Bài tập chương 2

Nhóm 1  
Hồ Huy Hoàng – 18110294  
Lâm Quốc Khánh – 18110304  
Nguyễn Trần Phúc – 18110339  
Nguyễn Huỳnh Minh Tiến - 18110377

# Câu 1. Ràng buộc khóa ngoại là gì? Tại sao các ràng buôc như vậy là quan trọng? Tính toàn vẹn tham chiếu là gì?

* Ràng buộc khóa ngoại là :
  + Một khóa dùng để liên kết hai bảng với nhau.
  + Một trường (hoặc tập hợp các trường) trong một bảng tham chiếu đến KHÓA CHÍNH trong một bảng khác.
* Các ràng buộc quan trọng vì:
  + Tạo ra sự liên kết giữa các bảng.
  + Đảm bảo việc toàn vẹn tham chiếu.
  + Chống lại những hành động có thể làm sai dữ liệu.
* Toàn vẹn tham chiếu là: là 1 bộ giá trị trong một quan hệ tham chiếu tới một quan hệ tham chiếu tới một bộ giá trị đã tồn tại trong một quan hệ khác.

# Câu 2. Cho lược đồ CSDL gồm các lược đồ quan hệ sau:



## 2.1. Liệt kê tất cả các ràng buộc khóa ngoại giữa các quan hệ này.

|  |  |  |
| --- | --- | --- |
| **Bảng** | **Cột** | **References đến** |
| Enrolled | sid | Cột sid của Students |
| cid | Cột cid của Courses |
| Teaches | fid | Cột fid của Facilty |
| cid | Cột cid của Courses |
| Meets\_In | cid | Cột cid của Courses |
| rno | Cột rno của Rooms |

## 2.2. Cho một ví dụ về một ràng buộc (hợp lý) bao gồm một hoặc nhiều quan hệ trên nhưng không phải là ràng buộc khóa chính hay khóa ngoại.

* Dữ liệu của cột capacity của bảng Rooms phải >= 0.
* Dữ liệu của cột gpa của bảng Students phải >= 0…

## 2.3. Viết các câu lệnh tạo CSDL trên bao gồm tất cả ràng buộc có thể có.

CREATE DATABASE [CoursesManager]

GO

USE [CoursesManager]

GO

CREATE TABLE [Students] (

[sid] VARCHAR(10) NOT NULL,

[name] NVARCHAR(40) NULL,

[login] VARCHAR(16) NULL,

[age] INT NULL,

[gpa] REAL NULL,

CONSTRAINT [PK\_Students] PRIMARY KEY([sid])

)

GO

CREATE TABLE [Faculty] (

[fid] VARCHAR(10) NOT NULL,

[fname] NVARCHAR(50) NULL,

[sal] REAL NULL,

CONSTRAINT [PK\_Faculty] PRIMARY KEY([fid]),

)

GO

CREATE TABLE [Courses] (

[cid] VARCHAR(10) NOT NULL,

[cname] NVARCHAR(50) NULL,

[credits] INT NULL,

CONSTRAINT [PK\_Courses] PRIMARY KEY([cid]),

)

GO

CREATE TABLE [Room] (

[rno] INT NOT NULL,

[address] NVARCHAR(100) NULL,

[capacity] INT NULL,

CONSTRAINT [PK\_Room] PRIMARY KEY ([rno]),

)

GO

CREATE TABLE [Enrolled] (

[sid] VARCHAR(10) NOT NULL,

[cid] VARCHAR(10) NOT NULL,

[grade] NVARCHAR(30) NULL,

CONSTRAINT [FK\_Enrolled\_Students] FOREIGN KEY([sid]) REFERENCES [Students]([sid]),

CONSTRAINT [FK\_Enrolled\_Courses] FOREIGN KEY([cid]) REFERENCES [Courses]([cid]),

CONSTRAINT [PK\_Enrolled] PRIMARY KEY([sid], [cid]),

)

GO

CREATE TABLE [Teaches] (

[fid] VARCHAR(10) NOT NULL,

[cid] VARCHAR(10) NOT NULL,

CONSTRAINT [FK\_Teaches\_Faculty] FOREIGN KEY([fid]) REFERENCES [Faculty]([fid]),

