# Bài tập Chương 3

## Bài tập 3.1

* Định nghĩa View:
* CREATE VIEW V(ENO, ENAME, PNO, RESP) AS  
  SELECT EMP.ENO, EMP.ENAME, ASG.PNO, ASG.RESP  
  FROM EMP JOIN ASG ON EMP.ENO = ASG.ENO  
  WHERE ASG.DUR = 24;
* Khả năng cập nhật: View V **không thể cập nhật** một cách tự động vì nó được định nghĩa trên một phép nối (join) và không bao gồm tất cả các khóa của các quan hệ cơ sở.
* Lưu trữ định nghĩa: Định nghĩa của view V nên được lưu trữ tại **Site 2**, vì cả ASG1 (DUR < 36 bao gồm DUR=24) và EMP2 (có thể kết nối với ASG1) đều có thể có mặt tại đây, giúp tăng tính cục bộ [locality of reference].

## Bài tập 3.2

SELECT V.ENAME  
FROM V NATURAL JOIN PROJ  
WHERE PROJ.PNAME = 'CAD/CAM';

## Bài tập 3.3

Truy vấn sau khi được sửa đổi để chạy trên các mảnh sẽ trả về một tập hợp rỗng. Biểu thức đại số quan hệ được rút gọn là: Phép nối ASG1 ⋈\_{PNO} PROJ1 là rỗng vì ASG1 chứa các PNO từ ‘P1’ đến ‘P3’, trong khi PROJ1 (PNAME = "CAD/CAM") chỉ chứa PNO là ‘P3’. Tuy nhiên, tuple (E3, P3, ...) trong ASG có DUR=10, không thuộc ASG1. Do đó, không có tuple nào trong ASG1 có thể nối với PROJ1, và kết quả cuối cùng là rỗng.

## Bài tập 3.4

* Thuật toán làm mới snapshot: Một thuật toán hiệu quả là làm mới tăng dần (incremental refresh).
  1. Tại các site chứa mảnh của quan hệ cơ sở, khi có cập nhật, hệ thống sẽ tạo ra các quan hệ vi phân (differential relations) Δ+ (chèn) và Δ- (xóa).
  2. Gửi các quan hệ vi phân này đến site chứa snapshot.
  3. Tại site snapshot, áp dụng các thay đổi:
     + Snapshot\_new = (Snapshot\_old - π\_snapshot(Δ-)) ∪ π\_snapshot(Δ+)
* Ví dụ về kết quả không nhất quán: Điều này xảy ra khi làm mới bị trễ (lazy refresh).
  + Thời điểm t1: Giao dịch T1 cập nhật một tuple trên quan hệ cơ sở tại Site 1.
  + Thời điểm t2: Giao dịch T2 đọc snapshot tại Site 2 và nhận được dữ liệu cũ.
  + Thời điểm t3: Giao dịch làm mới (refresh transaction) cập nhật snapshot tại Site 2 với dữ liệu từ T1. Kết quả T2 nhận được là không nhất quán với trạng thái hiện tại của CSDL.

## Bài tập 3.5

* Áp dụng Counting Algorithm:
  + Cập nhật: UPDATE ASG SET RESP='Engineer' WHERE ENO='E3' AND PNO='P3'.
  + Hành động này tương đương với việc xóa tuple <A. Lee, Consultant> và tăng số đếm (count) cho tuple <A. Lee, Engineer> trong view EG.
  + Kết quả: tuple <A. Lee, Consultant> bị xóa khỏi view EG.
* Để view EG tự bảo trì (self-maintainable): View cần chứa các thuộc tính khóa của các quan hệ cơ sở. Do đó, cần thêm các thuộc tính EMP.ENO và ASG.PNO vào định nghĩa của view EG.

