Assignment

**Exercise 1**. What is a foreign key constraint? Why are such constraints important? What is referential integrity?  
Ràng buộc khóa ngoài (**a foreign key constrain**)(còn được gọi là ràng buộc tham chiếu hoặc ràng buộc toàn vẹn tham chiếu ) : Các ràng buộc FOREIGN KEY thực thi tính toàn vẹn tham chiếu ( referential integrity ), về cơ bản nói rằng nếu giá trị cột A tham chiếu đến giá trị cột B thì giá trị cột B phải tồn tại hoặc A là null.

Các ràng buộc (constraints) quan trọng vì chúng cho phép người thiết kế chỉ định ngữ nghĩa của dữ liệu trong cơ sở dữ liệu. Ràng buộc là các quy tắc buộc DBMS phải kiểm tra xem dữ liệu có thỏa mãn ngữ nghĩa hay không.

Tính toàn vẹn tham chiếu (**referential integrity**)l  yêu cầu khóa ngoại phải có khóa chính phù hợp hoặc nó phải là null. Ràng buộc này được chỉ định giữa hai bảng (cha và con); nó duy trì sự tương ứng giữa các hàng trong các bảng này. Nó có nghĩa là tham chiếu từ một hàng trong bảng này sang bảng khác phải hợp lệ.

**Exercise 2.** Answer each of the following questions briefly. The questions are based on the following relational schema:

Emp(eid: integer, ename: string, age: integer, salary: real)  
Works(eid: integer, did: integer, pct\_time: integer)  
Dept(did: integer, *dname*: string, budget: real, managerid: integer)

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

- Một số ví dụ về ràng buộc khoá ngoại liên quan đến mối quan hệ Dept:

* Các giá trị trong cột did phải có ràng buộc not null và unique
* Các giá trị trong cột did của relation Work phải có trong cột did của relation Dept
* Managerid trong relation Dept phải có trong cột eid của relation Emp hoặc là bằng null.

- Các tùy chọn để thực thi ràng buộc này là gì khi người dùng cố gắng xóa một bộ dữ liệu Phòng?

* The Default Policy: từ chối xoá một bộ dữ liệu Dept mà có did được tham chiếu bởi khoá ngoại did của relation Works.
* The Cascade Policy: Theo chính sách này, các thay đổi đối với (các) thuộc tính did ( thuộc tính được tham chiếu ) sẽ được mô phỏng theo khoá ngoại ( khoá ngoại did của relation Work sẽ được set là null ở những bộ bị xoá có did tương ứng )
* The Set-Null Policy: set null ở khoá ngoại did của relation Works .

2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.

CREATE TABLE Emp (

eid INTEGER PRIMARY KEY,

ename STRING,

age INTEGER,

salary REAL

);

CREATE TABLE Works (

eid INTEGER,

did INTEGER,

pct\_time INTEGER,

PRIMARY KEY (eid, did),

FOREIGN KEY (eid) REFERENCES Emp(eid),

FOREIGN KEY (did) REFERENCES Dept(did)

);

CREATE TABLE Dept (

did INTEGER PRIMARY KEY,

dname STRING,

budget REAL,

managerid INTEGER,

FOREIGN KEY (managerid) REFERENCES Emp(eid)

);

3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.

ALTER TABLE Dept

ADD CONSTRAINT chk\_manager\_not\_null

CHECK (managerid IS NOT NULL);

4. Write an SQL statement to add John Doe as an employee with *eid* = 101, *age* = 32 and *salary* = 15*,* 000

insert into Emp values(101,'John Doe',32,15000)

5. Write an SQL statement to give every employee a 10 percent raise

update Emp

set salary = salary \* 1.1;

6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

delete from Dept

where dname = 'Toy';

trong relation Works , tôi đã có một ràng buộc FOREIGN KEY (did) REFERENCES Dept(did) thì khi xoá tuple có dname là 'Toy' , công việc ở quan hệ Works có tham chiếu qua Dept có dname ='Toy' sẽ bị conflict. ( ràng buộc tham chiếu bị xung đột )

**Exercise 3.** Consider the following relations:



1. List all the foreign key constraints among these relations.

Các ràng buộc tham chiếu trong relation:

+ Enrolled:

* Enrolled ( sid) là khoá ngoại tham chiếu đến Students( sid)
* Enrolled ( cid) là khoá ngoại tham chiếu đến Courses( cid)

+Teaches:

* Teaches (fid) là khoá ngoại tham chiếu đến Faculty(fid)
* Teaches (cid) là khoá ngoại tham chiếu đến Courses(cid)

+ Meets\_In:

* Meets\_In (cid) là khoá ngoại tham chiếu đến Course(cid)
* Meets\_In (rno) là khoá ngoại tham chiếu đến Rooms(rno)

2. Give an example of a (plausible) constraint involving one or more of these relations that is not a primary key or foreign key constraint.

