CHƯƠNG 4.
PHỤ THUỘC HÀM

MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM

- ***Chuẩn hóa*** là một kỹ thuật tạo ra một tập các quan hệ với các thuộc tính từ các yêu cầu cho trước về dữ liệu cần mô hình hóa của tổ chức.
- Quá trình chuẩn hóa đầu tiên được phát triển bởi Codd vào năm 1972.
- Việc chuẩn hóa thường được thực hiện như một chuỗi các kiểm tra trên một quan hệ để xác định xem nó có thỏa mãn hay vi phạm các yêu cầu của một dạng chuẩn cho trước nào đó hay không.

MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- Codd định nghĩa 3 dạng chuẩn: dạng chuẩn 1 (1NF), dạng chuẩn 2 (2NF), và dạng chuẩn 3 (3NF).
- Năm 1974, Boyce và Codd cùng giới thiệu một dạng chuẩn mạnh hơn 3NF được gọi là chuẩn Boyce-Codd (BCNF).
- Cả 4 dạng chuẩn đều dựa trên *sự phụ thuộc hàm giữa các thuộc tính trong một quan hệ*.
- ***Một phụ thuộc hàm*** mô tả mối quan hệ giữa các thuộc tính trong một quan hệ.

  Ví dụ: A, B là các thuộc tính hoặc tập các thuộc tính trong quan hệ R. B phụ thuộc hàm vào A (ký hiệu: A->B) nếu mỗi giá trị của A liên hệ tới duy nhất một giá trị của B.

MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- Năm 1977, 1979 và các năm tiếp theo, người ta giới thiệu dạng chuẩn 4 (4NF), dạng chuẩn 5 (5NF) và các dạng chuẩn mức cao hơn.

  => Tuy nhiên, rất hiếm khi gặp.

- ***Quá trình chuẩn hóa*** là một phương pháp chính thống để xác định các quan hệ dựa trên khóa chính, khóa dự bị và các phụ thuộc hàm giữa các thuộc tính.

- ***Chuẩn hóa*** giúp người thiết kế kiểm tra và chuyển các quan hệ về một dạng chuẩn hóa cụ thể nào đó nhằm ngăn chặn các dị thường khi thực hiện các thao tác cập nhật thông tin.

MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- Mục đích của việc thiết kế CSDL quan hệ: nhóm các thuộc tính vào thành các quan hệ sao cho tối thiểu hóa sự dư thừa dữ liệu => giảm không gian lưu trữ và tránh dị thường thông tin khi cập nhật dữ liệu.
- Xét ví dụ lược đồ quan hệ *staffbranch*.

staffbranch

staff#	**sname**	**position**	**salary**	**branch#**	**baddress**
SL21	Kristy	manager	30000	B005	22 Deer Road
SG37	Debi	assistant	12000	B003	163 Main Street
SG14	Alan	supervisor	18000	B003	163 Main Street
SA9	Traci	assistant	12000	B007	375 Fox Avenue
SG5	David	manager	24000	B003	163 Main Street
SL41	Anna	assistant	10000	B005	22 Deer Road

MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- Quan hệ *staffbranch* có dư thừa dữ liệu. Thông tin chi tiết của một chi nhánh ngân hàng (*branch*) sẽ bị lặp lại cho mỗi nhân viên (*staff*) làm việc tại chi nhánh đó.
- Nếu tách riêng quan hệ *staff* và *branch* thì thông tin về từng chi nhánh ngân hàng chỉ xuất hiện duy nhất một lần.

staff

staff#	sname	position	salary	branch#
SL21	Kristy	manager	30000	B005
SG37	Debi	assistant	12000	B003
SG14	Alan	supervisor	18000	B003
SA9	Traci	assistant	12000	B007
SG5	David	manager	24000	B003
SL41	Anna	assistant	10000	B005

branch

branch#	baddress
B005	22 Deer Road
B003	163 Main Street
B007	375 Fox Avenue

DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT

❖ ***Khi thực hiện chèn thêm dữ liệu:***

- Để chèn thêm thông tin chi tiết cho các nhân viên mới vào *staffbranch*, cần phải thêm thông tin chi tiết về chi nhánh tương ứng cho mỗi bản ghi nhân viên.

  Ví dụ: Nếu thêm nhân viên mới vào *branch* B007, thì phải thêm cả địa chỉ của B007. Còn đối với lược đồ (*staff, branch*) thì chỉ cần thêm thông tin vào quan hệ *staff* là đủ.

- Để chèn thêm thông tin cho một *branch* mới mà hiện tại chưa có nhân viên nào, cần chèn thêm các giá trị *Null* cho các thuộc tính về nhân viên, như *staff#, sname, …*

  ⇒ Tuy nhiên, thuộc tính *staff#* là khóa chính nên điều này không thể được (vi phạm tính toàn vẹn của khóa).