## Bài tập 3.6

Lược đồ quan hệ để lưu trữ quyền truy cập cho các nhóm người dùng trong danh mục phân tán có thể như sau: GROUP\_AUTH (GroupID, ObjectID, OperationType, Grantor, GrantOption)

Một lược đồ phân mảnh hợp lý là **phân mảnh ngang theo GroupID**. Vì giả định rằng tất cả thành viên của một nhóm ở cùng một site, việc phân mảnh theo GroupID và đặt mảnh đó tại site tương ứng sẽ cục bộ hóa việc kiểm tra quyền cho các thành viên của nhóm.

## Bài tập 3.7

Algorithm: Distributed\_REVOKE  
Input:   
 operation\_type, object, group\_id  
begin  
 // Giả sử site hiện tại là site quản lý group\_id  
 // Bước 1: Thu hồi quyền trực tiếp  
 DELETE FROM GROUP\_AUTH   
 WHERE GroupID = group\_id AND ObjectID = object AND OperationType = operation\_type;  
  
 // Bước 2: Xử lý thu hồi đệ quy (cascading revoke)  
 // Tìm tất cả các grantee mà group\_id đã cấp quyền  
 grantees\_to\_revoke ← SELECT Grantee FROM GRANT\_HIERARCHY   
 WHERE Grantor = group\_id AND ObjectID = object;  
   
 for each grantee in grantees\_to\_revoke do  
 // Gửi yêu cầu REVOKE đến site của grantee  
 // Thao tác này sẽ lại gọi đệ quy thuật toán Distributed\_REVOKE tại site đó  
 send\_message(grantee.site, "REVOKE", operation\_type, object, grantee);  
 end for;  
  
 // Xóa các bản ghi cấp quyền cũ  
 DELETE FROM GRANT\_HIERARCHY  
 WHERE Grantor = group\_id AND ObjectID = object;  
end

## Bài tập 3.8

Một phương án phân bổ PROJ\*\* trên hai site (Site S - Secret, Site C - Confidential) để tránh các kênh ẩn (covert channels) khi đọc là:

1. Phân mảnh và Sao chép:
   * Site C (Confidential): Lưu trữ một bản sao của tất cả dữ liệu được phân loại là C.
   * Site S (Secret): Lưu trữ tất cả dữ liệu, bao gồm cả dữ liệu S và một bản sao của toàn bộ dữ liệu C từ Site C.
2. Ràng buộc cập nhật:
   * Mọi cập nhật (viết) đối với dữ liệu C phải được thực hiện tại Site C và sau đó được truyền đến Site S.
   * Mọi cập nhật đối với dữ liệu S chỉ có thể được thực hiện tại Site S.
   * Theo quy tắc “no write down”, một chủ thể Secret tại Site S không được phép ghi vào bản sao dữ liệu C tại Site S. Cách này đảm bảo một truy vấn đọc từ một chủ thể Secret có thể được thực hiện hoàn toàn tại Site S mà không cần gửi thông tin Secret đến Site C, từ đó tránh được kênh ẩn.

## Bài tập 3.9

CHECK ON ASG (DUR <= 48);

Hoặc nếu muốn kiểm tra khi sửa đổi:

CHECK ON ASG WHEN MODIFY (NEW.DUR <= 48);

## Bài tập 3.10

Các pretest cho các ràng buộc trong các ví dụ từ 3.11 đến 3.14:

* Ví dụ 3.11 (NOT NULL): (EMP, INSERT, C) với C: ∀ NEW ∈ EMP+, NEW.ENO IS NOT NULL.
* Ví dụ 3.12 (UNIQUE): (ASG, INSERT, C) với C: ∀ NEW1, NEW2 ∈ ASG+, (NEW1.ENO, NEW1.PNO) ≠ (NEW2.ENO, NEW2.PNO) và ∀ NEW ∈ ASG+, ∀ a ∈ ASG, (NEW.ENO, NEW.PNO) ≠ (a.ENO, a.PNO).
* Ví dụ 3.13 (FOREIGN KEY):
  + (ASG, INSERT, C1) với C1: ∀ NEW ∈ ASG+, ∃j ∈ PROJ: NEW.PNO = j.PNO
  + (PROJ, DELETE, C2) với C2: ∀ g ∈ ASG, ∀ OLD ∈ PROJ−: g.PNO ≠ OLD.PNO
* Ví dụ 3.14 (FUNCTIONAL DEPENDENCY): (EMP, INSERT, C) với C: ∀ NEW ∈ EMP+, ∀ e ∈ EMP, (NEW.ENO = e.ENO ⇒ NEW.ENAME = e.ENAME).