+ sid và cid trong relation Enrolled is not null

+cid và rno trong relation Meets\_In is not null

đảm bảo rằng cột **grade** trong bảng **Enrolled** chỉ chứa các giá trị hợp lệ như 'A', 'B', 'C', 'D',:

ALTER TABLE Enrolled

ADD CONSTRAINT check\_max\_courses

CHECK (grade IN ('A', 'B', 'C', 'D', 'F'));

3. Create the above database with their constraints.

create table Students (

sid char(100) primary key,

name varchar(100) not null,

login char(100) unique,

age int,

gpa real

);

go

create table Faculty(

fid char(100) primary key,

fname varchar(100) not null,

sal real,

);

go

create table Courses(

cid char(100) primary key,

cname varchar(100) not null,

creadits int ,

);

go

create table Rooms (

rno int primary key,

address varchar(300) not null,

capacity int,

);

go

CREATE TABLE Enrolled (

sid CHAR(100),

cid CHAR(100),

grade VARCHAR(2),

PRIMARY KEY (sid, cid),

FOREIGN KEY (sid) REFERENCES Students(sid),

FOREIGN KEY (cid) REFERENCES Courses(cid),

);

create table Teaches(

fid char(100),

cid char(100),

primary key ( fid, cid),

foreign key (fid ) references Faculty ( fid),

foreign key (cid ) references Courses ( cid),

);

go

create table Meets\_In(

cid char(100),

rno int,

time char(100),

primary key (cid, rno,time),

foreign key (cid) references Courses (cid),

foreign key ( rno ) references Rooms (rno)

);

Go

ALTER TABLE Enrolled

ADD CONSTRAINT check\_max\_courses

CHECK (grade IN ('A', 'B', 'C', 'D', 'F'));

**Exercise 4.** Consider the following relational schema and briefly answer the questions that follow:

Emp(eid: integer, ename: string, age: integer, salary: real)  
Works(eid: integer, did: integer, pct\_time: integer)  
Dept(did: integer, budget: real, managerid: integer)

1. Define a constraint on Emp that will ensure that every employee makes at least $10,000.

ALTER TABLE Emp

ADD CONSTRAINT check\_salary\_min

CHECK (salary >= 10000);

1. Define a constraint on Dept (using trigger) that will ensure that all managers have age > 30

CREATE TRIGGER CheckManagerAge

ON Dept

AFTER INSERT, UPDATE

AS

BEGIN

-- Check if any new manager's age is less than or equal to 30

IF EXISTS (

SELECT 1

FROM inserted i INNER JOIN Emp e ON i.managerid = e.eid

WHERE e.age <= 30

)

BEGIN

RAISERROR('Managers must be older than 30 years.', 16, 1);

ROLLBACK TRANSACTION; -- Rollback the transaction to prevent the change

END

END;

go

1. Write SQL statements to delete all information about employees whose salaries exceed that of the manager of one or more departments that they work in. Be sure to ensure that all the relevant integrity constraints are satisfied after your updates.

**Exercise 5.** Discuss the strengths and weaknesses of the trigger mechanism. Contrast triggers with other integrity constraints supported by SQL.

**Exercise 6.** Consider the following relational schema. An employee can work in more than one department; the pct\_time ﬁeld of the Works relation shows the percentage of time that a given employee works in a given department.

Emp(eid: integer, ename: string, age: integer, salary: real)

Works(eid: integer, did: integer, pct\_time: integer)

Dept(did: integer, budget: real, managerid: integer)

Write integrity constraints (domain, key, foreign key, or CHECK constraints; or triggers to ensure each of the following requirements, considered independently.

1. Employees must make a minimum salary of $1000.
2. Every manager must be also be an employee.
3. The total percentage of all appointments for an employee must be under 100%.
4. A manager must always have a higher salary than any employee that he or she

manages.

1. Whenever an employee is given a raise, the manager’s salary must be increased to

be at least as much.

1. Whenever an employee is given a raise, the manager’s salary must be increased

to be at least as much. Further, whenever an employee is given a raise, the department’s budget must be increased to be greater than the sum of salaries of all employees in the department.

**Exercise 7.** Brieﬂy answer the following questions based on this schema:

Emp(eid: integer, ename: string, age: integer, salary: real)

Works(eid: integer, did: integer, pct time: integer)

Dept(did: integer, budget: real, managerid: integer)

1. Suppose you have a view SeniorEmp deﬁned as follows:

CREATE VIEW SeniorEmp (sname, sage, salary)

AS SELECT E.ename, E.age, E.salary

FROM Emp E

WHERE E.age > 50

Explain what the system will do to process the following query:

SELECT S.sname

FROM SeniorEmp S

WHERE S.salary > 100,000

2. Give an example of a view on Emp that could be automatically updated by updating Emp.

3. Give an example of a view on Emp that would be impossible to update (automatically) and explain why your example presents the update problem that it does.