|  |
| --- |
| 5. CHUẨN HÓA DỮ LIỆU |
| 5.1. Chuẩn hoá dựa trên khoá chính |
| 5.1.1. Xác định khoá cho một lược đồ quan hệ |
| Nếu R là một lược đồ quan hệ với các thuộc tính A1, A2,  ..., An và một tập các phụ thuộc hàm F trong đó X ⊆ {A1,A2,...,An} thì X là một khoá của R nếu:   1. X → A1A2...An \fn_cm \in F+, và 2. không có tập con thực sự nào Y ⊆ X mà Y → A1A2...An \fn_cm \inF+.   về cơ bản mà nói, định nghĩa này cho thấy bạn phải tạo ra bao đóng của tất cả các tập con có thể có của R và quyết định xem tập nào suy diễn được ra tất cả các thuộc tính của lược đồ. **Một ví dụ về việc xác định khoá:** cho một quan hệ  r = (C, T, H, R, S, G) với tập phụ thuộc hàm F = {C → T, HR → C, HT → R, CS → G, HS → R}  Bước 1: Tạo (Ai)+ với 1\fn_cm \leqslant  i \fn_cm \leqslant n     C+ = {CT},   T+ = {T},   H+ = {H}     R+ = {R},      S+ = {S},    G+ = {G} Nhìn vào kết quả cho thấy không có thuộc tính đơn nào là khoá cho r Bước 2: Tạo (AiAj)+ với 1 \fn_cm \leqslanti \fn_cm \leqslantn, 1 \fn_cm \leqslantj \fn_cm \leqslantn     (CT)+ = {C,T},   (CH)+ = {CHTR},   (CR)+ = {CRT}     (CS)+ = {CSGT},   (CG)+ = {CGT},    (TH)+ = {THRC}     (TR)+ = {TR},   (TS)+ = {TS}, (TG)+ = {TG}      (HR)+ = {HRCT},   (HS)+ = {HSRCTG},  (HG)+ = {HG}     (RS)+ = {RS},   (RG)+ = {RG},   (SG)+ = {SG}     Tập thuộc tính (HS) là khoá của R  Bước 3: Tạo (AiAjAk)+ với 1 \fn_cm \leqslanti \fn_cm \leqslantn, 1 \fn_cm \leqslantj \fn_cm \leqslantn, 1 \fn_cm \leqslantk \fn_cm \leqslantn     (CTH)+ = {CTHR},   (CTR)+ = {CTR}     (CTS)+ = {CTSG}, (CTG)+ = {CTG}     (CHR)+ = {CHRT},   (CHS)+ = {CHSTRG}     (CHG)+ = {CHGTR}, (CRS)+ = {CRSTG}     (CRG)+ = {CRGT},  (CSG)+ = {CSGT}     (THR)+ = {THRC},   (THS)+ = {THSRCG}     (THG)+ = {THGRC}, (TRS)+ = {TRS}     (TRG)+ = {TRG},  (TSG)+ = {TSG}     (HRS)+ = {HRSCTG},   (HRG)+ = {HRGCT}     (HSG)+ = {HSGRCT},    (RSG)+ = {RSG} các siêu khoá được thể hiện bằng màu đỏ. Bước 4: Tạo (AiAjAkAr)+ với 1 \fn_cm \leqslanti \fn_cm \leqslantn, 1 \fn_cm \leqslantj \fn_cm \leqslantn, 1 \fn_cm \leqslantk \fn_cm \leqslantn, 1 \fn_cm \leqslantr \fn_cm \leqslantn      (CTHR)+ = {CTHR},   (CTHS)+ = {CTHSRG}     (CTHG)+ = {CTHGR}, (CHRS)+ = {CHRSTG}     (CHRG)+ = {CHRGT},   (CRSG)+ = {CRSGT}     (THRS)+ = {THRSCG},  (THRG)+ = {THRGC}     (TRSG)+ = {TRSG},  (HRSG)+ = {HRSGCT}     (CTRS)+ = {CTRS},   (CTSG)+ = {CTSG}     (CSHG)+ = {CSHGTR},  (THSG)+ = {THSGRC}     (CTRG)+ = {CTRG} Siêu khoá được thể hiện bằng màu đỏ. Bước 5: Tạo (AiAjAkArAs)+ với 1 \fn_cm \leqslanti \fn_cm \leqslantn, 1 \fn_cm \leqslantj \fn_cm \leqslantn, 1 \fn_cm \leqslantk \fn_cm \leqslantn, 1 \fn_cm \leqslantr \fn_cm \leqslantn, 1 \fn_cm \leqslants \fn_cm \leqslantn     (CTHRS)+ =  {CTHSRG}     (CTHRG)+ = {CTHGR}      (CTHSG)+ = {CTHSGR}      (CHRSG)+ = {CHRSGT}      (CTRSG)+ = {CTRSG}      (THRSG)+ = {THRSGC}     Siêu khoá được thể hiện bằng màu đỏ. Bước 6: Tạo (AiAjAkArAsAt)+ với 1 \fn_cm \leqslanti \fn_cm \leqslantn, 1 \fn_cm \leqslantj \fn_cm \leqslantn, 1 \fn_cm \leqslantk \fn_cm \leqslantn, 1 \fn_cm \leqslantr \fn_cm \leqslantn, 1 \fn_cm \leqslants \fn_cm \leqslantn, 1 \fn_cm \leqslantt \fn_cm \leqslantn     (CTHRSG)+ =  {CTHSRG}     Siêu khoá được thể hiện bằng màu đỏ. Nhìn chung, cho 6 thuộc tính chúng ta sẽ phải xét số trường hợp là \fn_cm \binom{6}{1} + \binom{6}{2} + \binom{6}{3} + \binom{6}{4} + \binom{6}{5} + \binom{6}{6}=6 + 15 + 20 + 15 + 6 + 1 = 63 |
| 5.1.2. Khái niệm chuẩn hoá |
| Chuẩn hoá là một kỹ thuật chính thống cho việc phân tích các quan hệ dựa trên khoá chính (hoặc khoá dự bị)  và các thuộc tính và các phụ thuộc hàm. Kỹ thuật này liên quan tới một chuỗi các luật có thể được dùng để kiểm tra từng quan hệ sao cho một cơ sở dữ liệu có thể được chuẩn hoá tới một mức độ nào đó. Khi một yêu cầu không được thoả mãn, quan hệ đã vi phạm yêu cầu đó được phân chia ra thành một tập các quan hệ khác mà mỗi trong số chúng đều thoả mãn các yêu cầu của sự chuẩn hoá đó. Việc chuẩn hoá thường được thực thi như một chuỗi các bước. Mỗi bước liên quan tới một dạng chuẩn cụ thể có những thuộc tính được biết rõ. Mối quan hệ giữa các dạng chuẩn hoá được mô tả trong các phần sau. |
| 5.2. Các dạng chuẩn hoá |
