Assignment

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

* Ràng buộc khóa ngoại
* Ràng buộc khóa ngoại: một hay nhiều thuộc tính của một quan hệ này có thể được tham chiếu sang một thuộc tính của một quan hệ khác.
* Các thuộc tính được tham chiếu phải đảm bảo ràng buộc khóa chính hoặc not null
* Tại sao những ràng buộc như vậy quan trọng?

Vì nó sẽ đảm bảo tính toàn vẹn và đúng đắn của toàn bộ dữ liệu trong csdl

* Tính toàn vẹn tham chiếu:

Nó bao gồm các trường hợp thêm xóa sửa dữ liệu ở các thuộc tính đã tham chiếu cho nhau giữa hai bảng.

**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?

Được xóa mà không cần option

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

ALTER TABLE Works ADD CONSTRAINT eidFrKEY FOREIGN KEY (eid) REFERENCES Emp(eid)

ALTER TABLE Works ADD CONSTRAINT didFrKEY FOREIGN KEY (did)

REFERENCES Dept(did)

ALTER TABLE Dept ADD CONSTRAINT mnidFrKEY FOREIGN KEY (managerid) REFERENCES Emp (eid)

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

ALTER TABLE Dept ALTER COLUMN managerid INT NOT NULL

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\*0.1)

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’

Khi xóa bộ phận Toy thì id của bộ phận này cũng bị xóa, trong khi đó id của bộ phận này lại được tham chiếu để làm khóa ngoại ở bên bảng Works, cho nên khi xóa sẽ có 3 phương án:

* Không được xóa
* Set null.

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



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

Student(sid) – Enrolled(sid)

Courses(cid) – Enrolled(cid)

Courses(cid) – Teaches(cid)

Courses(cid) – Meets\_In(cid)

Rooms(rno) – Meets\_In(rno)

Faculty(fid) – Teaches(fid)

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.

Vidu: Meets\_In (time) not null

1. Create the above database with their constraints.

create Database manageSchool

go

create table dbo.Student

(

sid nvarchar primary key,

name nvarchar,

login nvarchar,

age int,

gpa float

)

create table dbo.Faculty

(

fid nvarchar primary key,

fname nvarchar,

sal float

)

create table dbo.Course

(

cid nvarchar primary key,

cname nvarchar,

credits int

)

create table dbo.Rooms

(

rno int primary key,

address nvarchar,

capacity int

)

create table dbo.Enrolled

(

sid nvarchar references Student(sid),

cid nvarchar references Course(cid),

grade nvarchar

primary key (sid, cid)

)

create table dbo.Teaches

(

fid nvarchar references Faculty(fid),

cid nvarchar references Course(cid)

primary key (fid, cid)

)

create table dbo.Meets\_In

(

cid nvarchar references Course(cid),

rno int references Rooms(rno),

time nvarchar

primary key(cid, rno, time)

)

**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 CONSTRAINT minsal CHECK(salary >= 10000)

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

CREATE OR REPLACE FUNCTION check\_manager\_age()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.managerid IS NOT NULL THEN

-- Kiểm tra tuổi của trưởng phòng mới

SELECT INTO age

FROM Emp

WHERE eid = NEW.managerid;

IF age <= 30 THEN

-- Nếu tuổi của trưởng phòng mới không đủ lớn, không cho phép thêm

RAISE EXCEPTION 'Manager must be over 30 years old.';

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER manager\_age\_trigger

BEFORE INSERT OR UPDATE ON Dept

FOR EACH ROW

EXECUTE FUNCTION check\_manager\_age();

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.

\*\*Điểm mạnh của trigger:\*\*

1. Tự động hóa các thao tác: Trigger cho phép tự động hóa các hành động phức tạp, như phân phối dữ liệu, thay đổi bản ghi trong cùng bảng và đồng bộ hóa dữ liệu giữa các bảng khác nhau.

2. Phát hiện và cảnh báo lỗi: Trigger có thể được sử dụng để phát hiện và cảnh báo về lỗi và các thay đổi không mong muốn trong dữ liệu, đưa ra các hành động cần thiết để giữ các dữ liệu được liên quan được đồng bộ.

