Họ tên: Nguyễn Huỳnh Minh Tiến

MSSV: 18110377

**BÀI TẬP CHƯƠNG 2.**

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 trong bảng cơ sở dữ liệu là trường phù hợp với trường khóa chính trong bảng khác. Trường khóa ngoại được sử dụng để tạo ra mối quan hệ với trường khóa chính của bảng chính. Điều này ngăn cản bạn chèn thêm bất kỳ dữ liệu không hợp lệ trong trường khóa ngoại vì trường khóa ngoại chỉ chấp nhận những giá trị đã có trong trường khóa chính.

Tính toàn vẹn tham chiếu là sự nhất quán về CSDL giữa 2 quan hệ tham chiếu với nhau.

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



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 |

1. 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…

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

CREATE DATABASE University;

GO

USE University

GO

CREATE TABLE Students

(

    s\_id     VARCHAR(10) PRIMARY KEY,

    s\_name   VARCHAR(50),

    s\_login  VARCHAR(20),

    age      VARCHAR(3),

    gpa      REAL

);

GO

CREATE TABLE Faculty

(

    f\_id    VARCHAR(10) PRIMARY KEY,

    f\_name  VARCHAR(50),

    sal     REAL

);

GO

CREATE TABLE Courses

(

    c\_id    VARCHAR(10) PRIMARY KEY,

    c\_name  VARCHAR(50),

    credits INT

);

GO

CREATE TABLE Rooms

(

    r\_no      INT PRIMARY KEY,

    r\_address VARCHAR(100),

    capacity  INT

);

GO

CREATE TABLE Enrolled

(

    s\_id    VARCHAR(10) FOREIGN KEY REFERENCES Students(s\_id),

    c\_id    VARCHAR(10) FOREIGN KEY REFERENCES Courses(c\_id),

    grade   VARCHAR(20)

);

GO

CREATE TABLE Teaches

(

    f\_id    VARCHAR(10) FOREIGN KEY REFERENCES Faculty(f\_id),

    c\_id    VARCHAR(10) FOREIGN KEY REFERENCES Courses(c\_id)

);

GO

CREATE TABLE Meets\_In

(

    c\_id    VARCHAR(10) FOREIGN KEY REFERENCES Courses(c\_id),

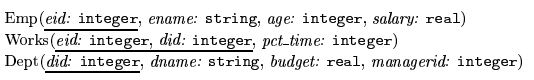
    r\_no    INT FOREIGN KEY REFERENCES Rooms(r\_no),

    at\_time VARCHAR(20)

);

GO

Câu 3. Cho lược đồ CSDL gồm các lược đồ quan hệ sau:



Hãy trả lời các câu hỏi sau :

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)

1. 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

1. 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

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

INSERT INTO Emp VALUES(101, 'John Doe', 32, 15.000);

GO

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

UPDATE Emp

SET salary = salary \* 1.1;

GO

1. 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';

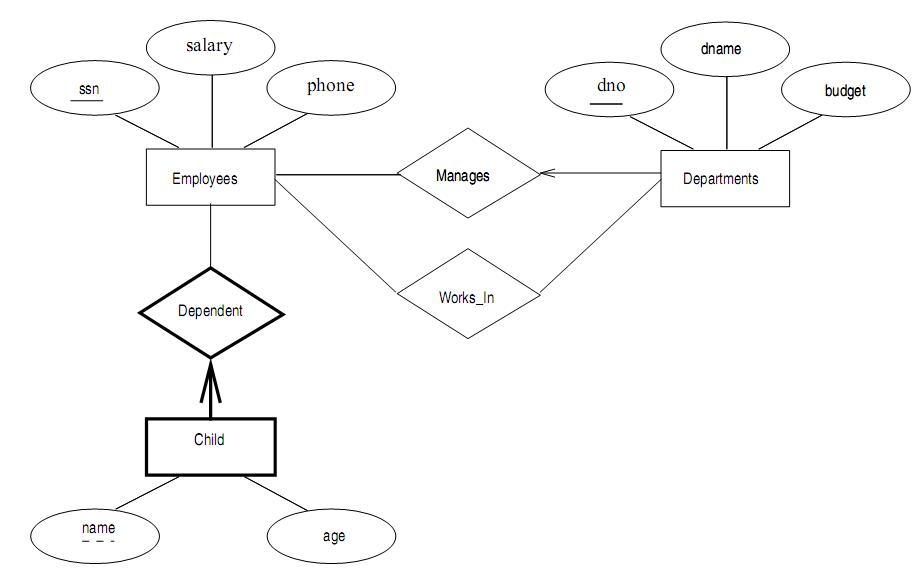
GO

-- 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),

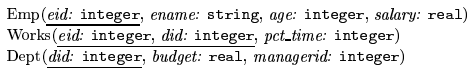
    CONSTRAINT PK\_WorksIn PRIMARY KEY (ssn, dno)

);

GO

Câu 5.