| Đối với mô hình quan hệ, một vấn đề rất quan trọng và thiết yếu là phải nhận ra được một quan hệ vừa được tạo ra đã ở dạng chuẩn một (1NF) chưa. Tất cả các dạng chuẩn mức cao hơn sau đó là tuỳ theo từng trường hợp, có thể có hoặc không. Tuy nhiên để tránh hiện tượng dị thường khi cập nhật mà chúng ta đã bàn luận đến trong bài trước, người thiết kế cơ sở dữ liệu thường được khuyến cáo phải đưa toàn bộ các quan hệ trong cơ sở dữ liệu về ít nhất là dạng chuẩn 3 (3NF). Như hình vẽ dưới đây mô tả thì một số quan hệ ở 1NF cũng sẽ ở 2NF, một số ở 2NF thì cũng ở 3NF, và cứ như  vậy với các mức chuẩn cao hơn.   |  | | --- | |  | |  |   Chúng ta sẽ xem các yêu cầu cho mỗi dạng chuẩn và một kỹ thuật phân tách để đạt được các lược đồ quan hệ trong các dạng chuẩn đó. |
| 5.2.1. Các dạng chuẩn ban đầu |
| 5.2.1.1. Dạng chưa phải chuẩn 1 |
| Các quan hệ ở dạng chưa chuẩn 1 là những quan hệ chứa một hoặc nhiều thuộc tính chưa phải nguyên tố. Nói một cách khác, trong một quan hệ và trong một bộ có nhiều thuộc đa trị. Một số mở rộng quan trọng tới mô hình quan hệ trong đó các quan hệ N1NF được sử dụng. Với hầu hết các phần còn lại trong phạm vi bài giảng chúng ta sẽ không bàn luận đến chúng một cách chi tiết. Các cơ sở dữ liệu quan hệ về thời gian và một số loại về không gian thường rơi vào loại N1NF *(Non-first normal form)* này. |
| 5.2.1.2. Dạng chuẩn 1 |
| Một quan hệ trong đó mỗi giá trị thuộc tính đều ở dạng nguyên tố thì quan hệ đó ở dạng 1NF (*First Normal Form 1*)  Chúng ta sẽ chỉ quan tâm tới các quan hệ dạng 1NF cho hầu hết các phần của môn học này. Lưu ý là các quan hệ chưa phải thuộc dạng chuẩn 1 nếu chúng chứa các thuộc tính ghép, các thuộc tính đa trị và các thuộc tính dẫn xuất. Khi gặp những thuộc tính đa trị ở mức khái niệm, nhắc lại rằng trong quá trình chuyển đổi sang mô hình dữ liệu thì cần tạo một bảng riêng biệt cho thuộc tính đa trị đó. Trước khi chuyển sang xem xét dạng chuẩn 2NF (*First Normal Form 2*), ta tìm hiểu một số thuật ngữ mới. ***Một khoá là một siêu khoá*** với các thuộc tính ràng buộc là nếu loại bỏ bất kể thuộc tính nào từ khoá này thì sẽ biến khoá đó không còn là một siêu khoá nữa. Nói một cách khác, khoá có số thuộc tính là nhỏ nhất. ***Một khoá dự bị*** cho một quan hệ là một tập các khoá nhỏ nhất cho lược đồ quan hệ đó. Khoá chính cho một quan hệ là một khoá dự bị được lựa chọn ra. Tất cả các khoá dự bị còn lại trở thành khoá phụ (hay khoá thứ cấp). ***Một thuộc tính khoá*** (chính) là thuộc tính của lược đồ quan hệ R mà là thành viên của một khoá dự bị nào đó của R. ***Thuộc tính không khoá*** là bất kỳ thuộc tính nào của R mà không phải là thành viên của một khoá dự bị nào cả. |
| 5.2.1.3. Dạng chuẩn 2 |
| Đây là dạng chuẩn dựa trên khái niệm của một phụ thuộc hàm đầy đủ. Một phụ thuộc hàm X → Y là một phụ thuộc hàm đầy đủ nếu loại bỏ bất kể thuộc tính A nào từ A sẽ làm cho phụ thuộc hàm này không còn đúng nữa.  với mọi thuộc tính A \fn_cm \in X, X-{A} không →  Y. Một phụ thuộc hàm X → Y là một phụ thuộc hàm một phần nếu một thuộc tính A nào đó có thể bị loại khỏi X mà phụ thuộc hàm đó vẫn đúng với một thuộc tính A \fn_cm \inX, X-{A}  → Y. ***Định nghĩa chuẩn 2NF:*** một lược đồ quan hệ R ở dạng 2 NF với một tập phụ thuộc hàm F nếu nó ở dạng 1NF và mọi thuộc tính không khoá đều phụ thuộc hàm đầy đủ vào mọi khoá của R. Một cách khác để phát biểu định nghĩa trên là: không tồn tại một thuộc tính không khoá nào mà phụ thuộc hàm một phần vào một khoá nào đó của R. Nói một cách khác, không có thuộc tính không khoá chỉ phụ thuộc vaò một phần của khoá của R Ví dụ về chuẩn 2NF: Cho quan hệ R = (A, D, P, G), F = {AD → PG, A → G} và K = {AD} Thì R không ở dạng 2NF bởi vì G phụ thuộc một phần vào khoá AD do AD → G nhưng A → G. Phân tách R thành:     R1 = (A, D, P)                  R2 = (A, G)     K1 = {AD}                         K2 = {A}     F1 = {AD → P}                 F2 = {A → G} |
| 5.2.1.4. Dạng chuẩn 3 |
| Dạng chuẩn 3NF dựa trên khái niệm về phụ thuộc bắc cầu. ***Định nghĩa phụ thuộc bắc cầu:*** cho một quan hệ R và một tập phụ thuộc hàm F của R, một tập con X \fn_cm \subseteq R và một thuộc tính A của R. A được nói là phụ thuộc hàm bắc cầu vào X nếu tồn tại Y \fn_cm \subseteqR mà X → Y, Y không → X và Y → A và  A \fn_cm \nsubseteq X \fn_cm \cupY. Một cách định nghĩa khác cho một phụ thuộc hàm bắc cầu là: một phụ thuộc hàm X → Y trong một lược đồ quan hệ R là phụ thuộc bắc cầu nếu có một tập thuộc tính Z \fn_cm \subseteqR mà Z không phải là tập con của bất kỳ khoá nào của R nhưng X → Z và Z → Y. ***Định nghĩa chuẩn 3NF:*** Một lược đồ quan hệ R ở dạng 3NF với một tập phụ thuộc hàm F nếu nó ở dạng 2NF và với bất kỳ phụ thuộc hàm X → A