3. Xử lý các ràng buộc và phân quyền: Trigger có thể được sử dụng để kiểm tra các ràng buộc và thực hiện các hành động phân quyền để bảo đảm tính toàn vẹn của dữ liệu.

4. Tăng tính tương thích: Trigger cho phép giảm sự phụ thuộc đối với ứng dụng và nâng cao tính tương thích với các phiên bản cũ hơn của phần mềm.

\*\*Điểm yếu của trigger:\*\*

1. Tốc độ thực thi: Trigger có thể làm chậm quá trình thêm, sửa hoặc xóa dữ liệu, tăng thời gian phản hồi của hệ thống.

2. Khó quản lý: Trigger có thể gây khó khăn trong việc quản lý và bảo trì, đặc biệt là trong các hệ thống có nhiều trigger.

3. Khó khăn trong khắc phục sự cố: Trigger có thể gây ra sự cố khó khăn để xử lý và khắc phục, đặc biệt là khi có các trigger tương tác hoặc gây ra sự phụ thuộc lồng nhau.

1. Độ phức tạp: Trigger có thể rất phức tạp và yêu cầu kiến thức chuyên môn cao để thiết kế và triển khai một cách hiệu quả.

**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.

ALTER TABLE Emp ADD CONSTRAINT checkSal CHECK(salary >= 1000)

1. Every manager must be also be an employee.

ALTER TABLE Dept ADD CONSTRAINT checkEmp FOREIGN KEY ( managerid ) REFERENCES Emp (eid)

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

CREATE OR REPLACE FUNCTION check\_total\_percentage()

RETURNS TRIGGER AS $$

BEGIN

-- Declare variables

DECLARE total\_pct INTEGER;

-- Calculate the total percentage of appointments for the employee

SELECT SUM(pct\_time) INTO total\_pct

FROM Works

WHERE eid = NEW.eid;

-- Check if the total percentage exceeds 100

IF total\_pct > 100 THEN

-- Raise an exception to prevent the insertion/update

RAISE EXCEPTION 'Total percentage exceeds 100%% for employee %', NEW.eid;

END IF;

-- If the total percentage is within limits, allow the insertion/update

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER enforce\_total\_percentage\_constraint

BEFORE INSERT OR UPDATE ON Works

FOR EACH ROW

EXECUTE FUNCTION check\_total\_percentage();

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

manages.

CREATE TRIGGER checkSal ON Dept

FOR INSERT, UPDATE

AS

BEGIN

DECLARE @newmnid INT

SELECT @newmnid = ne.managerid

FROM INSERTED ne

IF @newmnid NOT IN ( SELECT eid FROM Emp )

BEGIN

RAISERROR ('không tìm thấy quản lý này',16,10)

ROLLBACK TRAN;

RETURN;

END

SELECT did, MIN(eid) AS eid INTO ED

FROM Works

GROUP BY did;

SELECT managerid, eid INTO MD

FROM ED INNER JOIN Dept ON ED.did = Dept.did

SELECT managerid, Emp.eid, Emp.salary INTO SD

FROM MD INNER JOIN Emp ON MD.eid = Emp.eid

DECLARE @max REAL

SELECT @max = MAX(salary)

FROM SD

DECLARE @mnSal INT

SELECT @mnSal = salary

FROM Emp

WHERE eid = @newmnid

IF @mnSal < @max

BEGIN

RAISERROR ('lương quản lý phải lớn hơn nhân viên',16,10)

ROLLBACK TRAN;

END

END

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

be at least as much.

CREATE TRIGGER checkRAIS ON Emp

AFTER UPDATE

AS

BEGIN

DECLARE @newSaL INT, @oldSal INT

SELECT @newSal = ne.salary, @oldSal = ol.salary

FROM inserted ne, inserted ol

DECLARE @addition INT

SELECT @addition = @newSaL - @oldSal

DECLARE @ID INT;

DECLARE readf CURSOR FOR

SELECT managerid FROM Dept;

OPEN readf;

FETCH NEXT FROM readf INTO @ID;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Xử lý giá trị @value ở đây

FETCH NEXT FROM cursor\_name INTO @ID;

UPDATE Emp SET salary = salary + @addition

WHERE Emp.eid = @ID

END

CLOSE readf;

DEALLOCATE readf;

END

GO

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

Hệ thống sẽ thực hiện truy vấn trên view SeniorEmp để tìm người nhân viên có mức lương lớn hơn 100,000

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

CREATE VIEW YoungEmployees AS

SELECT \*

FROM Emp

WHERE age < 30;

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.