Cho lược đồ quan hệ sau:



Hãy trả lời các câu hỏi sau.

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 Cons\_Salary CHECK (salary >= 10.000);

GO

1. 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 dbo.AgeOfManager(@managerid int)

RETURNS INT AS

BEGIN

    DECLARE @age INT;

    SET @age = (SELECT Emp.age FROM Emp WHERE Emp.eid = @managerid);

    RETURN @age;

END;

GO

ALTER TABLE Dept

ADD CONSTRAINT Con\_Age

CHECK (dbo.AgeOfManager(managerid) > 30);

GO

1. 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.AgeOfManager(managerid) > 30);

GO

1. 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 FUNCTION dbo.GetSalaryOfManager(@eid INT)

RETURNS INT AS

BEGIN

    DECLARE @did INT;

    SET @did = (SELECT did FROM Works WHERE eid = @eid);

    DECLARE @managerid INT;

    SET @managerid = (SELECT managerid FROM Dept WHERE did = @did);

    DECLARE @salary INT;

    SET @salary = (SELECT Emp.salary FROM Emp WHERE Emp.eid = @managerid);

    RETURN @salary;

END

GO

CREATE PROCEDURE dbo.DeleteEmployees AS

BEGIN

    DECLARE @eid\_table TABLE(eid INT NOT NULL);

    INSERT INTO @eid\_table

        SELECT eid

        FROM Emp

        WHERE salary > dbo.GetSalaryOfManager(eid);

    DELETE FROM Works

    WHERE Works.eid IN (SELECT eid FROM @eid\_table);

    DELETE FROM Emp

    WHERE Emp.eid IN (SELECT eid FROM @eid\_table);

END

GO

EXEC dbo.DeleteEmployees;

GO

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 viêc ờ 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:

1. Mỗi nhân viên phải có lương tối thiểu là 1000

ALTER TABLE Emp

ADD CONSTRAINT CK\_Salary CHECK (salary >= 1000);

GO

1. Mọi người quản lý cũng là một nhân viên

ALTER TABLE Dept

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

GO

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

CREATE FUNCTION dbo.SumPctTime(@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.SumPctTime(eid) < 100);

GO

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

CREATE FUNCTION dbo.GetSalaryOfManager(@eid INT)

RETURNS INT AS

BEGIN

    DECLARE @did INT;

    SET @did = (SELECT did FROM Works WHERE eid = @eid);

    DECLARE @managerid INT;

    SET @managerid = (SELECT managerid FROM Dept WHERE did = @did);

    DECLARE @salary INT;

    SET @salary = (SELECT Emp.salary FROM Emp WHERE Emp.eid = @managerid);

    RETURN @salary;

END

GO

ALTER TABLE Emp

ADD CONSTRAINT CK\_SalaryEmp

CHECK (salary < dbo.GetSalaryOfManager(eid));

GO

1. Bất cứ khi nào một nhân viên được tăng lương, lương người quản lý cũng phải được tăng tương ứng.
2. 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.

Câu 8. Cho CSDL quan hệ của một công ty như dưới đây.

Dùng SQL để xác định các views sau :

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

CREATE VIEW view\_1 AS

SELECT dname as [Department name], ename as [Manager name], salary as [Manager salary]

FROM Dept, Emp

WHERE managerid = eid

GO

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

CREATE VIEW view\_2 AS

SELECT Emp.ename AS [Employee name], T.ename AS [Supervisor name], salary AS [Employee salary]

FROM Emp,

    (SELECT ename

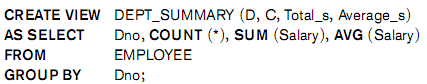
    FROM Dept, Emp

    WHERE managerid = eid AND dname = 'Research') AS T;