thì hoặc là (1) X là một siêu khoá của R hoặc (2) A là một thuộc tính khoá. ***Định nghĩa 3NF cách khác:*** Một lược đồ quan hệ R ở dạng 3NF với tập phụ thuộc hàm F nếu nó ở dạng 2NF và không có thuộc tính không khoá nào phụ thuộc hàm bắc cầu vào một khoá của R. Ví dụ: Cho R = (A, B, C, D) với K = {AB}, F = {AB → CD,  C → D,  D → C} thì R không ở dạng 3NF vì C → D và C không phải là siêu khoá của R. Một cách khác, R không ở dạng 3NF vì AB → C và C  → D và vì D là một thuộc tính không khoá nhưng lại phụ thuộc bắc cầu vào khoá AB. ***Tại sao cần chuẩn 3NF*** Chúng ta cùng phân tích xem tại sao lại cần dạng chuẩn 3NF thông qua việc xét cơ sở dữ liệu ví dụ sau đây với thể hiện dữ liệu của lược đồ quan hệ ở bảng sau đó. assign(flight, day, pilot-id, pilot-name)  K = {flight day} F = {pilot-id → pilot-name, pilot-name → pilot-id}   |  |  |  |  | | --- | --- | --- | --- | | **flight** | **day** | **pilot-id** | **pilot-name** | | 112 | Feb.11 | 317 | Mark | | 112 | Feb. 12 | 246 | Kristi | | 114 | Feb.13 | 317 | Mark |   Vì {flight day} là khoá nên rõ ràng là {flight day} → pilot-name. Nhưng trong F ta có pilot-name → pilot-id, và ta có {flight day} →  pilot-id. Bây giờ giả sử rằng ta thêm một bảng ghi mới vào bảng trên như bảng sau thì pilot-name ® pilot-id bị vi phạm. Một phụ thuộc hàm bắc cầu tồn tại vì  pilot-id → pilot-name và pilot-id không phải là siêu khoá.   |  |  |  |  | | --- | --- | --- | --- | | **flight** | **day** | **pilot-id** | **pilot-name** | | 112 | Feb.11 | 317 | Mark | | 112 | Feb. 12 | 246 | Kristi | | 114 | Feb.13 | 317 | Mark | | 112 | Feb. 11 | 319 | Mark | |
| 5.2.1.5. Chuẩn Boyce-Codd |
| Là một chuẩn đẹp hơn dạng chuẩn 3NF. ***Định nghĩa:*** một lược đồ quan hệ R ở dạng chuẩn BCNF với một tập các phụ thuộc hàm F nếu với bất kỳ phụ thuộc hàm dạng X → A nào và A ⊈ X, thì X là một siêu khoá của R. Ví dụ: Cho R = (A, B, C) với F = {AB → C,  C → A} K ={AB} thì R không ở dạng BCNF vì C → A và  C không phải là một siêu khoá của R. Lưu ý rằng sự khác nhau duy nhất trong định nghĩa của dạng chuẩn 3 và chuẩn BCNF là BCNF bỏ đi sự cho phép A trong X → A phải là thuộc tính khoá. Một khía cạnh thú vị đối với BCNF là Boyce và Codd ban đầu dự định dạng xây dựng dạng chuẩn này là một dạng đơn giản hơn chuẩn 3NF. Nói một cách khác, nó đã được mong muốn là nằm giữa 2NF và 3NF tuy nhiên sau đó nó được nhanh chóng chứng minh rằng thậm chí còn mạnh hơn cả 3 NF và vì vậy nó đứng ở vị trí giữa 3NF và 4NF. Trong thực tế, hầu hết các lược đồ quan hệ mà đã ở dạng chuẩn 3NF thì cũng ở dạng chuẩn BCNF. Chỉ nếu X → A trong lược đồ nhưng X không phải là một siêu khoá hoặc A là một thuộc tính khoá, thì lược đồ này ở dạng 3NF nhưng không ở dạng BCNF. |
| 5.3. Phân tách lược đồ quan hệ về các dạng chuẩn |
| Mục đích cơ bản của thiết kế cơ sở dữ liệu quan hệ là việc đảm bảo rằng mọi quan hệ trong cơ sở dữ liệu đều hoặc ở dạng chuẩn 3NF hoặc ở dạng chuẩn BCNF. 1NF và 2 NF không loại bỏ được một số lượng đủ các dị thường khi cập nhật để làm nên một sự khác biệt đáng kể trong khi 3NF và BCNF loại bỏ được hầu hết các dị thường khi cập nhật. Như chúng ta đã đề cập đến trước đây, thêm vào việc đảm bảo lược đồ quan hệ hoặc ở 3NF hoặc ở BCNF, người thiết kế phải đảm bảo rằng việc phân tách các lược đồ ban đầu trong cơ sở dữ liệu về hai dạng này phải thoả mãn hai thuộc tính là (1) kết nối không mất mát thông tin và (2) các phụ thuộc hàm được bảo toàn sau khi phân tách. Hiện tại cũng có một số các thuật toán phân tách đảm bảo chuẩn 3NF và đảm bảo không tổn thất thông tin và bảo toàn phụ thuộc hàm. Tuy nhiên, không có thuật toán nào đảm bảo phân tách được về BCNF mà vẫn đảm bảo kết nối không tổn thất thông tin và bảo toàn phụ thuộc hàm. Chỉ có một thuật toán đảm bảo không tổn thất thông tin về dạng BCNF nhưng không đảm bảo sự bảo toàn phụ thuộc hàm. Vì lý do này mà nhiều khi, chuẩn 3NF được coi là một dạng chuẩn mạnh trong khả năng có thể của một số lược đồ quan hệ vì nếu cố gắng đưa về BCNF vì có thể dẫn tới sự không bảo toàn các phụ thuộc hàm. Chúng ta sẽ xem xét hai thuộc tính này một cách kỹ càng trong phần tiếp sau đây. |
| 5.3.1. Bảo toàn các phụ thuộc hàm và thông tin |