  ⇒ Vậy, không thể nhập thông tin cho một chi nhánh mới mà không có nhân viên nào.

DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT (Cont.)

❖ ***Khi thực hiện xóa dữ liệu:***

- Giả sử trong *staffbranch* có một chi nhánh chỉ còn một nhân viên cuối cùng. Nếu xóa thông tin về nhân viên này thì các thông tin về chi nhánh đó cũng sẽ bị xóa khỏi CSDL.

  Ví dụ: Nếu xóa thông tin về nhân viên Traci khỏi quan hệ *staffbranch* thì thông tin về chi nhánh B007 cũng sẽ bị xóa.

  ⇒ Điều này không xảy ra đối với lược đồ (*staff, branch*) vì thông tin về nhân viên được lưu trữ tách riêng khỏi thông tin về chi nhánh.

DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT (Cont.)

❖ ***Khi thực hiện cập nhật dữ liệu:***

- Giả sử muốn thay đổi giá trị của một trong các thuộc tính của một chi nhánh nào đó trong quan hệ staffbranch, ví dụ: địa chỉ chi nhánh, thì cần phải cập nhật tất cả các bộ của các nhân viên làm việc tại chi nhánh đó.
- Nếu việc cập nhật thay đổi địa chỉ chi nhánh không được thực hiện trên tất cả các bộ liên quan trên quan hệ staffbranch thì CSDL sẽ không nhất quán (một chi nhánh sẽ có 2 địa chỉ khác nhau).

⇒ *Như vậy: trên quan hệ staffbranch, khi thực hiện các thao tác cập nhật dữ liệu thì xuất hiện dị thường thông tin. Việc này sẽ tránh được bằng cách tách lược đồ staffbranch thành 2 quan hệ staff và branch.*

DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT (Cont.)

❖ ***2 tính chất quan trọng khi tách một lược đồ quan hệ thành một tập các lược đồ nhỏ hơn:***

1. ***Kết nối không tổn thất thông tin:*** đảm bảo mọi thể hiện của quan hệ ban đầu có thể xác định được từ các thể hiện liên quan trong các quan hệ nhỏ hơn.
2. ***Bảo toàn sự phụ thuộc hàm:*** đảm bảo một ràng buộc trên quan hệ ban đầu có thể được duy trì bằng cách sử dụng đơn giản một số ràng buộc trên mỗi quan hệ nhỏ hơn.

   => Các quan hệ nhỏ hơn không cần kết nối với nhau để kiểm tra xem một ràng buộc trên các quan hệ ban đầu có bị vi phạm hay không.

KẾT NỐI KHÔNG TỔN THẤT THÔNG TIN

- Xét lược đồ quan hệ SP và sự phân tách nó thành 2 lược đồ S1 và S2.

SP

s#	p#	qty
S1	P1	10
S2	P2	50
S3	P3	10

S1

s#	qty
S1	10
S2	50
S3	10

S2

p#	qty
P1	10
P2	50
P3	10

$S1 * S2$

s#	p#	qty
S1	P1	10
S1	P3	10
S2	P2	50
S3	P1	10
S3	P3	10

Những bộ mới này không xuất hiện trong quan hệ ban đầu. Tuy vậy, không thể nói bộ nào là hợp lệ hay không hợp lệ. Khi sự phân tách xảy ra, quan hệ SP ban đầu có thể bị mất.

BẢO TOÀN CÁC PHỤ THUỘC HÀM

- Xét ví dụ:

  R = (A, B, C)

  F = {AB $\rightarrow$ C, C $\rightarrow$ A}

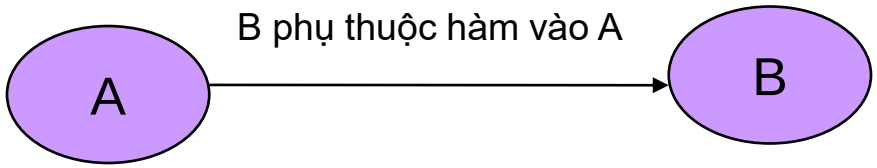
  γ = {(B, C), (A, C)}

  - Rõ ràng, C $\rightarrow$ A là được bảo toàn trong lược đồ quan hệ (A, C).
  - Làm thế nào để AB $\rightarrow$ C mà không cần kết nối 2 quan hệ trong lược đồ quan hệ γ? => Không thể.

    => Vậy, các phụ thuộc hàm không được bảo toàn trong γ.