## Bài tập 3.11

Dựa trên phân mảnh dọc đã cho:

* Pretest cho DUR ≤ 48 (từ bài 3.9) chỉ liên quan đến thuộc tính DUR của ASG. Do đó, nó chỉ cần được lưu trữ tại **Site 4**, nơi chứa mảnh ASG2 = {ENO, PNO, DUR}.

## Bài tập 3.12

* Ý nghĩa ràng buộc: Nếu một nhân viên trong bảng ASG có trách nhiệm là “Programmer”, thì chức danh (TITLE) của nhân viên đó trong bảng EMP cũng phải là “Programmer”.
* Pretests và lưu trữ:
  + Ràng buộc này liên quan đến EMP.TITLE và ASG.RESP.
  + Pretest khi INSERT vào ASG: (ASG, INSERT, C1) với C1: ∀ NEW ∈ ASG+, (NEW.RESP = "Programmer" ⇒ ∃e ∈ EMP: e.ENO = NEW.ENO AND e.TITLE = "Programmer").
    - Pretest này cần được lưu trữ tại các site có chứa EMP2 (Site 2) và ASG1 (Site 3). Việc kiểm tra sẽ yêu cầu truy cập cả hai site này.
  + Pretest khi MODIFY trên EMP.TITLE: (EMP, MODIFY, C2) với C2: ∀ NEW ∈ EMP+, (NEW.TITLE ≠ "Programmer" ⇒ ∀a ∈ ASG: a.ENO ≠ NEW.ENO OR a.RESP ≠ "Programmer").
    - Pretest này cũng cần được lưu trữ và kiểm tra tại các site chứa EMP2 và ASG1.
* Áp dụng ENFORCE cho INSERT vào ASG:
  1. Thuật toán ENFORCE sẽ chạy tại site chủ của truy vấn.
  2. Nó sẽ thực thi một truy vấn con để kiểm tra pretest C1 bằng cách nối (join) các tuple mới ASG+ với quan hệ EMP tại Site 2.
  3. Các tuple ASG+ sẽ được gửi đến Site 2.
  4. Site 2 thực hiện phép nối, lọc và trả về các tuple ASG+ vi phạm ràng buộc.
  5. Nếu kết quả trả về là rỗng, việc chèn được chấp nhận; ngược lại, nó bị từ chối.

## Bài tập 3.13

Một chiến lược đơn giản để kiểm tra ràng buộc khóa duy nhất toàn cục trên EMP trong một hệ thống không có hỗ trợ giao dịch toàn cục:

1. Giai đoạn kiểm tra (Detection Phase): Khi một giao dịch cục bộ tại Site 1 muốn chèn một tuple mới vào EMP của nó, trình quản lý toàn vẹn sẽ gửi một “yêu cầu kiểm tra khóa” (check-key request) chứa giá trị khóa mới đến Site 2.
2. Giai đoạn phản hồi (Response Phase): Site 2 nhận yêu cầu, kiểm tra xem khóa đã tồn tại trong EMP của nó hay chưa và gửi lại “OK” hoặc “DUPLICATE”.
3. Giai đoạn thực thi (Execution Phase): Nếu Site 1 nhận được “OK”, nó sẽ tiến hành chèn tuple. Xử lý vi phạm: Nếu một vi phạm được phát hiện (ví dụ: Site 1 nhận “DUPLICATE” hoặc cả hai site cố gắng chèn cùng một khóa đồng thời và phát hiện ra sau đó), một **giao dịch bù trừ (compensating transaction)** phải được thực thi để xóa tuple đã được chèn sai. Vì không có abort toàn cục, việc khắc phục phải được thực hiện bằng một giao dịch mới.