CONSTRAINT [FK\_Teaches\_Courses]FOREIGN KEY([cid]) REFERENCES [Courses]([cid]),

CONSTRAINT [PK\_Teaches] PRIMARY KEY([fid], [cid]),

)

GO

CREATE TABLE [MeetsIn] (

[cid] VARCHAR(10) NOT NULL,

[rno] INT NOT NULL,

[time] VARCHAR(10) NULL,

CONSTRAINT [FK\_MeetsIn\_Room] FOREIGN KEY([rno]) REFERENCES [Room]([rno]),

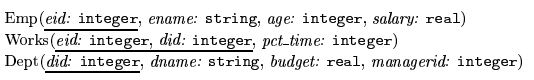
CONSTRAINT [FK\_MeetsIn\_Courses] FOREIGN KEY([cid]) REFERENCES [Courses]([cid]),

CONSTRAINT [PK\_MeetsIn] PRIMARY KEY([rno], [cid]),

)

GO

# Câu 3. Cho lược đồ CSDL gồm các lược đồ quan hệ sau, hãy trả lời các câu hỏi sau :



## 3.1. Cho một ví dụ về ràng buộc khóa ngoại có liên quan tới quan hệ Dept. Các lựa chọn nào có thể thực hiện để tuân thủ ràng buôc này khi người dùng xóa một bộ trong quan hệ Dept.

## Cột managerid của bảng Dept là khóa ngoại, references đến cột eid của bảng Emp.

## Cột did của bảng Works là khóa ngoại, references đến cột did của bảng Dept.

## Khi xóa một bộ trong quan hệ Dept, ta có thể:

## Xóa hết tất cả các bộ trong bảng Works nếu giá trị của khóa ngoại did của bảng Works chính là gí trị của cột did trong bộ bị xóa trong bảng Dept. (ON DELETE CASCADE)

## Set giá trị NULL cho tất cả các bộ trong bảng Works nếu giá trị của khóa ngoại did của bảng Works chính là gí trị của cột did trong bộ bị xóa trong bảng Dept. (ON DELETE SET NULL)

## 3.2. Viết các câu lệnh SQL để tạo các quan hệ trên bao gồm tất cả các RBTV khóa chính và khóa ngoại.

CREATE DATABASE Company;

GO

USE Company;

GO

CREATE TABLE Emp

(

    eid     INT PRIMARY KEY,

    ename   VARCHAR(30),

    age     INT,

    salary  REAL

);

GO

CREATE TABLE Dept

(

    did       INT PRIMARY KEY,

    dname     VARCHAR(50),

    budget    REAL,

    managerid INT

);

GO

CREATE TABLE Works

(

    eid         INT,

    did         INT,

    pct\_time    INT,

    CONSTRAINT FK\_EmpWorks FOREIGN KEY (eid) REFERENCES Emp(eid),

    CONSTRAINT FK\_DeptWorks FOREIGN KEY (did) REFERENCES Dept(did),

CONSTRAINT PK\_Works PRIMARY KEY (eid, did)

);

GO

## 3.3. Dùng SQL định nghĩa lại quan hệ Dept sao cho mọi department được đảm bảo có một người quản lý.

CREATE TABLE Dept

(

    did       INT PRIMARY KEY,

    dname     VARCHAR(50),

    budget    REAL,

    managerid INT NOT NULL

);

GO

## 3.4. Viết một câu lệnh SQL để thêm nhân viên ‘John Doe’ với eid = 101, age = 32, và salary = 15.000

CREATE TABLE Dept

(

    did       INT PRIMARY KEY,

    dname     VARCHAR(50),

    budget    REAL,

    managerid INT NOT NULL

);

GO

## 3.5. Viết một câu lệnh SQL để tăng 10% lương cho mọi nhân viên.

UPDATE Emp

SET salary = salary \* 1.1;

GO

## 3.6. Viết câu lệnh SQL để xóa department ‘Toy’. Với ràng buộc toàn vẹn tham chiếu bạn đã chọn cho lược đồ này, hãy giải thích điều gì xảy ra khi câu lệnh xóa này được thực hiện.