CREATE VIEW AvgSalary AS

SELECT AVG(salary) AS average\_salary

FROM Emp;

Chương 2.

1. Ràng buộc toàn vẹn là gì?

Là các ràng buộc mà toàn bộ dữ liệu trong cơ sở dữ liệu phải tuân theo để đảm bảo tính toàn vẹn và đúng đắn của dữ liệu

1. Tại sao các quan hệ phải có ràng buộc toàn vẹn?

Để đảm bảo tính đúng đắn và toàn vẹn của dữ liệu

1. Khóa chính (primary key) của một quan hệ Q

Là một khóa ứng viên được chọn từ tập các khóa ứng viên của Q, có thể xác 1 bộ duy nhất

1. Khóa ngoại (foreign key) của một quan hệ Q

Là một thuộc tính được tham chiếu từ một khóa chính hay một thuộc tính đảm bảo unique từ một quan hệ khác

1. Ràng buộc về miền giá trị của 1 thuộc tính

Là các giá trị của một thuộc tính phải đảm bảo nằm trong một miền xác định

1. Một thuộc tính có ràng buộc Unique?

Là các giá trị trong một thuộc tính phải đảm bảo là duy nhất và không được trùng nhau

1. Một thuộc tính có ràng buộc not null nghĩa là?

Là một thuộc tính không được có giá trị null

1. Một thuộc tính có ràng buộc Check (điều kiện) nghĩa là?

Là một thuộc tính có bộ giá trị đảm bảo được điều kiện Check được thiết lập trước đó

1. Câu lệnh khai báo, thêm và xóa các ràng buộc của quan hệ Q.

* Khi không cần khai báo tên cho ràng buộc

[thuộc tính] [kiểu ràng buộc]

* Khi cần gán tên cho ràng buộc

[tên – kiểu giá trị thuộc tính] CONSTRAINT [tên ràng buộc] [kiểu ràng buộc]

CONSTRAINT [tên ràng buộc] [kiểu ràng buộc]

Xóa – thêm

**ALTER TABLE [tên qh]** **DROP CONSTRAINT** [tên ràng buộc];

**ALTER TABLE** [tên qh] **ADD CONSTRAINT** [tên ràng buộc]

[kiểu ràng buộc]

1. Cho Q1(A int, B int), Q2(A int, C int). A trong Q2 là khóa ngoại tham chiếu qua A của Q1. Thao tác nào dưới đây không thực hiện được
2. Thêm 1 dòng có A = 1 vào Q2, nhưng trong Q1 chưa có dòng nào có A = 1.
3. Sửa 1 dòng trong Q2 có A = 1 thành A=0, nhưng trong Q1 chưa có dòng nào có A = 0.
4. Cả (1) và (2) đều đúng
5. Cả (1) và (2) đều sai
6. Trigger là gì?

Trigger là một chương trình con sẽ tự động được kích hoạt khi có biến cố trong CSDL

1. Ý nghĩa của biến cố, điều kiện và hành động trong trigger

Trigger sẽ được gọi khi có biến cố xảy ra do người lập trình chỉ định. Sau khi được kích hoạt, trigger sẽ kiểm tra điều kiện, nếu thỏa mã sẽ thực hiện các hành động được định sẵn trong trigger.

1. Ý nghĩa của For và After trong câu lệnh tạo trigger

For: ngay khi xảy ra biến cố thì mới thực hiện trigger

Alter: sau khi xảy ra biến cố thì mới thực hiện trigger

1. Ý nghĩa của câu lệnh dưới

Create trigger AA on Q after delete as declare @new int, @old int select @old=ol.B from deleted ol if (@old >2) print ('Notify Customer Relations');

Tạo một trigger AA trong bảng Q, sẽ được kích hoạt sau biến cố delete, sau khi xóa nếu giá trị cũ của thuộc tính B mà < 2 thì thông báo ('Notify Customer Relations');

1. Instead of trigger là gì?

Là một loại trigger đặc biệt, nó cho phép bạn bỏ qua câu lệnh INSERT, DELETE, UPDATE, trên một table hoặc view.

