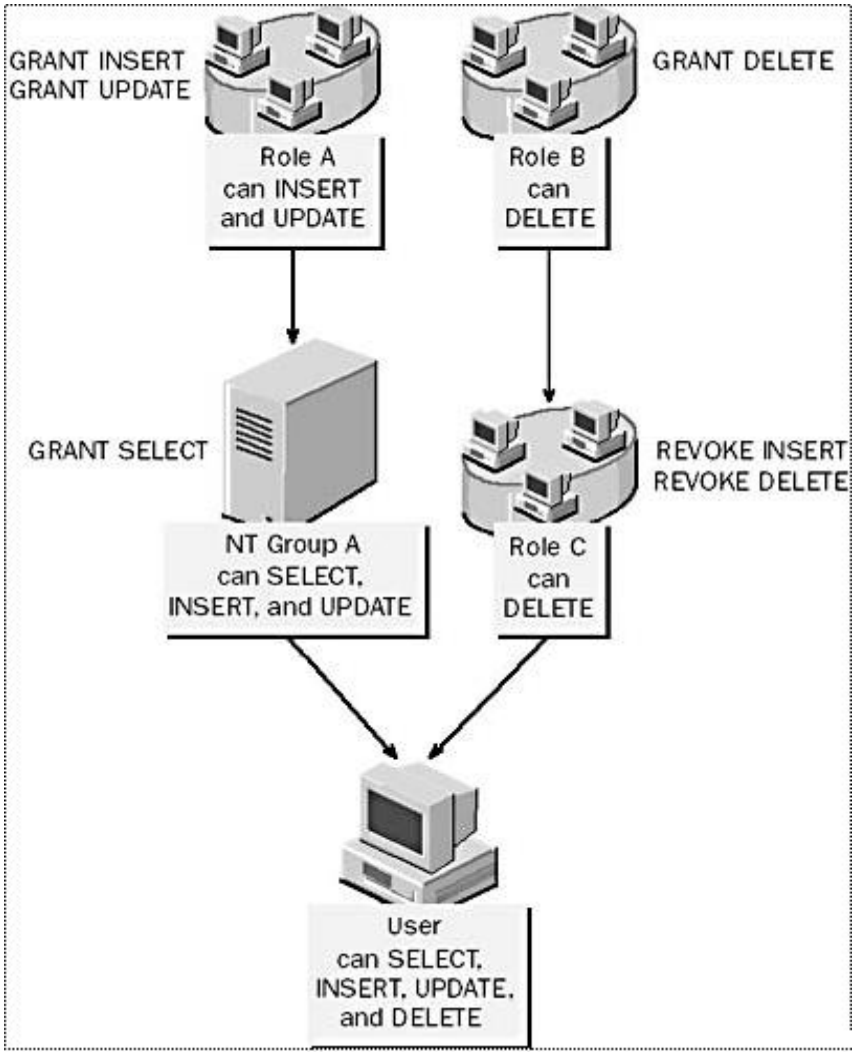
RBAC2

- Nhận xét: Mô hình RBAC2 đã giải quyết được các ràng buộc giữa các vai trò, đảm bảo các vai trò được cấp sẽ đúng đắn và thống nhất, làm tăng thêm sự bảo mật nếu biết vai trò nào loại trừ nhau và không cấp cho cùng người dùng nếu sử dụng SoD tĩnh hoặc cấp nhưng không cho kích hoạt cùng lúc nếu sử dụng SoD động.