| Bất kể khi nào một cập nhật được thực hiện đối với cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu phải có khả năng kiểm tra xem việc cập nhật đó sẽ không gây ra một thể hiện dữ liệu  không hợp lệ với các phụ thuộc hàm trong F+. Để kiểm tra các cập nhật này theo một cách có hiệu quả, cơ sở dữ liệu phải được thiết kế với một tập các lược đồ cho phép việc kiểm tra lại có thể thực hiện được mà không cần đến các phép toán kết nối. Nếu một phụ thuộc hàm nào đó không được bảo toàn thì cách duy nhất để có ràng buộc này là thực hiện phép kết nối hai hoặc nhiều hơn hai quan hệ trong quá trình phân tách này để nhận được một quan hệ bao gồm tất cả các thuộc tính vế trái và vế phải của phụ thuộc hàm bị mất đó trong cùng một bảng, sau đó có thể kiểm tra lại sự phụ thuộc vẫn được bảo toàn sau khi cập nhật. Hiển nhiên là việc này đòi hỏi rất nhiều công để hiện thực hoá một cách có hiệu quả. Một cách không chính thức, việc bảo toàn các phụ thuộc hàm có nghĩa là nếu X → Y từ F xuất hiện hoặc tường minh trong một lược đồ quan hệ của phép tách hoặc có thể suy diễn ra từ các phụ thuộc hàm xuất hiện trong một số lược đồ khác của phép tách thì tập phụ thuộc hàm ban đầu được bảo toàn trong lược đồ phân tách này. Một điều quan trọng cần nhớ là bảo toàn tập phụ thuộc hàm không có nghĩa là mọi phụ thuộc hàm trong F đều có mặt một cách tường minh trong một lược đồ quan hệ nào đó của phép tách, mà tốt hơn nên hiểu là hợp của tất cả các phụ thuộc hàm của từng lược đồ quan hệ trong phép tách phải tương đương với F (khái niệm tương đương ở đây có nghĩa là bao đóng của chúng là như nhau chứ chúng không nhất thiết phải giống hệt nhau). Phép chiếu của một tập các phụ thuộc hàm lên một tập các thuộc tính Z, ký hiệu là F[Z] (đôi khi còn gọi là \fn_cm \piZ(F)), là một tập các phụ thuộc hàm X → Y trong F+ sao cho X \fn_cm \cup Y \fn_cm \subseteq Z. Một lược đồ phân tách \fn_cm \gamma = {R1, R2, …, Rm} là bảo toàn phụ thuộc hàm trên tập F nếu hợp của các phép chiếu của F lên mỗi  Ri (1\fn_cm \leq i \fn_cm \leqm) trong \fn_cm \gammatương đương với F: (F[R1] \fn_cm \cup F[R2] \fn_cm \cup… \fn_cm \cupF[Rm])+ = F+ Thông thường thì có thể tìm thấy lược đồ phân tách D bảo toàn phụ thuộc hàm với tập phụ thuộc hàm F sao cho mỗi lược đồ quan hệ trong D đều ở dạng chuẩn 3NF. Tiếp tới chúng ta sẽ xem đến một thuật toán đảm bảo một sự phân tách về 3NF mà vẫn bảo toàn phụ thuộc hàm. |
| 5.3.1.1. Thuật toán kiểm tra tính bảo toàn phụ thuộc hàm |
| Algorithm Preserve  // đầu vào: một phân tách  D= (R1, R2, …, Rk),  một tập các phụ thuộc hàm F, một phụ thuộc hàm X → Y  //đầu ra:  true nếu D  được bảo toàn F, false nếu trái lại  Preserve (D , F, X → Y)     Z = X;     while (những thay đổi tới Z xảy ra) do            for i = 1 to k do  // có k lược đồ trong D                   Z = Z \fn_cm \cup ( (Z \fn_cm \cap Ri )+ \fn_cm \capRi )            endfor;      endwhile;      if Y \fn_cm \subseteq Z           then return true;  // Z ⊨ X → Y           else return false; end. Sự hoạt động của thuật toán trên như sau: tập Z được tính về có bản như sau: G=\fn_cm \bigcup_{i=1}^{k} F[Ri] Lưu ý là G không được tính trên thực tế mà cũng hiếm khi được kiểm tra xem thực sự G có phủ F hay không. Để kiểm tra xem G có phủ F hay không chúng ta cần xét mỗi phụ thuộc hàm X → Y trong F và xác định xem X\fn_cm _{G}^{+} có chứa Y hay không. Vì vậy kỹ thuật này là tính X\fn_cm _{G}^{+} mà không cần G có sẵn bằng cách lặp đi lặp lại xem xét bao đóng của F với các kết quả phép chiếu của F lên các R­i khác nhau. Một ví dụ chạy thuật toán trên: Giả sử  R = (A, B, C, D)  với  F={A→B, B→C,C→D,D→A} và phép tách D = {(AB),(BC),(CD)}; G = F[AB] \fn_cm \cup F[BC] \fn_cm \cupF[CD]  Z = Z \fn_cm \cup((Z \fn_cm \capRi)\fn_cm \cap Ri)  **Kiểm tra cho mỗi phụ thuộc hàm trong F:**  Kiểm tra cho  A → B  Z    = A,                     = {A} \fn_cm \cup((A \fn_cm \capAB)+  \fn_cm \capAB)                     = {A} \fn_cm \cup((A)+  \fn_cm \capAB)                     = {A}  \fn_cm \cup(ABCD  \fn_cm \capAB)                     = {A}  \fn_cm \cup{AB}                     = **{AB}**              Z     = {AB}                     = {AB} \fn_cm \cup((AB \fn_cm \capBC)+  \fn_cm \capBC)                     = {AB} \fn_cm \cup((B)+  \fn_cm \capBC)                     = {AB}  \fn_cm \cup(BCDA  \fn_cm \capBC)                     = {AB}  \fn_cm \cup{BC}                     = **{ABC}**              Z     = {ABC}                     = {ABC}  \fn_cm \cup((ABC \fn_cm \capCD)+  \fn_cm \capCD)                     = {ABC} \fn_cm \cup((C) \fn_cm \capCD)                     = {ABC}  \fn_cm \cup(CDAB \fn_cm \capCD)                     = {ABC}  \fn_cm \cup{CD}                     = **{ABCD}                                    Vì vậy G chứa A ®B**  Kiểm tra B →C                  Z = B,                     = {B} \fn_cm \cup((B \fn_cm \capAB)+  \fn_cm \capAB)                     = {B} \fn_cm \cup((B)+ \fn_cm \capAB)                     = {B}  \fn_cm \cup(BCDA  \fn_cm \capAB)                     = {B}  \fn_cm \cup{AB}                     = **{AB}**                  Z = {AB}                     = {AB} \fn_cm \cup((AB \fn_cm \capBC)+  \fn_cm \capBC)                     = {AB} \fn_cm \cup((B)+  \fn_cm \capBC)                     = {AB}  \fn_cm \cup(BCDA \fn_cm \capBC)                     = {AB}  \fn_cm \cup{BC}                     = **{ABC}**                  Z = {ABC}                     = {ABC} \fn_cm \cup((ABC \fn_cm \capCD)+  \fn_cm \capCD)                     = {ABC} \fn_cm \cup((C)+  \fn_cm \capCD)                     = {ABC}  \fn_cm \cup(CDAB  \fn_cm \capCD)                     = {ABC}  \fn_cm \cup{CD}                     = **{ABC}**                                        Vì vậy **G chứa B ®C**  Kiểm tra C →D                  Z = C,                     = {C} \fn_cm \cup((C \fn_cm \capAB)+  \fn_cm \capAB)                     = {C} \fn_cm \cup((\fn_cm \o)+  \fn_cm \capAB)                     = {C}  \fn_cm \cup(\fn_cm \o)                     = **{C}**                  Z = {C}                     = {C} \fn_cm \cup((C \fn_cm \capBC)+  \fn_cm \capBC)                     = {C} \fn_cm \cup((C)+  \fn_cm \capBC)                     = {C} \fn_cm \cup(CDAB  \fn_cm \capBC)                     = {C} \fn_cm \cup{BC}                     = **{BC}**                  Z = {BC}                     = {BC} \fn_cm \cup((BC \fn_cm \capCD)+ \fn_cm \capCD)                     = {BC} \fn_cm \cup((C)+  \fn_cm \capCD)                     = {BC} \fn_cm \cup(CDAB  \fn_cm \capCD)                     = {BC} \fn_cm \cup{CD}                     = **{BCD}**                                        Vì vậy **G chứa C ®D**  Kiểm tra D →A                  Z = D,                     = {D} \fn_cm \cup((D \fn_cm \capAB)+  \fn_cm \capAB)                     = {D} \fn_cm \cup((\fn_cm \o)+  \fn_cm \capAB)                     = {D} \fn_cm \cup(\fn_cm \o)                     = **{D}**                  Z = {D}                     = {D} \fn_cm \cup((D \fn_cm \capBC)+  \fn_cm \capBC)                     = {D} \fn_cm \cup((\fn_cm \o)+  \fn_cm \capBC)                     = {D} \fn_cm \cup(\fn_cm \o)                     = **{D}**                  Z = {D}                     = {D} \fn_cm \cup((D \fn_cm \capCD)+  \fn_cm \capCD)                     = {D} \fn_cm \cup((D)+  \fn_cm \capCD)                     = {D} \fn_cm \cup(DABC  \fn_cm \capCD)                     = {D} \fn_cm \cup{CD}                     = **{DC}**                                        G có thay đổi vì vậy tiếp tục**.**  Kiểm tra D →A tiếp tục ở lượt thứ hai qua D.                  Z = DC,                     = {DC} \fn_cm \cup((DC \fn_cm \capAB)+  \fn_cm \capAB)                     = {DC} \fn_cm \cup((\fn_cm \o)+  \fn_cm \capAB)                     = {DC} \fn_cm \cup(\fn_cm \o)                     = **{DC}**                  Z = {DC}                     = {DC} \fn_cm \cup((DC \fn_cm \capBC)+  \fn_cm \capBC)                     = {DC} \fn_cm \cup((C)+  \fn_cm \capBC)                     = {D} \fn_cm \cup(CDAB \fn_cm \capBC)                     = {D} \fn_cm \cup(BC)                     = **{DBC}**                  Z = {DBC}                     = {DBC} \fn_cm \cup((DBC \fn_cm \capCD)+  \fn_cm \capCD)                     = {DBC} \fn_cm \cup((CD)+  \fn_cm \capCD)                     = {DBC} \fn_cm \cup(CDAB  \fn_cm \capCD)                     = {DBC} \fn_cm \cup  {CD}                     = **{DBC}**                                        G có thay đổi vì vậy tiếp tục**.**  Kiểm tra cho D →A tiếp tục trên lượt thứ ba qua D.                  Z = DBC,                     = {DBC} \fn_cm \cup((DBC \fn_cm \capAB)+  \fn_cm \capAB)                     = {DBC} \fn_cm \cup((B)+  \fn_cm \capAB)                     = {DBC} \fn_cm \cup(BCDA \fn_cm \capAB)                     = {DBC} \fn_cm \cup(AB)                     = **{DBCA}**  Cuối cùng, chúng ta đưa mọi thuộc tính vào trong R.  Vì vậy, G chứa D → A. Vì vậy, D bảo toàn phụ thuộc hàm trong F. |
| 5.3.1.2. Thuật toán kiểm tra tính kết nối không tổn thất thông tin |