ĐỊNH NGHĨA PHỤ THUỘC HÀM

- ***Một phụ thuộc hàm*** thể hiện ngữ nghĩa của các thuộc tính trong một quan hệ: một thuộc tính có quan hệ với thuộc tính khác như thế nào và xác định các phụ thuộc hàm giữa các thuộc tính đó. Phụ thuộc hàm là ràng buộc giữa các thuộc tính.
- Xét một quan hệ với các thuộc tính A và B, với thuộc tính B là phụ thuộc hàm vào thuộc tính A: Nếu biết giá trị của A sẽ tìm thấy một giá trị duy nhất của B. Nhưng với mỗi giá trị của B cho trước có thể có nhiều giá trị khác nhau tương ứng của A.



ĐỊNH NGHĨA PHỤ THUỘC HÀM (Cont.)

- ***Đối tượng xác định của một phụ thuộc hàm*** là thuộc tính hoặc một nhóm các thuộc tính nằm bên trái của mũi tên trong một phụ thuộc hàm.
- ***Đối tượng (thuộc tính) hệ quả của phụ thuộc hàm*** là một thuộc tính hoặc nhóm thuộc tính nằm phía bên phải mũi tên trong phụ thuộc hàm.
- Ví dụ: A→B:

  A là đối tượng xác định và B là đối tượng hệ quả của A.

  => Nói: A xác định B hay B phụ thuộc hàm vào A.

XÁC ĐỊNH CÁC PHỤ THUỘC HÀM

- Quay lại ví dụ quan hệ staff (trong phần trước):
  - Thể hiện của quan hệ này có một phụ thuộc hàm staff#→position. Không có phụ thuộc theo chiều ngược lại.
  - Mối quan hệ giữa staff# và position là 1:1, nghĩa là mỗi nhân viên có một vị trí tương ứng duy nhất.
  - Quan hệ giữa position và staff# là 1:nhiều, nghĩa là nhiều nhân viên có cùng một vị trí.
  - Position không xác định staff# hay staff# không phụ thuộc hàm vào position.
- Với mục đích chuẩn hóa: chỉ quan tâm tới việc xác định các phụ thuộc hàm giữa các thuộc tính của quan hệ 1:1.

XÁC ĐỊNH CÁC PHỤ THUỘC HÀM (Cont.)

- Khi xác định các phụ thuộc hàm giữa các thuộc tính trong một quan hệ, cần phân biệt rõ ràng giữa các giá trị nhận được bởi một thuộc tính tại một thời điểm và tập tất cả các giá trị có thể nhận được bởi thuộc tính đó tại các thời điểm khác nhau.

  => Một phụ thuộc hàm là một đặc điểm chung của một lược đồ quan hệ chứ không phải là một đặc điểm riêng của một thể hiện cụ thể của lược đồ đó.

- Cần xác định các phụ thuộc hàm thỏa mãn cho tất cả các giá trị có thể có của các thuộc tính của một quan hệ.

  => thể hiện các loại ràng buộc toàn vẹn cần xác định.

VÍ DỤ XÁC ĐỊNH CÁC PHỤ THUỘC HÀM

- ***Xét quan hệ staffbranch:***

  Để xác định các phụ thuộc hàm, cần hiểu rõ ngữ nghĩa của các thuộc tính khác nhau trong mỗi lược đồ quan hệ.

  Ví dụ: vị trí của mỗi nhân viên và chi nhánh sẽ xác định lương của họ.

=> *Cần hiểu rõ về tổ chức. Đây là nhiệm vụ của giai đoạn phân tích yêu cầu bài toán và giai đoạn thiết kế mức khái niệm.*

VÍ DỤ XÁC ĐỊNH CÁC PHỤ THUỘC HÀM (Cont.)

