Assignment

**Exercise 1**. What is a foreign key constraint? Why are such constraints important? What is referential integrity?

Ràng buộc khóa ngoại (ràng buộc tham chiếu): Các giá trị trong cột khóa ngoại ở bảng tham chiếu phải có trong cột tương ứng ở bảng được tham chiếu.

Nó quan trọng bởi vì nó duy trì tính toàn vẹn tham chiếu

Tính toàn vẹn tham chiếu (referential integrity) là sự phụ thuộc của khóa ngoại vào khóa chính

**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?

managerid là khóa ngoại tham chiếu đến eid của Emp

did là khóa ngoại của bảng Works tham chiếu đến did của bảng Dept

Có thể có các lựa chọn :

+Ràng buộc khóa ngoại

+Tạo một trigger kiểm tra trước khi thực hiện thao tác xóa

1. Write the SQL statements required to create the preceding relations(các quan hệ được xác định trước đó), including appropriate versions of all primary and foreign key integrity constraints(ràng buộc toàn vẹn).

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

CREATE TABLE Emp (

eid int PRIMARY KEY,

ename nvarchar(100),

age int,

salary real

);

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

CREATE TABLE Works (

eid int,

did int,

pct\_time int,

FOREIGN KEY (eid) REFERENCES Emp(eid)

FOREIGN KEY (did) REFERENCES Dept(did)

);

Dept(did: integer, *dname*: string, budget: real, managerid: integer)

CREATE TABLE Dept (

did int PRIMARY KEY,

dname nvarchar(100),

budget real,

managerid int,

FOREIGN KEY (managerid) REFERENCES Emp(eid)

);

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

CREATE TABLE Dept (

did int PRIMARY KEY,

dname nvarchar(100),

budget real,

managerid int,

FOREIGN KEY (managerid) REFERENCES Emp(eid)

);

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

INSERT INTO Emp(eid,age,salary)

VALUES (101,32,15000);

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

UPDATE Emp

Set salary = salary + salary \*1,1;

1. 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’;

Khóa ngoại did của Works tham chiếu đến đến khóa chính did trong Dept nếu thực hiện câu lệnh xóa , hệ quản trị sẽ không cho xóa

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



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

FOREIGN KEY Enrolled(sid) REFERENCES Students(sid)

FOREIGN KEY Enrolled(cid) REFERENCES Courses(cid)

FOREIGN KEY Teaches(fid) REFERENCES Faculty(fid)

FOREIGN KEY Teaches(cid) REFERENCES Courses(cid)

FOREIGN KEY Meets\_In(cid) REFERENCES Courses(cid)

FOREIGN KEY Meets\_In(rno) REFERENCES Rooms(rno)

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

Student(Name,Age,Gpa) NOTNULL

Student(Login) UNIQUE

Faculty(fname,sal) NOTNULL

Courses(cname,credit) NOT NULL

Rooms(address,capacity) NOTNULL

Enrolled(grade) NOT NULL

Meets\_In(time) NOT NULL

1. Create the above database with their constraints.

CREATE TABLE Students (

sid nvarchar(100) PRIMARY KEY,

name nvarchar(100),

login nvarchar(100) UNIQUE,

age int NOT NULL,

gpa real NOT NULL

);

CREATE TABLE Faculty (

fid nvarchar(100) PRIMARY KEY,

fname nvarchar(100) NOTNULL,

sal real NOT NULL

);

CREATE TABLE Courses (

cid nvarchar(100) PRIMARY KEY,

cname nvarchar(100) NOT NULL,

credits int NOT NULL

);

CREATE TABLE Rooms (

rno int PRIMARY KEY,

address nvarchar(100) NOT NULL,

capacity int NOT NULL

);

CREATE TABLE Enrolled (

sid nvarchar(100) ,

cid nvarchar(100) ,

grade nvarchar(100) NOT NULL,

FOREIGN KEY sid REFERENCES Students(sid)

FOREIGN KEY cid REFERENCES Courses(cid)

ON DELETE SET NULL

ON UPDATE CASCADE

);

CREATE TABLE Teaches(

fid nvarchar(100) ,

cid nvarchar(100),

FOREIGN KEY fid REFERENCES Faculty(fid)

FOREIGN KEY cid REFERENCES Courses(cid)

ON DELETE SET NULL

ON UPDATE CASCADE

);

CREATE TABLE Meets\_In(

cid nvarchar(100) ,

rno int ,

FOREIGN KEY cid REFERENCES Courses(cid)

FOREIGN KEY rno REFERENCES Rooms(rno)

ON DELETE SET NULL

ON UPDATE CASCADE

)

**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.

CONSTRAINT RightSalary CHECK(salary >=10000)

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

CONSTRAINT RightAge CHECK(age>30)

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.