DELETE FROM Dept

WHERE dname = 'Toy';

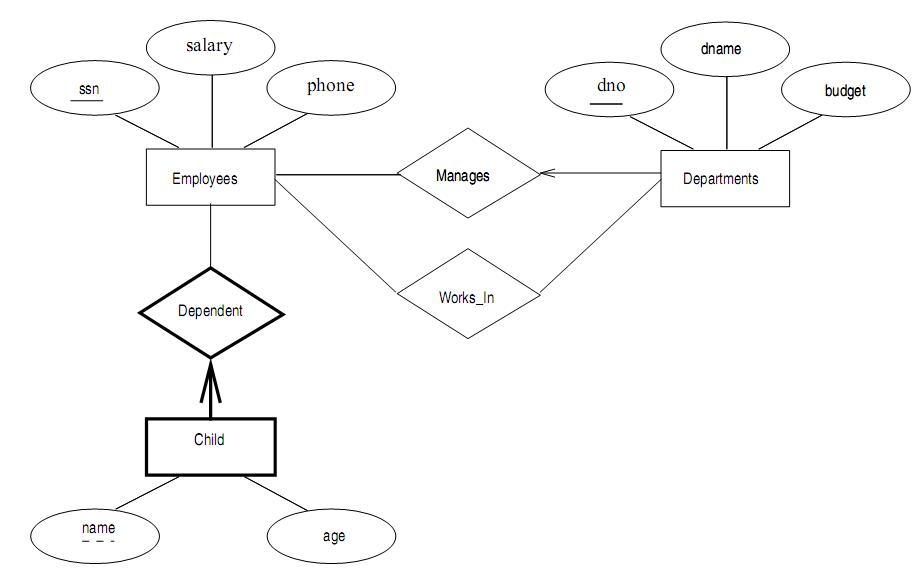
GO

-- Vì Bảng Works\_In, có cột did tham chiếu đến Dept(did).

-- Do đó, khi xóa 1 record trong bảng Dept thì sẽ vi phạm toàn vẹn dữ liệu  về tham chiếu.

-- Vì thế, câu truy vấn trên không thực hiện được trên SQL Server.

# Câu 4. Giả sử bạn đã thiết kế lược đồ ER cho CSDL một công ty như sau. Hãy viết các câu lệnh SQL để tạo các quan hệ tương ứng bao gồm nhiều nhất các ràng buộc có thể. Nếu có ràng buộc nào bạn không định nghĩa được, hãy giải thích tại sao.



CREATE DATABASE Company;

GO

USE Company;

GO

CREATE TABLE Employees

(

    ssn     INT PRIMARY KEY,

    salary  REAL,

    phone   VARCHAR(15)

);

GO

CREATE TABLE Departments

(

    dno       INT PRIMARY KEY,

    dname     VARCHAR(30),

    budget    REAL,

    managerid INT REFERENCES Employees(ssn)

);

GO

CREATE TABLE Child

(

    ssn     INT FOREIGN KEY REFERENCES Employees(ssn),

    cname   VARCHAR(30),

    age     INT,

    CONSTRAINT PK\_Child PRIMARY KEY (ssn, cname)

);

GO

CREATE TABLE Works\_In

(

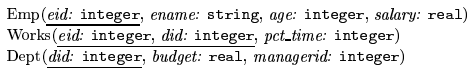
    ssn INT FOREIGN KEY REFERENCES Employees(ssn),

    dno INT FOREIGN KEY REFERENCES Departments(dno),

    CONSTRAINT PK\_WorksIn PRIMARY KEY (ssn, dno)

);

# Câu 5. Cho lược đồ quan hệ như sau, hãy trả lời các câu hỏi sau.



## 5.1. Định nghĩa một ràng buộc mức bảng trên Emp sao cho đảm bảo mọi nhân viên đều có lương ít nhất là 10.000

ALTER TABLE EMP ADD CONSTRAINT CHK\_SALARY CHECK (SALARY >= 10000)

## 5.2. Định nghĩa một ràng buộc mức bảng trên Dept sao cho đảm bảo tất cả người quản lý đều có tuổi lớn hơn 30.

CREATE FUNCTION UFN\_IS\_OLDER\_THAN\_30(@EMP\_ID INT)

RETURNS BIT AS

BEGIN

DECLARE @RESULT BIT

SELECT @RESULT = (CASE WHEN AGE > 30 THEN 1 ELSE 0 END) FROM EMP WHERE EID = 4

RETURN @RESULT

END

GO

ALTER TABLE DEPT ADD CONSTRAINT CHK\_DEPT\_MANAGER\_AGE CHECK (dbo.UFN\_IS\_OLDER\_THAN\_30(MANAGERID) = 1)

## 5.3. Định nghĩa một assertion trên Dept sao cho đảm bảo là mọi người quản lý đều có tuổi lớn hơn 30. So sánh assertion này với ràng buộc tạo ở câu 2. Hãy giải thích cái nào tốt hơn.

