Chương 2.

1. Ràng buộc toàn vẹn là gì?

Tập các quy tắc mà mọi dữ liệu trong CSDL phải tuân theo nhằm đảm bảo tính toàn vẹn của cơ sở dữ liệu

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

Để đảm bảo tính toàn vẹn của cơ sở dữ liệu

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

Các giá trị của khóa chính phải là UNIQUE và NOTNULL

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

Các giá trị trong cột khóa ngoại của bảng tham chiếu phải có trong cột tương ứng của bảng được tham chiếu

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

Các giá trị có trong cột phải nằm trong miền giá trị của cột

1. Một thuộc tính có ràng buộc Unique?

Gía trị của mỗi tuple của thuộc tính đó phải khác nhau

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

Gía trị của thuộc tính đó không được null

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

Khi insert , dữ liệu vào phải thỏa điều kiện Check của thuộc tính đó mới insert dữ liệu được

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

+khai báo :

CREATE TABLE MovieStar (

name CHAR(30) CONSTRAINT *NamelsKey* PRIMARY KEY,

address VARCHAR(255),

gender CHAR(1),

birthdate DATE,

**CONSTRAINT *RightTitle***

CHECK (gender = ’F’ OR name NOT LIKE ’Ms.%’ )

);

+Thêm :

ALTER TABLE name\_table ADD CONSTRAINT tên\_ràng\_buộc ĐiềuKiệnRàngBuộc(tên thuộc tính)

+Xóa

ALTER TABLE name\_table ADD CONSTRAINT tên\_ràng\_buộc ĐiềuKiệnRàngBuộc(tên thuộc tính)

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 chuỗi các hoạt động có liên quan đến các sự kiện nhất định và Trigger được thực hiện bất cứ khi nào các sự kiện này thực thi

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

Biến cố(certain event)

Trigger thực thi chỉ khi các biến cố đặc biệt xảy ra bởi các chương trình của database ví dụ như insert , delete , update

Nếu điều kiện của trigger thỏa , những hoạt động có liên quan đến trigger sẽ được thực thi bởi hệ quản trị cơ sở dữ liệu

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

For biến cố : khi biến cố vừa xảy ra thì các hành động trong trigger sẽ thực thi

After biến cố : khi biến cổ xảy ra xong thì các hành động trong trigger sẽ được thực thi

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 ở bảng Q , sau khi câu lệnh delete thực thi xong trên bảng Q , các hành động liên quan đến trigger sẽ được thực thi :

-Khai báo biến @new và @old kiểu int

chúng ta sử dụng câu lệnh SELECT để lấy giá trị của cột B từ bảng tạm thời "Deleted" có tên là ol , đây là một bảng ảo chứa các bản ghi mới được được xóa từ bảng Q , gán vào biến @old

-Nếu giá trị @old lớn hơn 2 thì in ra view Notify Customer Relations

1. Instead of trigger là gì?

Là một loại trigger đặc biệt cho phép bạn bỏ qua câu lệnh INSERT , UPDATE hoặc DELETE trên một bảng 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 ở bảng Q , sau khi câu lệnh delete thực thi xong trên bảng Q , các hành động liên quan đến trigger sẽ được thực thi :

-Khai báo biến @new và @old kiểu int

chúng ta sử dụng câu lệnh SELECT để lấy giá trị của cột B từ bảng tạm thời "Deleted" có tên là ol , đây là một bảng ảo chứa các bản ghi mới được được xóa từ bảng Q , gán vào biến @old

-Nếu giá trị @old lớn hơn 2 thì in ra view Notify Customer Relations và hủy bỏ mọi thay đổi được thực hiện trong cơ sở dữ liệu quay về trạng thái trước khi câu lệnh delete bắt đầu

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 câu lệnh delete thực thi xong trên bảng Q , các hành động liên quan đến trigger sẽ được thực thi

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?

khi câu lệnh delete thực thi xong trên bảng Q , các hành động liên quan đến trigger sẽ được thực thi