| Algorithm Lossless //đầu vào: một lược đồ quan hệ R= (A1, A2, …, An),   một tập các phụ thuộc hàm F, một lược đồ phân tách  D = {R1, R2, ..., Rk) // đầu ra:  true nếu D có tính kết nối không tổn thất thông, false nếu trái lại Lossless (R, F, D)  Tạo một ma trận có *n* cột và *k* hàng với cột *y* liên quan tới một thuộc tính Ay (1 \fn_cm \leq y \fn_cm \leqn) và hàng *x* liên quan tới lược đồ quan hệ Rx (1 \fn_cm \leqx \fn_cm \leqk).  Gọi ma trận này là T.  Điền vào ma trận theo cách sau: tại Txy điền vào kí hiệu ay nếu Ay nằm trong Rx và kí hiệu bxy nếu không phải vậy. Lặp lại các công việc sau với mỗi phụ thuộc hàm X → Y trong F cho đến khi không còn thay đổi đối với T. Mỗi khi một phụ thuộc hàm được xét đến, tìm kiếm các hàng trong T đồng nhất trong tất cả các cột liên quan tới các thuộc tính trong A. Chuyển giá trị của tất cả các ô trên các hàng có cùng giá trị X thành giá trị của Y theo cách sau: nếu bất kể một biểu tượng Y nào là ay thì chuyển tất cả chúng thành ay, nếu trong chúng không có ay thì cho chúng thành một giá trị bất kỳ trong số các giá trị bxy. Nếu sau khi thực hiện tất cả những thay đổi có thể đối với T một trong các hàng trở thành a1a2...an thì   trả về yes, trái lại trả về no.  end.  Xét một ví dụ chạy thuật toán này:  Cho R = (A, B, C, D, E); F = {A→C,B→C,C→D,DE→C,CE→A}; D = {(AD), (AB), (BE), (CDE), (AE)}  ma trận khởi tạo T:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b23 | b24 | b25 | | (BE) | b31 | a2 | b33 | b34 | a5 | | (CDE) | b41 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | b53 | b54 | a5 |   Xem xét mỗi phụ thuộc hàm trong F cho đến khi không có thay đổi nào được thực hiện vơi ma trận. A→C:  đồng hoá  b13, b23, b53một cách tuỳ ý chọn một giá trị cho chúng như bảng dưới đây   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b13 | b24 | b25 | | (BE) | b31 | a2 | b33 | b34 | a5 | | (CDE) | b41 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | b13 | b54 | a5 |   B→C:  đồng hoá  b13, b33một cách tuỳ ý chọn một giá trị cho chúng như bảng dưới đây   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b13 | b24 | b25 | | (BE) | b31 | a2 | b13 | b34 | a5 | | (CDE) | b41 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | b13 | b54 | a5 |   C→D:  đồng hoá  a4,  b24, b34,b54một cách tuỳ ý chọn một giá trị cho chúng như bảng dưới đây   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b13 | a4 | b25 | | (BE) | b31 | a2 | b13 | a4 | a5 | | (CDE) | b41 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | b13 | a4 | a5 |   DE→C:  đồng hoá  a3, b13một cách tuỳ ý chọn một giá trị cho chúng như bảng dưới đây   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b13 | a4 | b25 | | (BE) | b31 | a2 | a3 | a4 | a5 | | (CDE) | b41 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | a3 | a4 | a5 |   CE→A:  đồng hoá  a1, b31,b41một cách tuỳ ý chọn một giá trị cho chúng như bảng dưới đây   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b13 | a4 | b25 | | (BE) | a1 | a2 | a3 | a4 | a5 | | (CDE) | a1 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | a3 | a4 | a5 |   Lượt đầu tiên duyệt F đã hoàn thành, tuy nhiên hàng (BE) tất cả đã trở thành ai vì vậy kết thúc và trả về true, sự phân tách này thoả mãn kết nối không tổn thất thông tin.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (AD) | a1 | b12 | b13 | a4 | b15 | | (AB) | a1 | a2 | b13 | a4 | b25 | | (BE) | a1 | a2 | a3 | a4 | a5 | | (CDE) | a1 | b42 | a3 | a4 | a5 | | (AE) | a1 | b52 | a3 | a4 | a5 | |
| 5.3.2. Thuật toán phân tách lược đồ quan hệ về dạng chuẩn 3 |
| 5.3.2.1. Thuật toán số 1 |
| Algorithm 3NF.1 // đầu vào: một lược đồ quan hệ R= (A1, A2, …, An),   một tập các phụ thuộc hàm F, một tập các khoá dự bị K. // đầu ra:  một phân tách về 3NF của R, được gọi là D, thoả mãn tính kết nối không mất mát thông tin và bảo toàn các phụ thuộc hàm.  3NF.1 (R, F, K)     a = 0;     for each fd X → Y in F do                a = a +1;                Ra = XY;     endfor     if [không có lược đồ Rb (1 \fn_cm \leq b \fn_cm \leq a) nào chứa một khoá dự bị của R] then                a = a + 1;                Ra = một khoá dự bị nào đó của R     endif     if [\fn_cm \tiny \bigcup_{b=1}^{a} Rb \fn_cm \tiny \neqR] then   //bị thiếu một số thuộc tính                 Ra+1 =R - \fn_cm \tiny \bigcup_{b=1}^{a}Rb     return D = {R1, R2, ..., Ra+1} end.  Một ví dụ sử dụng thuật toán 3NF.1  Cho lược đồ quan hệ R = (A, B, C, D, E) với K = {AB, AC} và   F = {AB → CDE, AC → BDE, B → C, C → B, C → D, B → E}  Bước 1: D = {(ABCDE), (ACBDE), (BC), (CB), (CD), (BE)}  Tối giản tới: D = {(ABCDE), (BC), (CD), (BE)}  Bước 2: Kiểm tra xem D có chứa một khoá dự bị của R không? Có, trong (ABCDE)  Bước 3: Tất cả các thuộc tính của R có được chứa trong D? Có Trả về D là **{(ABCDE), (BC), (CD), (BE)}** |