ALTER TABLE Dept

ADD ASSERTION Con\_AgeOfManager

CHECK (dbo.UFN\_IS\_OLDER\_THAN\_30(managerid) = 1);

GO

## 5.4. Viết câu lệnh SQL để xóa tất cả thông tin về những Employees có lương cao hơn lương của người quản lý của họ. Phải đảm bảo là tất cả các ràng buộc toàn vẹn liên quan phải được thỏa mãn sau khi cập nhật.

CREATE PROCEDURE USP\_DELETE\_EMP\_HIGH\_SALARY\_THAN\_MANAGER

AS

BEGIN

DECLARE @minSalaryManager REAL

-- Lấy lương thấp nhất quản lý

SET @minSalaryManager = (SELECT MIN(salary) FROM dbo.Emp INNER JOIN dbo.Dept ON Dept.ManagerId = Emp.eid)

-- Xóa những nhân viên(không là người quản lý) có lương cao hơn người quản lý

DELETE dbo.Works WHERE eid in (

SELECT eid from dbo.Emp

WHERE salary > @minSalaryManager AND eid NOT IN (SELECT ManagerId FROM dbo.Dept)

);

DELETE dbo.Emp WHERE salary > @minSalaryManager AND eid NOT IN (SELECT ManagerId FROM dbo.Dept);

END

GO

EXEC dbo.USP\_DELETE\_EMP\_HIGH\_SALARY\_THAN\_MANAGER

# Câu 6. Hãy cho biết điểm mạnh và điểm yếu của cơ chế trigger. Trình bày sự tương phản giữa trigger với các ràng buộc toàn vẹn khác được hỗ trợ bởi SQL.

Điểm mạnh của Trigger:

* Tính toán cập nhật dữ liệu tự động
* Kiểm tra dữ liệu nhập
* Kiểm tra ràng buộc phức tạp
* Sử dụng Trigger để kiểm tra tính toàn vẹn của cơ sở dữ liệu.
* Trigger có thể bắt lỗi logic ở mức cơ sở dữ liệu.
* Có thể dùng trigger là một cách khác để thay thế việc thực hiện những công việc hẹn giờ theo lịch.
* Trigger rất hiệu quả khi sử dụng để kiểm soát những thay đổi của dữ liệu trong bảng.

Điểm yếu của trigger:

* Không tạo và tham chiếu bảng
* Không tạo,thay đổi xóa cấu trúc đối tượng sẵn có trong CSDL
* Không gán cấp quyền cho người dùng.
* Trigger có thể tạo ra 1 bảng ảo mà khi thao tác trên CSDL nó sẽ tự động cập nhật
* Trigger chỉ là một phần mở rộng của việc kiểm tra tính hợp lệ của dữ liệu chứ không thể thay thế hoàn toàn được công việc này.
* Trigger hoạt động ngầm trong csdl, không hiển thị ở tầng giao diện. Do đó, khó chỉ ra được điều gì xảy ra ở tầng csdl.
* Trigger thực hiện các update lên bảng dữ liệu vì thế nó làm tăng lượng công việc lên csdl và làm cho hệ thống chạy chậm.

# Câu 7. Cho lược đồ quan hệ dưới đây. Một nhân viên có thể làm việc ở nhiều phòng ban. Cột pct\_time trong quan hê Works chỉ tỉ lệ % thời gian nhân viên đó làm việc cho phòng ban tương ứng. Viết các RBTV, assertions hoặc triggers để đảm bảo các yêu cầu sau:

## 7.1. Mỗi nhân viên phải có lương tối thiểu là 1000

ALTER TABLE [EMP] ADD CONSTRAINT [CK\_MINIMUM\_SALARY] CHECK (SALARY >= 10000)  
GO

## 7.2. Mọi người quản lý cũng là một nhân viên

Cách 1:

CREATE FUNCTION UFN\_IS\_EMP(@EMP\_ID INT)