- ***Các phụ thuộc hàm được xác định như sau:***

  staff# → sname, position, salary, branch#, baddress

  branch# → baddress

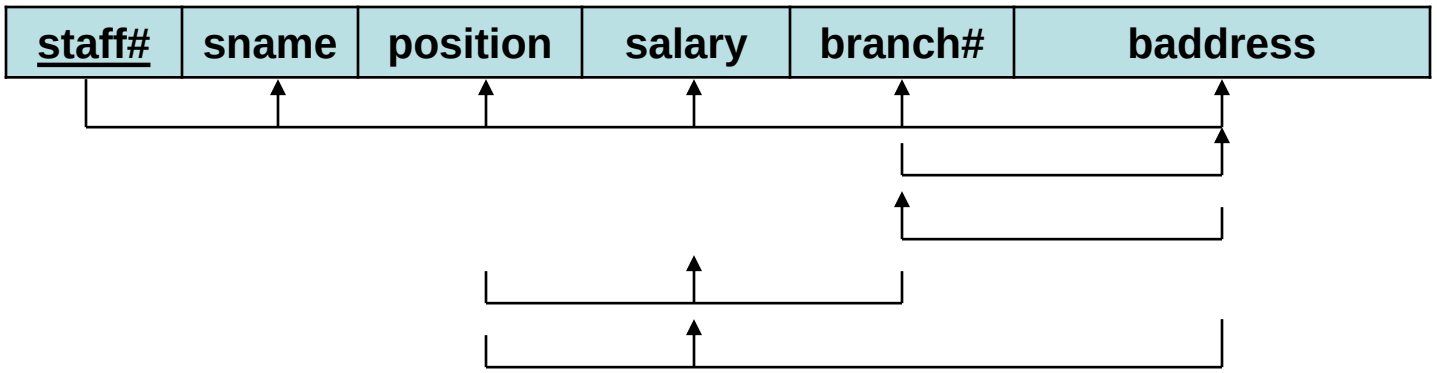
  baddress → branch#

  branch#, position → salary

  baddress, position → salary

- Có thể dùng một dạng ký pháp lược đồ để biểu diễn như sau:

staffbranch



PHỤ THUỘC HÀM HIỂN NHIÊN

- Trong quá trình xác định các phụ thuộc hàm, cần bỏ qua các phụ thuộc hàm hiển nhiên đúng.
- ***Phụ thuộc hàm hiển nhiên đúng:*** khi và chỉ khi vế phải của phụ thuộc hàm là một tập con của vế trái.

  Ví dụ: { staff#, sname} → sname

  { staff#, sname} → staff#
- Các phụ thuộc hàm này không cung cấp thêm các thông tin cần thiết nào về các ràng buộc toàn vẹn đối với quan hệ.

  => Trong phạm vi chuẩn hóa, các phụ thuộc hàm này được bỏ qua.

CÁC ĐẶC TÍNH CỦA PHỤ THUỘC HÀM

Các đặc tính có ích của các phụ thuộc hàm cho việc chuẩn hóa:

- Tồn tại mối quan hệ 1:1 giữa các thuộc tính trong đối tượng xác định và đối tượng hệ quả.
- Phụ thuộc hàm là bất biến theo thời gian, nghĩa là nó thỏa mãn tất cả các thể hiện có thể của quan hệ.
- Các phụ thuộc hàm là không hiển nhiên. Tất cả các phụ thuộc hàm hiển nhiên đúng đều được bỏ qua.

CÁC LUẬT SUY DIỄN CHO CÁC PHỤ THUỘC HÀM (Cont.)

- Một tập các luật suy diễn là cần thiết để suy diễn ra tập các phụ thuộc hàm dựa vào F.
- Sáu luật suy diễn được biết đến nhiều nhất được áp dụng cho các phụ thuộc hàm như sau:

  IR1: Luật phản xạ: nếu $X \supseteq Y$, thì $X \rightarrow Y$

  IR2: Luật tăng trưởng: nếu $X \rightarrow Y$, thì $XZ \rightarrow YZ$

  IR3: Luật bắc cầu: nếu $X \rightarrow Y$ và $Y \rightarrow Z$, thì $X \rightarrow Z$

$\Rightarrow$ *3 luật suy diễn này là Hệ tiên đề Amstrong: đóng vai trò là một tập luật cần thiết và đầy đủ cho việc tạo ra bao đóng của một tập các phụ thuộc hàm.*

CÁC LUẬT SUY DIỄN CHO CÁC PHỤ THUỘC HÀM (Cont.)

IR4: Luật chiếu:

Nếu $X \rightarrow YZ$, thì $X \rightarrow Y$ và $X \rightarrow Z$

IR5: Luật cộng thêm:

Nếu $X \rightarrow Y$ và $X \rightarrow Z$, thì $X \rightarrow YZ$

IR6: Luật giả bắc cầu:

Nếu $X \rightarrow Y$ và $YZ \rightarrow W$, thì $XZ \rightarrow W$

VÍ DỤ

Cho lược đồ quan hệ: R = (A,B,C,D,E,F,G,H, I, J) và

F = {AB $\rightarrow$ E, AG $\rightarrow$ J, BE $\rightarrow$ I, E $\rightarrow$ G, GI $\rightarrow$ H}

a. Hỏi F $\models$ AB $\rightarrow$ GH? b. Hỏi F $\models$ BE $\rightarrow$ H?

Chứng minh:

1. AB $\rightarrow$ E, đã cho trong F
2. AB $\rightarrow$ B, luật phản xạ IR1
3. AB $\rightarrow$ BE, luật cộng thêm IR5 từ bước 1 và 2
4. BE $\rightarrow$ I, đã cho trong F
5. AB $\rightarrow$ I, luật bắc cầu IR3 từ bước 3 và 4
6. E $\rightarrow$ G, đã cho trong F
7. AB $\rightarrow$ G, luật bắc cầu IR3 từ bước 1 và 6
8. AB $\rightarrow$ GI, luật cộng thêm IR5 từ bước 5 và 7
9. GI $\rightarrow$ H, đã cho trong F
10. AB $\rightarrow$ H, luật bắc cầu IR3 từ bước 8 và 9
11. AB $\rightarrow$ GH, luật cộng thêm IR5 từ bước 7 và 10

=> kết quả được chứng minh.