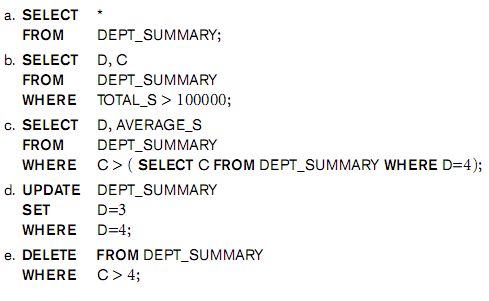
GO

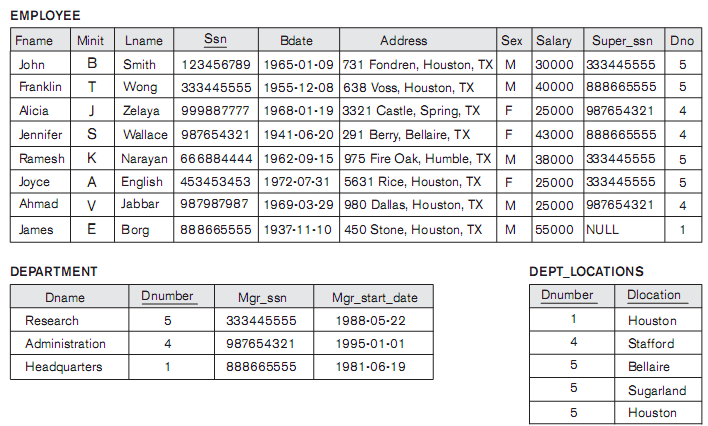
1. Một view có project name, controlling department name, number of employees và tổng số giờ được làm việc mỗi tuần của mỗi dự án.
2. Một view có project name, controlling department name, number of employees và tổng số giờ được làm việc mỗi tuần của mỗi dự án với những dự án có nhiều hơn một nhân viên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

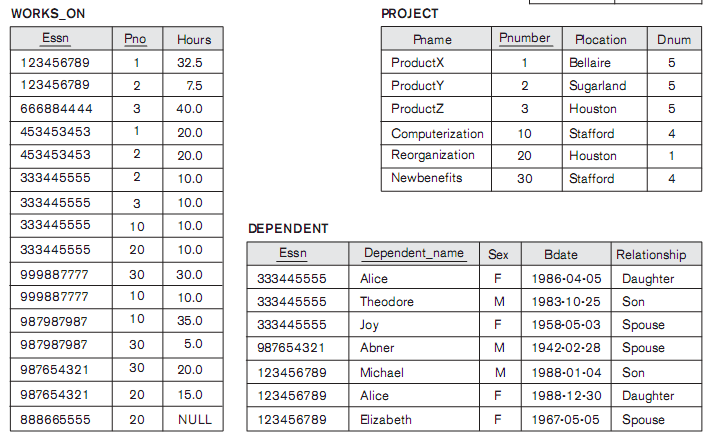
Câu 9 Cho view DEPT\_SUMMARY được định nghĩa trên CSDL một công ty như sau.



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.







Trả lời:

a. Câu query:

SELECT \*

FROM DEPT\_SUMMARY;

=> Câu query trên được cho phép thực hiện trên view. Câu query tương ứng:

SELECT Dno AS D, COUNT(\*) AS C, SUM(Salary) AS Total\_s, AVG(Salary) AS Average\_s

FROM Employee

GROUP BY Dno;

b. Câu query:

SELECT D, C

FROM DEPT\_SUMMARY

WHERE TOTAL\_S > 100000;

=> Câu query trên được cho phép thực hiện trên view. Câu query tương ứng:

SELECT tmp.D AS D, tmp.C AS C

FROM

    (SELECT Dno AS D, COUNT(\*) AS C, SUM(Salary) AS Total\_s

    FROM Employee

    GROUP BY Dno) AS tmp

WHERE tmp.TOTAL\_S > 100000;

c. Câu query:

SELECT D, AVERAGE\_S

FROM DEPT\_SUMMARY

WHERE C > (SELECT C FROM DEPT\_SUMMARY WHERE D = 0);

=> Câu query trên được cho phép thực hiện trên view. Câu query tương ứng:

WITH TmpTable AS

(

    SELECT  Dno AS D,

            COUNT(\*) AS C,

            SUM(Salary) AS Total\_s,

            AVG(Salary) AS Average\_s

    FROM Employee

    GROUP BY Dno

)

SELECT  TmpTable.D AS D,

        TmpTable.Average\_s AS AVERAGE\_S

FROM TmpTable

WHERE TmpTable.C > (SELECT TmpTable.C

                    FROM TmpTable

                    WHERE TmpTable.D = 0);

d. Câu lệnh cập nhật

UPDATE DEPT\_SUMMARY

SET D = 3

WHERE D = 4;

e. Câu lệnh xóa:

DELETE FROM DEPT\_SUMMARY

WHERE C > 4;