RETURNS BIT AS

BEGIN

DECLARE @RESULT BIT

SELECT @RESULT = (CASE WHEN @EMP\_ID IN (SELECT eid FROM dbo.Emp) THEN 1 ELSE 0 END)

RETURN @RESULT

END

GO

ALTER TABLE dbo.Dept ADD CONSTRAINT CHK\_MANGER\_IS\_EMPLOYEE CHECK(dbo.UFN\_IS\_EMP(ManagerId) = 1)

GO

Cách 2:

ALTER TABLE Dept

ADD CONSTRAINT FK\_Manager FOREIGN KEY (managerid) REFERENCES Emp(eid);

GO

## 7.3. Tổng tỉ lệ % thời gian làm việc cho các phòng ban của một nhân viên phải dưới 100%.

CREATE FUNCTION dbo.UFN\_SUM\_PCT\_TIME(@eid INT)

RETURNS INT AS

BEGIN

    DECLARE @sum INT;

    SET @sum = 0;

    SELECT @sum = SUM(pct\_time)

    FROM Works

    WHERE eid = @eid;

    RETURN @sum;

END

GO

ALTER TABLE Works ADD CONSTRAINT CK\_PctTime   
CHECK (dbo. UFN\_SUM\_PCT\_TIME(eid) < 100);

GO

## 7.4. Một người quản lý phải luôn có lương cao hơn bất kỳ một nhân viên nào mà người đó quản lý.

CREATE FUNCTION UFN\_MAX\_EMPLOYEE\_SALARY(@EMP\_ID INT)

RETURNS DECIMAL(19,4) AS

BEGIN

DECLARE @RESULT DECIMAL(19,4)

IF (@EMP\_ID NOT IN (SELECT MANAGERID FROM dbo.Dept))

BEGIN

SET @RESULT = 0

END

ELSE

BEGIN

SELECT @RESULT = MAX(SALARY)

FROM EMP AS E, DEPT AS D, WORKS AS W

WHERE E.EID = W.EID AND W.EID = E.EID AND E.EID = @EMP\_ID

GROUP BY (D.DID)

END

RETURN @RESULT

END

GO

ALTER TABLE dbo.EMP ADD CONSTRAINT CHK\_MANAGER\_SALARY CHECK (SALARY > dbo.UFN\_MAX\_EMPLOYEE\_SALARY(EID))

## 7.5. Bất cứ khi nào một nhân viên được tăng lương, lương người quản lý cũng phải được tăng tương ứng.

CREATE PROCEDURE USP\_UPDATE\_MANAGER\_SALARY @EMP\_ID INT, @AMOUNT DECIMAL(19,4)

AS BEGIN

UPDATE dbo.Emp

SET SALARY = SALARY + @AMOUNT

WHERE EID IN (

SELECT managerid

FROM dbo.Emp AS E, dbo.Works AS W, dbo.Dept AS D

WHERE D.did = W.did AND W.eid = E.eid AND E.eid = @EMP\_ID

)

END

GO

CREATE TRIGGER UPDATE\_MANAGER\_SALARY\_BY\_EMP\_SALARY ON EMP AFTER UPDATE

AS BEGIN

DECLARE @OLD\_SALARY DECIMAL(19, 4)

DECLARE @NEW\_SALARY DECIMAL(19, 4)

DECLARE @EMP\_ID INT

SELECT @OLD\_SALARY = deleted.salary FROM deleted

SELECT @NEW\_SALARY = inserted.salary, @EMP\_ID = inserted.eid FROM inserted

IF @NEW\_SALARY - @OLD\_SALARY > 0

BEGIN

SET @NEW\_SALARY = @NEW\_SALARY - @OLD\_SALARY

-- calculate amount and store to @NEW\_SALARY

EXEC USP\_UPDATE\_MANAGER\_SALARY @EMP\_ID, @NEW\_SALARY

END

END

GO

## 7.6. Bất cứ khi nào một nhân viên được tăng lương, lương người quản lý cũng phải được tăng tương ứng. Hơn nữa, bất cứ khi nào một nhân viên được tăng lương, ngân sách của phòng ban tương ứng cũng phải được tăng lớn hơn tổng lương của tất cả nhân viên thuộc phòng đó.