CÁC LUẬT SUY DIỄN CHO CÁC PHỤ THUỘC HÀM

- Gọi F là tập các phụ thuộc hàm xác định trên lược đồ quan hệ R.
- Ngoài các phụ thuộc hàm hiển nhiên, còn có nhiều các phụ thuộc hàm khác cũng thỏa mãn với các thể hiện của quan hệ mà đã thỏa mãn các phụ thuộc hàm trong F.
- ***Tập tất cả các phụ thuộc hàm được suy diễn từ một tập phụ thuộc hàm F được gọi là bao đóng của F và được ký hiệu là F^+.***
  - Ký hiệu: $F \models X \rightarrow Y$ cho biết phụ thuộc hàm $X \rightarrow Y$ được suy diễn từ tập phụ thuộc hàm F.
  - $F^+ \equiv \{X \rightarrow Y \mid F \models X \rightarrow Y\}$

XÁC ĐỊNH BAO ĐÓNG

- F^+ là bao đóng của tập phụ thuộc hàm F. F^+ là tập (nhỏ nhất) tất cả các phụ thuộc hàm được sinh ra nhờ hệ tiên đề Amstrong.
- F^+ là hữu hạn nhưng có kích thước tăng theo cấp số nhân so với số thuộc tính của R.

Ví dụ: Quan hệ R=(A,B,C) và F = {AB $\rightarrow$C, C $\rightarrow$ B}, F^+ sẽ bao gồm 29 phụ thuộc hàm (kể cả các phụ thuộc hàm hiển nhiên).

XÁC ĐỊNH BAO ĐÓNG (Cont.)

- Để xác định liệu một phụ thuộc hàm $X \rightarrow Y$ có thỏa mãn lược đồ quan hệ R với tập phụ thuộc hàm F hay không thì cần xem $F \models X \rightarrow Y$ không, hoặc chính xác hơn là xem $X \rightarrow Y$ có nằm trong F^+ hay không.

$\Rightarrow$ ***Mong muốn*** *kiểm tra được $X \rightarrow Y$ có nằm trong F^+ hay không mà không cần sinh ra tất cả các thành phần của bao đóng.*

$\Rightarrow$ ***Thực hiện bằng cách:*** *chỉ cần sinh ra bao đóng của tập thuộc tính X, ký hiệu là X^+, và kiểm tra xem liệu Y thuộc X^+ hay không.*

THUẬT TOÁN TÍNH BAO ĐÓNG

- X^+ được gọi là bao đóng của tập thuộc tính X trên tập phụ thuộc hàm F nếu mọi thuộc tính trong X^+ đều được sinh ra từ X nhờ F.

```
Thuật toán Closure  {trả về X+ trên F}
Đầu vào:  tập thuộc tính X, và một tập phụ thuộc hàm F
Đầu ra: Tính X+ trên F
Closure (X, F)
{
    X+ ← X;
    repeat
        oldX+ ← X+;
        for mỗi phụ thuộc hàm W→ Z trong F do
            if W ⊆ X+ then X+ ← X+ ∪ Z;
    until (oldX+ = X+);
}
```

VÍ DỤ TÍNH BAO ĐÓNG

Cho F = {A $\rightarrow$ D, AB $\rightarrow$ E, BI $\rightarrow$ E, CD $\rightarrow$ I, E $\rightarrow$ C}.

Tính $(AE)^+$

Vòng 1

X^+ = {A, E}
Xét A $\rightarrow$ D, A $\subseteq$ X^+, nên thêm D vào X^+, X^+ = {A, E, D}
Xét AB $\rightarrow$ E, AB không thuộc X^+
Xét BI $\rightarrow$ E, BI không thuộc X^+
Xét CD $\rightarrow$ I, CD không thuộc X^+
Xét E $\rightarrow$ C, E$\subseteq$ X^+, nên thêm C vào X^+, X^+ = {A, E, D, C}
Có thay đổi xảy ra với X^+ so với ban đầu vì vậy cần thêm một vòng lặp nữa

Vòng 2

X^+ = {A, E, D, C}
Xét A $\rightarrow$ D, A thuộc X^+, nhưng không có thay đổi
Xét AB $\rightarrow$ E, AB không thuộc X^+
Xét BI $\rightarrow$ E, BI không thuộc X^+
Xét CD $\rightarrow$ I, CD $\subseteq$ X^+, nên thêm I vào X^+, X^+ = {A, E, D, C, I}
Xét E $\rightarrow$ C, E thuộc X^+, nhưng không có thay đổi
Có thay đổi xảy ra với X^+ so với cuối vòng 1 vì vậy cần thêm một vòng lặp nữa

VÍ DỤ TÍNH BAO ĐÓNG (Cont.)