| 5.3.2.2. Thuật toán số 2 |
| Algorithm 3NF.2  // đầu vào:  một lược đồ quan hệ R= (A1, A2, …, An),   một tập các phụ thuộc hàm F, một tập các khoá dự bị K.  // đầu ra:  một phân tách về 3NF của R, được gọi là D, không đảm bảo thoả mãn cả tính kết nối không mất mát thông tin và bảo toàn các phụ thuộc hàm.  //  Thuật toán này dựa trên ý tưởng loại bỏ các phụ thuộc hàm bắc cầu.  3NF.2 (R, F, K)      do  if [K → Y → A với A là một thuộc tính không khoá và không phải là một thành phần của K hoặc y] then phân tách R thành:  R1 = {R – A} với K1 = {K} và R2 = {YA} với K2 = {Y}.      repeat until không tồn tại các phụ thuộc hàm bắc cầu trong bất kể một lược đồ nào      D =  Hợp của tất cả các lược đồ 3NF được sinh ra ở trên.      kiểm tra tính kết nối không mất mát thông tin      kiểm tra tính bảo toàn phụ thuộc hàm  end.  Một ví dụ về sử dụng thuật toán 3NF.2  Cho lược đồ quan hệ R = (A, B, C, D, E) với K = {AB, AC} và   F = {AB → CDE, AC → BDE, B →C, C →B, C →D, B →E} Bước1: R không ở 3NF vì AB  → C  → D  Phân tách thành:  R1 = (A, B, C, E) với K1 = K = {AB, AC} và R2 = (C, D) with K2 = {C}  Bước 2: R2 ở 3NF.   R1 không ở 3NF vì AB →B → E  Phân tách R1 thành:  R11 = (A, B, C) với K11= K1 = K = {AB, AC}  R12 = (B, E) với K12 = {B}  Bước 3: R2, R11, và R12 đều ở 3NF Bước 4: Kiểm tra tính kết nối không mất mát thông tin như sau  AB →CDE: (lần thứ nhất: không đồng nhất ô nào cả) AC → BDE: (lần thứ nhất: không đồng nhất ô nào cả) B → C: (lần thứ nhất: đồng nhất a3 & b33) C → B: (lần thứ nhất: đồng nhất a2 & b12) C → D: (lần thứ nhất: đồng nhất b14, b24, b34) – dừng vì hàng thứ hai tất cả trở thành a B → E: (lần thứ nhất: đồng nhất a5, b15, b25)  Phân tách có thuộc tính kết nối không tổn thất thông tin   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | | (CD) | b11 | a2 | a3 | a4 | b15 | | (ABC) | a1 | a2 | a3 | a4 | b15 | | (BE) | b31 | a2 | a3 | a4 | a5 |   Bước 5: Kiểm tra xem có bảo toàn phụ thuộc hàm  Cho                  R = (A, B, C, D, E)                         F = {AB → CDE, AC → BDE, B → C, C →B, C → D, B →E}}                         D = {(CD), (ABC), (BE)} G = F[CD] \fn_cm \small \cup F[ABC] \fn_cm \small \cupF[BE]   Z = Z \fn_cm \small \cup((Z \fn_cm \small \cupRi)+  \fn_cm \small \cupRi)  Kiểm tra cho AB → CDE  Z = AB,     = {AB} \fn_cm \small \cup((AB \fn_cm \small \cap CD)+  \fn_cm \small \capCD)     = {AB} \fn_cm \small \cup((\fn_cm \small \o)+  \fn_cm \small \capCD)    = {AB} \fn_cm \small \cup(\fn_cm \small \o \fn_cm \small \capCD)     = {AB} \fn_cm \small \cup(\fn_cm \small \o)    = **{AB}**  Z = {AB} \fn_cm \small \cup((AB \fn_cm \small \capABC)+  \fn_cm \small \capABC)                = {AB} \fn_cm \small \cup((AB)+  \fn_cm \small \capABC)    = {AB} \fn_cm \small \cup(ABCDE \fn_cm \small \capABC)                         = {AB} \fn_cm \small \cup(ABC)    = **{ABC}** Z = {ABC} \fn_cm \small \cup((ABC \fn_cm \small \capBE)+  \fn_cm \small \capBE)                 = {ABC} \fn_cm \small \cup((B)+  \fn_cm \small \capBE)    = {ABC} \fn_cm \small \cup(BCDE \fn_cm \small \capBE)                             = {ABC} \fn_cm \small \cup(BE)    = **{ABCE}** Z = {ABCE} \fn_cm \small \cup((ABCE \fn_cm \small \capCD)+  \fn_cm \small \capCD)           = {ABCE} \fn_cm \small \cup((C)+  \fn_cm \small \capCD)    = {ABCE} \fn_cm \small \cup(CBDE \fn_cm \small \capCD)                          = {ABCE} \fn_cm \small \cup(CD)    = **{ABCDE} vì vậy AB** → **CDE được bảo toàm**  Kiểm tra cho AC → BDE  Z = AC     = {AC} \fn_cm \small \cup((AC \fn_cm \small \capCD)+ \fn_cm \small \capCD)    = {AC} \fn_cm \small \cup((C)+  \fn_cm \small \capCD)    = {AC} \fn_cm \small \cup(CBDE \fn_cm \small \capCD)                   = {AC} \fn_cm \small \cup(CD)    = **{ACD}** Z = {ACD} \fn_cm \small \cup((ACD \fn_cm \small \capABC)+  \fn_cm \small \capABC)         = {ACD} \fn_cm \small \cup((AC)+  \fn_cm \small \capABC)    = {ACD} \fn_cm \small \cup(ACBDE \fn_cm \small \capABC)                      = {ACD} \fn_cm \small \cup(ABC)    = **{ABCD}**  Z = {ABCD} \fn_cm \small \cup((ABCD \fn_cm \small \capBE)+  \fn_cm \small \capBE)           = {ABCD} \fn_cm \small \cup((B)+  \fn_cm \small \capBE)    = {ABCD} \fn_cm \small \cup(BCDE \fn_cm \small \capBE)                          = {ABCD} \fn_cm \small \cup(BE)    = **{ABCDE} vì vậy AC** → **BDE được bảo toàn**  Kiểm tra cho B → C  Z = B     = {B} \fn_cm \small \cup((B \fn_cm \small \capCD)+  \fn_cm \small \capCD)             = {B} \fn_cm \small \cup((C)+  \fn_cm \small \capCD)    = {B} \fn_cm \small \cup(CBDE \fn_cm \small \capCD)                     = {B} \fn_cm \small \cup(CD)    = **{BCD} vì vậy B** → **C được bảo toàn**  Kiểm tra cho C → B  Z = C     = {C} \fn_cm \small \cup((C \fn_cm \small \capCD)+  \fn_cm \small \capCD)             = {C} \fn_cm \small \cup((C)+  \fn_cm \small \capCD)    = {C} \fn_cm \small \cup(CBDE \fn_cm \small \capCD)                     = {C} \fn_cm \small \cup(CD)    = **{CD}** Z = {CD} \fn_cm \small \cup((CD \fn_cm \small \capABC)+  \fn_cm \small \capABC)                   = {CD} \fn_cm \small \cup((C)+  \fn_cm \small \capABC)    = {CD} \fn_cm \small \cup(CBDE \fn_cm \small \capABC)                           = {CD} \fn_cm \small \cup(BC)    = **{BCD} vì vậy C** → **B được bảo toàn**  Kiểm tracho  C →D  Z = C     = {C} \fn_cm \small \cup((C \fn_cm \small \capCD)+  \fn_cm \small \capCD)                          = {C} \fn_cm \small \cup((C)+  \fn_cm \small \capCD)    = {C} \fn_cm \small \cup(CBDE \fn_cm \small \capCD)                                = {C} \fn_cm \small \cup(CD)    = **{CD} vì vậy C** →**D được bảo toàn**  Kiểm tra cho B → E  Z = B     = {B} \fn_cm \small \cup((B \fn_cm \small \capCD)+  \fn_cm \small \capCD)               = {B} \fn_cm \small \cup((\fn_cm \small \o)+  \fn_cm \small \capCD)    = {B} \fn_cm \small \cup(\fn_cm \small \o)                                      = **{B}** Z = {B} \fn_cm \small \cup((B \fn_cm \small \capABC)+  \fn_cm \small \capABC)         = {B} \fn_cm \small \cup((B)+  \fn_cm \small \capABC)    = {B} \fn_cm \small \cup(BCDE \fn_cm \small \capABC)                   = {BC} \fn_cm \small \cup(BC)    = **{BC}** Z = {BC}     = {BC} \fn_cm \small \cup((BC \fn_cm \small \capABC)+  \fn_cm \small \capABC)     = {BC} \fn_cm \small \cup((C)+  \fn_cm \small \capABC)    = {BC} \fn_cm \small \cup(CBDE \fn_cm \small \capABC)                  = {BC} \fn_cm \small \cup(BC)    = **{BC}**  Z = {BC}    = {BC} \fn_cm \small \cup((BC \fn_cm \small \capBE)+  \fn_cm \small \capBE)           = {BC} \fn_cm \small \cup((B)+  \fn_cm \small \capBE)    = {BC} \fn_cm \small \cup(BCDE \fn_cm \small \capBE)                      = {BC} \fn_cm \small \cup(BE)    = **{BCE} vì vậy, B → E được bảo toàn**  ***Tại sao lại sử dụng 3NF.2 mà không sử dụng 3NF.1*** Bạn biết rằng thuật toán 3NF.1 sẽ đảm bảo việc phân tách về 3NF thoả mãn cả hai thuộc tính quan trọng là kết nối không tổn thất thông tin và bảo toàn phụ thuộc hàm, trong khi thuật toán 3NF.2 không đảm bảo cả hai thuộc tính đó, vậy tại sao bạn lại cần sử dụng thuật toán 3NF.2 mà không sử dụng 3NF.1 trong một số trường hợp. Câu trả lời rất đơn giản, Thuật toán 3NF.2 sẽ sinh ra ít lược đồ quan hệ hơn thuật toán 3NF.1. Mặc dù cả hai thuộc tính bảo toàn thông tin và bảo toàn phụ thuộc hàm đều cần phải được kiểm tra một cách độc lập khi sử dụng thuật toán 3NF.2 |
| 5.3.2.3. Thuật toán số 3 |
| Algorithm 3NF.3  // đầu vào: một lược đồ quan hệ R= (A1, A2, …, An),   một tập các phụ thuộc hàm F. // đầu ra:  một phân tách về 3NF của R, được gọi là D, thoả mãn tính kết nối không mất mát thông tin và bảo toàn các phụ thuộc hàm.  //  Thuật toán này dựa trên một phủ tối thiểu của F..  3NF.3 (R, F)      tìm phủ tối thiểu của F, gọi phủ này là G      for mỗi vế trái X xuất hiện trong G do            tạo một lược đồ quan  { X \fn_cm \small \cup A1 \fn_cm \small \cupA2 \fn_cm \small \cup... \fn_cm \small \cupAm} với Ai (1 \fn_cm \small \leq i \fn_cm \small \leqm) thể hiện                tất cả vế phải của các phụ thuộc hàm trong G với vế trái là X.      đưa toàn bộ các thuộc tính còn lại, nếu có, vào một lược đồ  nếu không có lược đồ nào chứa một khoá của R thì tạo thêm một lược đồ mới chứa   mọi khoá dự bị của R.  end.  Thuật toán 3NF.3 rất giống với thuật toán 3NF.1, chỉ khác về cách tạo ra các lược đồ phân tách   * Ở thuật toán 3Nf.1, các lược đồ được tạo ra trực tiếp từ F * Ở thuật toán 3NF.2, các lược đồ được tạo ra từ một phủ tối thiểu của F.   Nhìn chung, thuật toán 3NF.3 sẽ tạo ra ít lược đồ quan hệ hơn thuật toán 3NF.1. |