CREATE PROCEDURE USP\_UPDATE\_MANAGER\_SALARY @EMP\_ID INT, @AMOUNT DECIMAL(19,4)

AS BEGIN

UPDATE dbo.Emp

SET SALARY = SALARY + @AMOUNT

WHERE EID IN (

SELECT managerid

FROM dbo.Emp AS E, dbo.Works AS W, dbo.Dept AS D

WHERE D.did = W.did AND W.eid = E.eid AND E.eid = @EMP\_ID

)

END

GO

CREATE PROCEDURE USP\_UPDATE\_DEPT\_BUDGET @EMP\_ID INT, @AMOUNT DECIMAL(19,4)

AS BEGIN

UPDATE dbo.Dept

SET BUDGET = BUDGET + @AMOUNT

WHERE DID IN (

SELECT DID

FROM dbo.Emp AS E, dbo.Dept AS D , dbo.WORKS AS W

WHERE W.did = D.DID AND W.EID = E.EID AND E.EID = @EMP\_ID

)

END

GO

CREATE TRIGGER UPDATE\_DEPT\_BUDGET\_BY\_EMP\_SALARY ON EMP AFTER UPDATE

AS BEGIN

DECLARE @OLD\_SALARY DECIMAL(19, 4)

DECLARE @NEW\_SALARY DECIMAL(19, 4)

DECLARE @EMP\_ID INT

SELECT @OLD\_SALARY = deleted.salary FROM deleted

SELECT @NEW\_SALARY = inserted.salary, @EMP\_ID = inserted.eid FROM inserted

IF @NEW\_SALARY - @OLD\_SALARY > 0

BEGIN

SET @NEW\_SALARY = @NEW\_SALARY - @OLD\_SALARY

-- calculate amount and store to @NEW\_SALARY

EXEC USP\_UPDATE\_MANAGER\_SALARY @EMP\_ID, @NEW\_SALARY

EXEC USP\_UPDATE\_DEPT\_BUDGET @EMP\_ID, @NEW\_SALARY

END

END

GO

# Câu 8. Cho CSDL quan hệ của một công ty như dưới đây. Dùng SQL để xác định các views sau:

## 8.1. Một view có department name, manager name và manager salary của mọi phòng ban.

CREATE VIEW DEPT\_MANAGER\_INF(DepartmentName, ManagerName, ManagerSalary)

AS SELECT D.DNAME, CONCAT(E.FNAME, ' ', E.LNAME), E.SALARY

FROM dbo.DEPARTMENT AS D INNER JOIN dbo.EMPLOYEE AS E

ON E.SSN = D.MGR\_SSN

GO

## 8.2. Một view có employee name, supervisor name và employee salary của mỗi nhân viên thuộc phòng ‘Research’

CREATE VIEW DEPT\_RESEARCH (EmployeeName, SupervisorName, EmployeeSalary)

AS SELECT CONCAT(E.FNAME, ' ', E.LNAME), CONCAT(SE.FNAME, ' ', SE.LNAME), E.Salary

FROM dbo.EMPLOYEE AS E INNER JOIN dbo. DEPARTMENT AS D

ON D.Dnumber = E.Dno AND D.Dname = 'Research'

INNER JOIN dbo.EMPLOYEE AS SE ON E.SUPER\_SSN = SE.SSN

## 8.3. Một view có project name, controlling department name, number of employees và tổng số giờ được làm việc mỗi tuần của mỗi dự án.

CREATE VIEW PROJECT\_MANAGER(ProjectName, DepartmentName, NumberOfEmployee, SumOfHours)

AS SELECT P.PNAME, D.DNAME, COUNT(E.SSN), SUM(W.HOURS)

FROM PROJECT AS P, EMPLOYEE AS E, WORKS\_ON AS W, DEPARTMENT AS D

WHERE P.PNUMBER = W.PNO AND E.SSN = W.ESSN AND P.DNUM = P.DNUM

GROUP BY P.PNAME, D.DNAME

## 8.4. Một view có project name, controlling department name, number of employees và tổng số giờ được làm việc mỗi tuần của mỗi dự án với những dự án có nhiều hơn một nhân viên làm việc cho dự án đó.