Vòng 3

$X^+ = \{A, E, D, C, I\}$
Xét $A \rightarrow D$, A thuộc X^+, nhưng không có thay đổi
Xét $AB \rightarrow E$, AB không thuộc X^+
Xét $BI \rightarrow E$, BI không thuộc X^+
Xét $CD \rightarrow I$, CD thuộc X^+, nhưng không có thay đổi
Xét $E \rightarrow C$, E thuộc X^+, nhưng không có thay đổi
Không có thay đổi nào xảy ra với X^+ so với cuối vòng 2, vì vậy thuật toán dừng.

Vậy: $(AE)^+ = \{A, E, C, D, I\}$

$\Rightarrow$ Điều này có nghĩa là các phụ thuộc hàm sau sẽ thuộc F^+:
$AE \rightarrow AECDI$

THUẬT TOÁN MEMBER

Thuật toán Member {xác định các thành viên trong F^+}

Đầu vào: một tập các phụ thuộc hàm F,
và một phụ thuộc hàm đơn $X \rightarrow Y$

Đầu ra: Trả lại kết quả đúng (true) nếu $F \models X \rightarrow Y$,
ngược lại trả lại kết quả sai (false)

Member (F, $X \rightarrow Y$)
{
    if $Y \subseteq$ Closure(X,F)
        then return true;
        else return false;
}

PHỦ VÀ SỰ TƯƠNG ĐƯƠNG CỦA PHỤ THUỘC HÀM

- Một tập phụ thuộc hàm F được phủ bởi một tập phụ thuộc hàm G (hay nói cách khác G phủ F) nếu mọi phụ thuộc hàm trong F đều nằm trong G^+.

  $\Rightarrow$ F được phủ nếu mọi phụ thuộc hàm trong F có thể được suy diễn từ G.

- Hai tập phụ thuộc hàm F và G là tương đương (ký hiệu: $F \equiv G$) nếu $F^+=G^+$.

  $\Rightarrow$ Mọi phụ thuộc hàm trong G có thể được suy diễn từ F và mọi phụ thuộc hàm trong F có thể được suy diễn từ G.

- Để xác định xem G có phủ F hay không: tính X^+ trên G cho mỗi phụ thuộc hàm $X \rightarrow Y$ trong F, nếu $Y \subseteq X^+$ cho mọi $X \rightarrow Y$ thì G phủ F.

PHỦ VÀ SỰ TƯƠNG ĐƯƠNG CỦA PHỤ THUỘC HÀM (Cont.)

- Số phụ thuộc hàm càng ít đòi hỏi càng ít không gian nhớ.
  => *Chi phí thấp khi thực hiện các thao tác cập nhật.*

- Có nhiều loại phủ, từ không dư thừa đến các phủ tối thiểu.
  => *Không xem xét đến tất cả các loại này.*

- ***Ý tưởng:*** Sinh ra một tập các phụ thuộc hàm G tương đương với tập F ban đầu nhưng lại có số lượng phụ thuộc hàm càng ít càng tốt và càng tối giản càng tốt (phủ tối thiểu).

- ***Thuật toán Member:*** kiểm tra xem một phụ thuộc hàm $X \rightarrow Y$ có thuộc vào tập F hay không. Thời gian chạy thuật toán phụ thuộc vào kích cỡ của tập phụ thuộc hàm => tập phụ thuộc hàm càng nhỏ thì thời gian chạy thuật toán càng nhanh.

PHỦ KHÔNG DƯ THỪA

- Một tập các phụ thuộc hàm F được cho là không dư thừa nếu không có tập con thực sự G nào của F mà G tương đương với F. Nói cách khác, trong F không có phụ thuộc hàm dư thừa.

  Ngược lại, nếu tồn tại thì F được gọi là dư thừa.

- F là một phủ không dư thừa của G nếu F là một phủ của G và F không dư thừa.

- ***Thuật toán Nonredundant:*** sinh ra một phủ không dư thừa.

THUẬT TOÁN NONREDUNDANT

Thuật toán Nonredundant {sinh ra một phủ không dư thừa}
Đầu vào: một tập phụ thuộc hàm G
Đầu ra: một phủ không dư thừa của G

```
Nonredundant (G)
{
    F ← G;
    for mỗi phụ thuộc hàm X → Y ∈ G do
        if Member(F – {X → Y}, X → Y)
            then F ← F – {X → Y};
     return (F);
}
```

VÍ DỤ: TÌM PHỦ KHÔNG DƯ THỪA

Cho G = {A → B, B → A, B → C, A → C}, tìm một phủ không dư thừa của G.