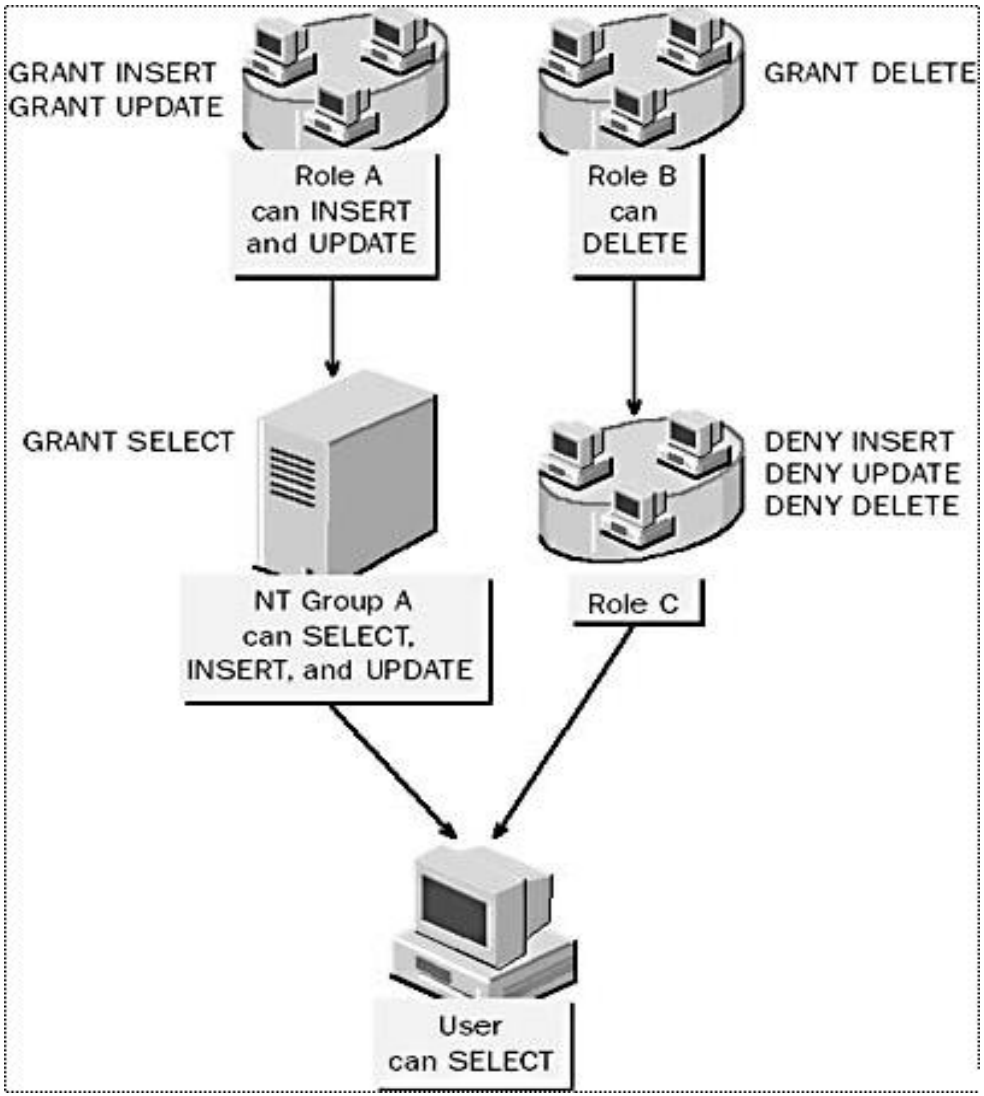
RBAC3

- Đây là mô hình tổng quát nhất và đầy đủ nhất, đảm bảo tốt việc kế thừa vai trò và ràng buộc giữa các vai trò.

Ví dụ


GRANT INSERT
GRANT UPDATE
Role A
can INSERT
and UPDATE
GRANT DELETE
Role B
can
DELETE
GRANT SELECT
NT Group A
can SELECT,
INSERT, and UPDATE
REVOKE INSERT
REVOKE DELETE
Role C
can
DELETE
User
can SELECT,
INSERT, UPDATE,
and DELETE

Ví dụ


GRANT INSERT
GRANT UPDATE
Role A
can INSERT
and UPDATE
Role B
can
DELETE
GRANT DELETE
GRANT SELECT
NT Group A
can SELECT,
INSERT, and UPDATE
Role C
DENY INSERT
DENY UPDATE
DENY DELETE
User
can SELECT

Ví dụ

Account	Permission assigned	Result
Role A	GRANT SELECT	Members of role A have SELECT permission
Role B, member of role A	GRANT INSERT	Members of role B have SELECT permissions (because role B is a member of role A) and INSERT permission
User A, member of role B	DENY INSERT	User A has SELECT permission because it is a member of role A. User A does not have INSERT permission because INSERT has been denied to this user
Role A	DENY SELECT	Members of role A do not have SELECT permission

Account	Permission assigned	Result
Role B, member of role A	GRANT SELECT	Members of role B do not have SELECT permission because role B is a member of role A, which denies the SELECT permission
User A, member of role B	GRANT INSERT	User A has INSERT permission only
Role A	GRANT SELECT	Members of role A have SELECT permission
Role B, member of role A	REVOKE SELECT	Members of role B have SELECT permission because they still get it from role A
User A, member of role B	GRANT INSERT	User A has SELECT permissions (because the user is a member of role B) and INSERT permissions

MAC

- MAC (Mandatory Access Control)