CREATE VIEW PROJECT\_MANAGER\_GREAT\_THAN\_1\_EMPLOYEE(ProjectName, DepartmentName, NumberOfEmployee, SumOfHours)

AS SELECT P.PNAME, D.DNAME, COUNT(E.SSN), SUM(W.HOURS)

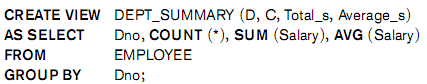
FROM PROJECT AS P, EMPLOYEE AS E, WORKS\_ON AS W, DEPARTMENT AS D

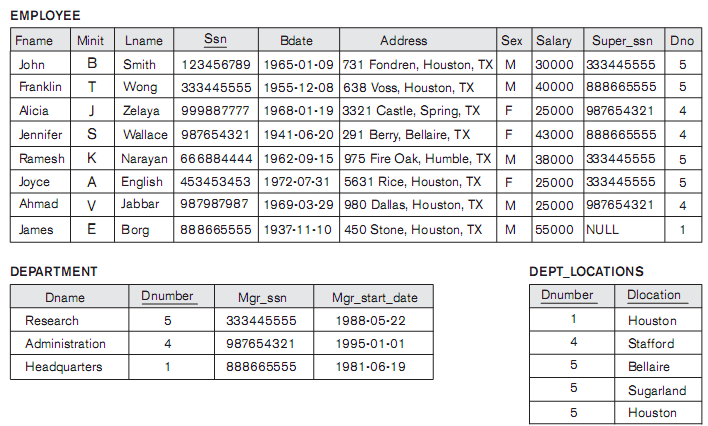
WHERE P.PNUMBER = W.PNO AND E.SSN = W.ESSN AND P.DNUM = P.DNUM

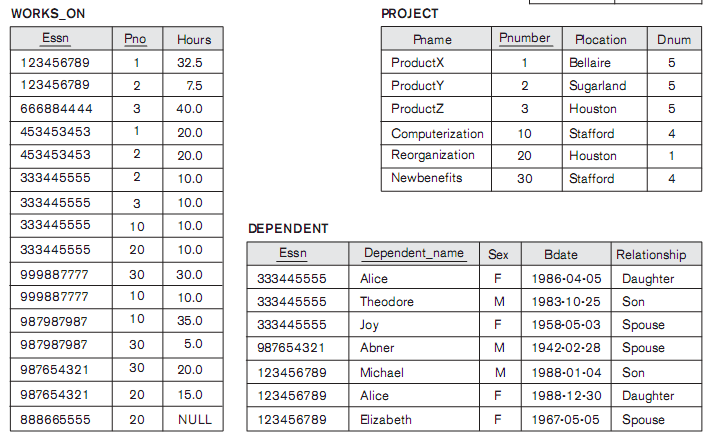
GROUP BY P.PNAME, D.DNAME

HAVING COUNT(E.SSN) > 1

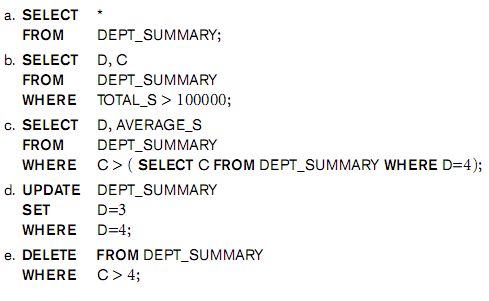
# Câu 9. Cho view DEPT\_SUMMARY được định nghĩa trên CSDL một công ty như sau.







Hãy cho biết các truy vấn và thao tác cập nhật nào sau đây được phép trên view. Nếu một query hoặc update nào được phép, hãy cho biết câu truy vấn hoặc cập nhật tương ứng được thực hiện trên các quan hệ cơ sở tương ứng với view.



|  |  |  |
| --- | --- | --- |
|  | Có thể thực hiện | Câu lệnh thực hiện trên các quan hệ cơ sở tương ứng với view |
| a | Có | SELECT Dno AS D, COUNT(\*) AS C, SUM(Salary) AS Total\_s, AVG(Salary) AS Average\_s FROM dbo.EMPLOYEE GROUP BY Dno |
| b | Có | SELECT Dno AS D, COUNT(\*) AS C FROM dbo.EMPLOYEE GROUP BY Dno HAVING SUM(Salary) > 100000 |
| c | Có | SELECT Dno AS D, AVG(Salary) AS Average\_s FROM dbo.EMPLOYEE GROUP BY Dno HAVING COUNT(\*) > (SELECT COUNT(\*) AS C FROM dbo.EMPLOYEE GROUP BY Dno HAVING Dno = 4) |
| d | Không |  |
| e | Không |  |