F ← G

Member({B → A, B → C, A → C}, A → B)

Closure(A, {B → A, B → C, A → C})

A^+ = {A, C}, nên A → B là không dư thừa

Member({A → B, B → C, A → C}, B → A)

Closure(B, {A → B, B → C, A → C})

B^+ = {B, C}, nên B → A là không dư thừa

Member({A → B, B → A, A → C}, B → C)

Closure(B, {A → B, B → A, A → C})

B^+ = {B, A, C}, nên B → C là dư thừa, F = F – {B → C}

Member({A → B, B → A}, A → C)

Closure(A, {A → B, B → A})

A^+ = {A, B}, nên A → C là không dư thừa

Vậy F = {A → B, B → A, A → C} là một phủ không dư thừa của G

VÍ DỤ 2: TÌM PHỦ KHÔNG DƯ THỪA

Nếu G = {A → B, A → C, B → A, B → C}, giống tập phụ thuộc hàm trước nhưng khác nhau về thứ tự của các phụ thuộc hàm. Kết quả sẽ sinh ra một phủ không dư thừa khác!

F ← G

Member({A → C, B → A, B → C}, A → B)

Closure(A, {A → C, B → A, B → C})

A^+ = {A, C}, nên A → B không dư thừa

Member({A → B, B → A, B → C}, A → C)

Closure(A, {A → B, B → A, B → C})

A^+ = {A, B, C}, nên A → C là dư thừa, F = F – {A → C}

Member({A → B, B → C}, B → A)

Closure(B, {A → B, B → C})

B^+ = {B, C}, nên B → A là không dư thừa

Member({A → B, B → A}, B → C)

Closure(B, {A → B, B → A})

B^+ = {B, A}, nên B → C là không dư thừa.

Vậy F = {A → B, B → A, B → C} là một phủ không dư thừa của G

PHỦ KHÔNG DƯ THỪA (Cont.)

Một số lưu ý:

- Với một tập các phụ thuộc hàm cho trước có thể có nhiều hơn một phủ không dư thừa.
- Phủ không dư thừa của một tập các phụ thuộc hàm G có thể chứa những phụ thuộc hàm không nằm trong G.

Ví dụ: G = {A $\rightarrow$ B, B $\rightarrow$ A, B $\rightarrow$ C, A $\rightarrow$ C}

thì F = {A $\rightarrow$ B, B $\rightarrow$ A, AB $\rightarrow$ C} là một phủ không dư thừa của G. Tuy nhiên, F chứa phụ thuộc hàm AB $\rightarrow$ C không thuộc G.

CÁC THUỘC TÍNH DƯ THỪA

- Nếu F là một tập các phụ thuộc hàm không dư thừa thì F không thể nhỏ hơn bằng cách loại bỏ các phụ thuộc hàm.
- Để giảm kích cỡ của F, thực hiện loại bỏ các thuộc tính dư thừa từ các phụ thuộc hàm trong F.
- Nếu F là tập các phụ thuộc hàm trên lược đồ quan hệ R và $X \rightarrow Y \in F$ thì thuộc tính A được gọi là dư thừa trong $X \rightarrow Y \in F$ nếu:

  1. $X = AZ$, $X \neq Z$ and $\{F - \{X \rightarrow Y\}\} \cup \{Z \rightarrow Y\} \equiv F$, hoặc
  2. $Y = AW$, $Y \neq W$ and $\{F - \{X \rightarrow Y\}\} \cup \{X \rightarrow W\} \equiv F$

$\Rightarrow$ ***Một thuộc tính A dư thừa trong $X \rightarrow Y$ nếu A có thể được loại bỏ khỏi vế trái hoặc vế phải của phụ thuộc hàm mà không làm thay đổi F^+.***

CÁC THUỘC TÍNH DƯ THỪA (Cont.)

- ***Ví dụ:***

  Cho F = {A→ BC, B→ C, AB→ D}

  thuộc tính C là dư thừa trong vế phải của A→ BC

  tương tự, A là dư thừa trong vế trái của AB→ D

TẬP PHỤ THUỘC HÀM TỐI GIẢN TRÁI VÀ TỐI GIẢN PHẢI

Cho F là một tập các phụ thuộc hàm trên lược đồ R và cho $X \rightarrow Y \in F$.