DELETE FROM Emp WHERE eid IN (SELECT eid FROM Works,Dept WHERE Works.did=Dept.did AND Works.eid>Dept.managerid);

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

Trigger là 1 produce được thực thi tự động để trả lời lại những thay đổi của database . Sự thuận lợi của cơ chế trigger bao gồm khả năng thực thi các hành động dựa vào kết quả của điều kiện truy vấn.Những ràng buộc toàn vẹn có thế thực thi các hành động này.Hành động này bao gồm UPDATE,DELETE,INSERT,viết câu lệnh để tạo bảng mới hoặc view mới , thay đổi chỉnh sách bảo mật.Trigger có thể được thực thi trước hoặc sau khi database thay đổi ( có thể sử dụng lại dữ liệu cũ hoặc dữ liệu mới kiểu như những tuple bị xóa hoặc thêm sẽ được lưu trong “deleted” hoặc “inserted”)

Nhưng cũng có một số bất lợi. Có thể gây ra sự phức tạp khi cố gắng hòa hợp giữa sự truy cập database và trigger events.Bên cạnh đó ràng buộc toàn vẹn được tích hợp vào việc tối ưu hóa database , nó sẽ khó khăn cho một database tự động tối ưu với trigger . Nếu tính nhất quán của database là mục tiêu chính thì ràng buộc toàn vẹn có vai trò ngang với triggers.Các ràng buộc toàn vẹn thường dễ hiểu hơn trigger.

**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.

CONSTRAINT RightSalary CHECK(salary >=1000)

1. Every manager must be also be an employee.

FOREIGN KEY Dept(managerid) REFERENCES Emp(eid)

1. The total percentage of all appointments for an employee must be under 100%.

CONSTRAINT Rightpct\_time CHECK (pct<100)

1. A manager must always have a higher salary than any employee that he or she

manages.

Create trigger RightManagerSalary on Dept

for insert

as

declare @managerid

select @managerid=ne.managerid

from inserted ne

if(SELECT Salary FROM Emp WHERE @mangerid = eid) < (

SELECT Salary FROM Emp

WHERE eid IN ( SELECT eid from (select eid,managerid from Works,Dept where Works.did=Dept.did) Where managerid = @managerid)

Begin

ROLLBACK

End

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.

Create trigger raiseSalary on Emp

for UPDATE as

declare @eid int, @raiseSalary int, @sumSalary real , @budget real

select @eid = ne.eid,@raiseSalary = ne.salary-ol.salary

from inserted ne, deleted ol

where ne.eid = ol.eid

begin

UPDATE Emp

SET salary = salary + @raiseSalary

Where eid in ( select eid from Works,Dept where Works.did=Dept.did and managerid = @eid )

SELECT @sumSalary = sum(salary)

From Emp where eid IN (SELECT eid FROM Works WHERE did IN (SELECT did FROM Works WHERE eid = @eid))

SELECT @budget=budget from Dept where did in (SELECT did FROM Works WHERE eid = @eid)

If( @budget < @sumSalary )

Begin

Update Dept

SET budget=budget+@sumsalary

Where did in (SELECT did FROM Works WHERE eid = @eid)

End

End

Lấy địa chỉ bộ phận của nhân viên được tăng lương

SELECT did FROM Works WHERE eid = @eid

Lấy toàn bộ nhân viên của bộ phận đó

SELECT eid FROM Works

WHERE did IN (SELECT did FROM Works WHERE eid = @eid)

Nhân viên -> bộ phận -> tổng lương tất cả nhân viên trong bộ phận đó

**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

Liệt kê tên của các SeniorEmp có lương lớn hơn 100000

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

CREATE VIEW SeniorEmp (sname, sage, salary)

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

FROM Emp E

WHERE E.age > 50

1. 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.

CREATE VIEW AvgAgeNumberOfEmp AS

SELECT AVG(age) from Emp

Không vì những view không được cập nhật là những VIEW :

view được định nghĩa bởi sử dụng GROUP và các hàm tính toán

view được định nghĩa bởi gộp nhiều bảng