1. Ý nghĩa của câu lệnh dưới

Create trigger AA on Q after delete as declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations'); Rollback; end

Tạo một trigger AA trong bảng Q , được kích hoạt sau khi xảy ra biến cố delete. Khai báo biến @old để lưu giá trị vừa được xóa từ cột B của bảng tạm ol. Nếu giá trị này > 2 thì hoàn tác, thông báo và dừng chương trình

1. Cho

Create trigger AA on Q after delete as declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations'); Rollback; end

Từ khóa “after” trong câu lệnh trên có nghĩa là gì?

Sau khi biến cố delete xảy ra thì trigger mới hoạt động

1. Cho

Create trigger AA on Q for delete as declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations');

Từ khóa “for” trong câu lệnh trên có nghĩa là gì?

Ngay khi xảy ra biến cố delete thì trigger được kích hoạt

1. Ý nghĩa của câu lệnh dưới

Create trigger BB on Q1 instead of insert as Declare @newA int, @newB int Select @newA = n.A, @newB = n.B From Inserted n Insert into Q2 values(@newA, @newB)

Tạo một trigger BB trên bảng Q1, khi biến cố insert xảy ra trên Q1 thì sẽ bỏ qua để thực hiện các thao tác khác: gán giá trị được thêm vào cột A vào biến @newA, gán giá trị được thêm vào cột B vào biến @newB, sau đó đưa giá trị của 2 biến này vào bảng Q2

1. View là gì?

View giúp người ta tạo ra các bảng ảo ( Virtual table) chứa các thông tin, dữ liệu được lưu trữ sẵn cho người dùng sử dụng. Tuy nhiên, View không tồn tại như một cấu trúc lưu trữ dữ liệu trong csdl.

1. Mục đích của sử dụng view

* Để tập trung trên dữ liệu được xác định
* Để đơn giản hóa thao tác dữ liệu.
* Để tùy biến dữ liệu
* Để trích xuất dữ liệu tạo báo cáo
* Để bảo mật dữ liệu che giấu thông tin

1. Trong SQL server, cú pháp câu lệnh tạo, xóa, sửa view?

Tạo view

CREATE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition;

Lệnh xóa view

DROP VIEW<name>

Lệnh sửa view

ALTER VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition

1. Phân loại View?

* Standard view
* Indexed view
* Partitioned view

1. Ý nghĩa của câu lệnh

Create view cc as Select A, max(B) as ln from Q1 group by A;

Tạo một view cc bằng cách chọn cột A, giá trị max tại cột B với bí danh ln, được gom nhóm theo A.

1. Cho PBan(MaPB int, TenPB char(20), MaNQL int). Viết câu lệnh thêm ràng buộc đảm bảo mỗi phòng ban có 1 người quản lý

**ALTER TABLE Pban.MaNQL** **ADD CONSTRAINT** MaNQLunique **UNIQUE**

1. Cho NV(MaNV int, Hoten char(30), Luong int) và câu lệnh alter table Emp add constraint chk\_luong\_min check (salary > 10000). Giải thích ý nghĩa của câu lệnh này.

Câu lệnh trên thêm ràng buộc là các giá trị trong cột salary phải trên 10000

1. Cho NV(MaNV int, Hoten char(30), Luong int, MaPB int) và view được định nghĩa như sau Create view V\_NV(MaPB, SoNV, TongLg, LgTB) as Select MaPB, Count(\*), Sum(Luong), Avg(Luong) From NV Group by MaPB;

Các câu SQL nào thực hiện được trên view V\_NV

1. Select \* From V\_NV
2. Delete from V\_NV Where MaPB = 1
3. Cả (1) và (2) đều đúng
4. Cả (1) và (2) đều sai