- $X \rightarrow Y$ được gọi là tối giản trái nếu X không chứa các thuộc tính dư thừa A.
  - ***Một phụ thuộc hàm tối giản về trái cũng được gọi là một phụ thuộc hàm đầy đủ.***
- $X \rightarrow Y$ được gọi là tối giản phải nếu Y không chứa các thuộc tính dư thừa A.
- $X \rightarrow Y$ được gọi là tối giản nếu nó tối giản trái, tối giản phải và Y khác rỗng.

THUẬT TOÁN TỐI GIẢN TRÁI

Thuật toán tối giản trái sinh ra một tập các phụ thuộc hàm tối giản về trái.

```
Thuật toán tối giản trái  {trả lại tập các phụ thuộc hàm tối giản về trái F}
Đầu vào:  tập các phụ thuộc hàm G
Đầu ra: một phủ tối giản trái của G
Left-Reduce (G)
{
     F ← G;
     for mỗi phụ thuộc hàm X→ Y trong G do
          for mỗi thuộc tính A trong X do
               if Member(F, (X-A) → Y)
                    then loại bỏ A khỏi X trong X→ Y của F
     return(F);
}
```

THUẬT TOÁN TỐI GIẢN PHẢI

Thuật toán tối giản phải sinh ra một tập các phụ thuộc hàm tối giản về phải.

Thuật toán tối giản phải {trả lại tập các phụ thuộc hàm tối giản về trái F}
Đầu vào: tập các phụ thuộc hàm G
Đầu ra: một phủ tối giản phải của G
Right-Reduce (G)
{
    F ← G;
    for mỗi phụ thuộc hàm X→ Y trong G do
        for mỗi thuộc tính A trong Y do
            if Member(F – {X→ Y} ∪ {X → (Y- A)}, X → A)
                then loại bỏ A khỏi Y trong X→ Y của F
    return(F);
}

THUẬT TOÁN TỐI GIẢN PHỤ THUỘC HÀM

Thuật toán tối giản phụ thuộc hàm sinh ra một tập các phụ thuộc hàm tối giản.

```
Thuật toán tối giản phụ thuộc hàm
        {trả lại tập các phụ thuộc hàm tối giản F}
Đầu vào: tập các phụ thuộc hàm G
Đầu ra: một tập các phụ thuộc hàm tối giản G
Reduce (G)
{
    F ← Right-Reduce( Left-Reduce(G));
    loại bỏ tất cả các phụ thuộc hàm dạng X→ null từ F
    return(F);
}
```

Nếu G chứa một phụ thuộc hàm dư thừa X→ Y, thì mọi thuộc tính trong Y sẽ là dư thừa, và do vậy sẽ tối giản tới X → null, nên sẽ phải loại bỏ các phụ thuộc hàm này.

THUẬT TOÁN TỐI GIẢN PHỤ THUỘC HÀM (Cont.)

- Thứ tự thuật toán thực hiện rất quan trọng. ***Tập các phụ thuộc hàm phải được tối giản về trái trước rồi mới tối giản về phải.***

- ***Ví dụ:***

  Cho G = {B $\rightarrow$ A , D $\rightarrow$ A , BA $\rightarrow$ D}

  G là tối giản phải nhưng không phải tối giản trái. Nếu tối giản trái G để sinh ra F = {B $\rightarrow$ A , D $\rightarrow$ A , B $\rightarrow$ D}

  thì F là tối giản trái nhưng không phải tối giản phải!

  B $\rightarrow$ A là dư thừa bên vế phải vì B $\rightarrow$ D $\rightarrow$ A

PHỦ TỐI THIỂU

- ***Định nghĩa:*** Một tập phụ thuộc hàm F là tối thiểu nếu:
  1. Mọi phụ thuộc hàm đều có vế phải là một thuộc tính
  2. F là không dư thừa
  3. Không có phụ thuộc hàm nào dạng $X \rightarrow A$ có thể được thay thế bởi dạng $Y \rightarrow A$ với $Y \subseteq X$ và vẫn là một tập tương đương. Nói cách khác F là một tối giản trái.
- ***Ví dụ:***

  $G = \{A \rightarrow BCE,\ AB \rightarrow DE,\ BI \rightarrow J\}$

  Một phủ tối thiểu của G:

  $F = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, BI \rightarrow J\}$

THUẬT TOÁN TÌM PHỦ TỐI THIỂU

Thuật toán tìm phủ tối thiểu {trả lại phủ tối thiểu của F}
Đầu vào: tập các phụ thuộc hàm F
Đầu ra: một phủ tối thiểu của F
MinCover (F)
{
    G $\leftarrow$ F;
    thay thế mỗi phụ thuộc hàm $X \rightarrow A_1A_2...A_n$ của G
        thành n phụ thuộc hàm $X \rightarrow A_1$, $X \rightarrow A_2$,..., $X \rightarrow A_n$
    Left-Reduce(G);
    Nonredundant(G);
    return(G);
}