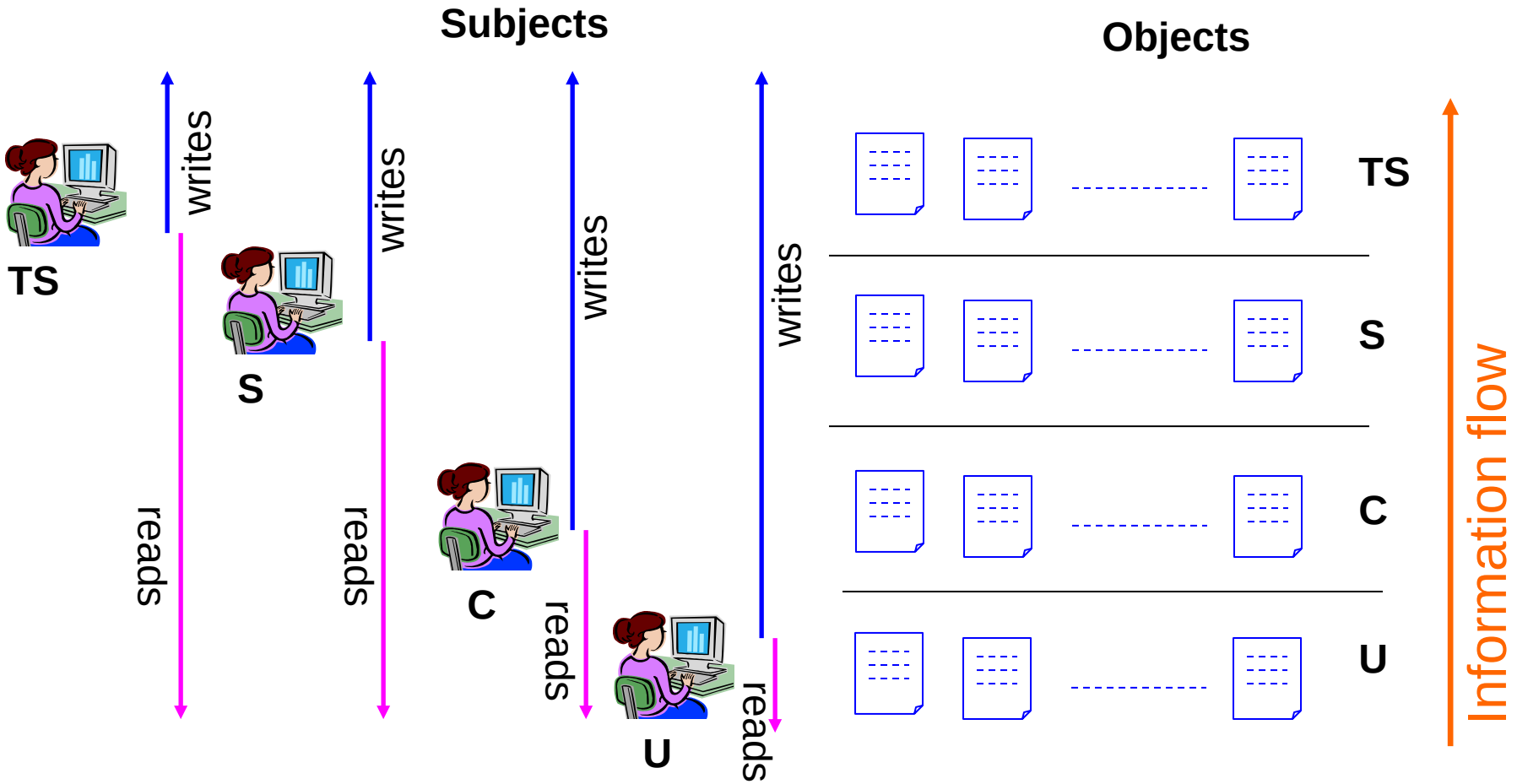
MAC

- Điều khiển truy xuất dựa trên sự phân lớp chủ thể truy xuất và đối tượng dữ liệu.
- MAC được dùng trong môi trường cần tính chất an ninh cao: như chính phủ, quân đội, …
- MAC được ORACLE cài đặt.

MAC

- Đối tượng dữ liệu (Object): tables, views, tuples.
- Chủ thể truy cập (Subject): users, user programs.
- Security class (or level, or labels)
  - Top Secret (TS), Secret (S), Confidential (C), Unclassified (U), where TS > S > C > U
- Mỗi chủ thể và mỗi đối tượng được xếp vào một class.
  - No read – up: Chủ thể S có thể Đọc đối tượng O nếu Class(S) >= Class(O).
  - No write – down: Chủ thể S có thể Ghi đối tượng O nếu class(S) <= Class (O).
    - Tuy nhiên, thực tế các hệ thống không cho phép write up, mà chỉ cho phép write cùng mức. Hãy kiểm tra điều này trên Oracle?

MAC


Subjects
Objects
TS
S
C
U
writes
reads
TS
S
C
U
Information flow

MAC – Nhận xét

- Nguyên lý về đơn vị dữ liệu của đối tượng bảo mật.
  - Là toàn bộ CSDL, hay tập tin, ở các thuộc tính hay từng item.
- Không có kỹ thuật tự động cho việc gán nhãn bảo mật.
- Nhiều người cùng truy cập tại một thời điểm.
  - Vì áp dụng chính sách dòng thông tin nên những người có mức bảo mật cao hơn bị hạn chế ghi xuống các hạng mục dữ liệu có sự phân loại thấp hơn. Ví dụ có chủ thể s1 và s2 có nhãn(s1)>nhãn(s2), mục dữ liệu d với nhãn(d) = nhãn(s2), và luật thương mại cho rằng để ghi dữ liệu lên d của s2 cần sự chấp thuận của s1. Điều này thì không thích hợp cho các ứng dụng thương mại của công nghệ CSDL MLS.

MAC

- MAC còn được gọi là điều khiển truy cập đa cấp (Multilevel security – MLS), ứng dụng trên CSDL quan hệ, có CSDL quan hệ đa cấp (Multilevel Relational Model - MLR).
- Hệ thống quản lý dữ liệu đáp ứng các thuộc tính của việc bảo mật đa cấp được thiết kế dựa trên mô hình nền tảng là Bell và LaPadula.

- Hết chương 3.