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 ở bảng Q1, nếu câu lệnh insert được thực thi vào Q1 thì câu lệnh insert không được thực thi thay vào đó các hành động liên quan các hành động liên quan đến trigger sẽ được thực thi(trigger BB) :

-Khai báo biến @newA và @newB kiểu int

- Tiếp theo, chúng ta sử dụng câu lệnh SELECT để lấy giá trị của cột A và cột B từ bảng tạm thời "Inserted", đây là một bảng ảo chứa các bản ghi mới được chèn vào bảng Q1 , gán vào biến @newA và @newB

- Sau đó thực hiện insert vào bảng Q2 với @newA,@newB

1. View là gì?

View là một bảng ảo bao gồm hàng và cột được định nghĩa bởi những truy vấn dữ trên một hay nhiều bảng ( base tables ) hoặc những views khác trong database

1. Mục đích của sử dụng view

Tập trung , đơn giản hóa và điều chỉnh theo nhận thức của mỗi người dùng database

Một cơ chế bảo mật an toàn cho phép người dùng truy cập dữ liệu thông qua view

Không cho phép người dùng truy cập chính xác cơ sở dữ liệu bên dưới

Cung cấp tính năng backward compatible(tương thích ngược) để mô phỏng sự thay đổi của lược đồ cơ sở dữ liệu

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

Câu lệnh tạo :

CREATE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition;

Câu lệnh xóa

DROP VIEW <name>

Câu lệnh sửa view

ALTER VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition;

1. Phân loại View?

View được cập nhật và View không được cập nhật ( Un-updatable View)

View không được cập nhật là view được định nghĩa bởi sử dụng groups và các hàm tổng hợp ( aggregate function) sẽ không được update

View được định nghĩa bởi nhiều bảng bằng cách sử dụng joins cũng không được cập nhật

1. Ý nghĩa của câu lệnh

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

Tạo ra một view cc có cột thuộc tính A và ln ( giá trị của ln là giá trị max của thuộc tính B ) từ bảng Q1 , GROUP BY bởi thuộc tính 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 ADD CONSTRAINT RangBuocKhoaChinh MaPB PRIMARY KEY

ALTER TABLE Pban ADD CONSTRAINT RangBuocQuanLi MaNQL NOT NULL

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.

Thêm ràng buộc tên chk\_luong\_min vào bảng NV , ràng buộc có ý nghĩa giá trị của thuộc tính salary phải lớn hơn 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

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

ON DELETE SET NULL

ON UPDATE CASCADE

);

CREATE TABLE Faculty (

fid nvarchar(100) PRIMARY KEY,

fname nvarchar(100) NOTNULL,

sal real NOT NULL

ON DELETE SET NULL

ON UPDATE CASCADE

);

CREATE TABLE Courses (

cid nvarchar(100) PRIMARY KEY,

cname nvarchar(100) NOT NULL,

credits int NOT NULL

ON DELETE SET NULL

ON UPDATE CASCADE

);

CREATE TABLE Rooms (

rno int PRIMARY KEY,

address nvarchar(100) NOT NULL,

capacity int NOT NULL

ON DELETE SET NULL

ON UPDATE CASCADE

);

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)

);

CREATE TABLE Teaches(

fid nvarchar(100) ,

cid nvarchar(100),

FOREIGN KEY fid REFERENCES Faculty(fid),

FOREIGN KEY cid REFERENCES Courses(cid)

);

CREATE TABLE Meets\_In(

cid nvarchar(100) ,

rno int ,

FOREIGN KEY cid REFERENCES Courses(cid),

FOREIGN KEY rno REFERENCES Rooms(rno)

)

**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 int

select @managerid=ne.managerid

from inserted ne

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

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

Begin

ROLLBACK

End

-- danh sách công nhân và người chủ của công nhân đó

select eid,managerid from Works,Dept where Works.did=Dept.did) as A Where managerid = @managerid

--những người công nhân mà @managerid làm chủ

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

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

go

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