**Câu hỏi ôn tập chương 2**

1. Ràng buộc toàn vẹn là gì?

Ràng buộc toàn vẹn là 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?

- Bảo đám tính kết dính của các thành phần tạo nên CSDL

- Bảo đảm tính nhất quán của dữ liệu

- Bảo đảm CSDL luôn biểu diễn đúng ngữ nghĩa thực tế

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

Một trong các khóa ứng viên sẽ được chọn làm khóa chính cho quan hệ.

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

Một hay nhiều thuộc tính (khóa chính hay thuộc tính có ràng buộc unique) trong một quan hệ A xuất hiện trong một quan hệ B khác được gọi là khóa ngoại trong quan hệ B.

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

Ràng buộc về miền giá trị của 1 thuộc tính là các giá trị trong cột phải nằm trong miền giá trị của cột.

1. Một thuộc tính có ràng buộc Unique?

Một thuộc tính có ràng buộc Unique thì mỗi giá trị trong cột có ràng buộc này phải là duy nhất trong cột đó.

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

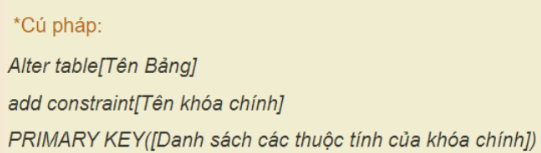
Ràng buộc not null sẽ không cho phép một cột chứa các giá trị NULL. Điều này sẽ buộc một trường luôn chứa ít nhất một giá trị, có nghĩa là bạn không thể chèn bản ghi mới hay cập nhật một bảng ghi mà không thêm một giá trị vào trường này (để trống)

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

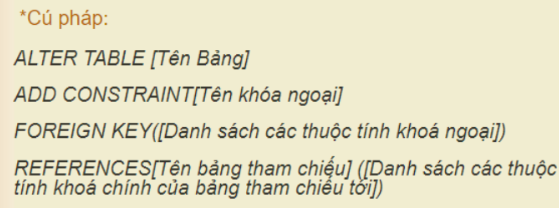
CHECK là ràng buộc miền giá trị, nghĩa là nó chỉ giá trị của bạn nhập vào chỉ nằm trong một phạm vi nhất định

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

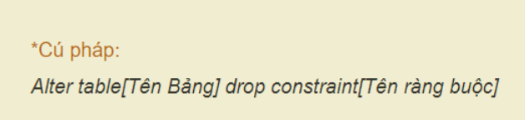
- Câu lệnh thêm ràng buộc khóa chính:



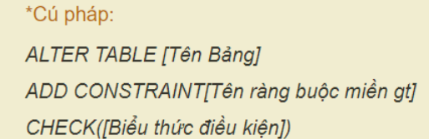
- Câu lệnh thêm ràng buộc khóa ngoại:



- Câu lệnh xóa ràng buộc:



- Câu lệnh thêm ràng buộc miền giá trị:



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 loạt các hành động có liên quan đến một số sự kiện nhất định và được thực hiện bất cứ khi nào những sự kiện này phát sinh.

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

Trigger chỉ được gọi khi xảy ra một số sự kiện nhất định, do người lập trình cơsở dữ liệu chỉ định(thường là chèn, xóa hoặc cập nhật). Sau khi được gọi bởi sự kiện kích hoạt của nó, trình kích hoạt sẽ kiểm tra một điều kiện.Nếu điều kiện của trình kích hoạt được thỏa mãn, thì hành động được liên kết với trình kích hoạt sẽ được thực hiện bởi DBMS.

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

For: ngay khi biến cố xảy ra thì thực hiện Trigger

After: sau khi biến cố xảy ra 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');

Sau khi người dùng xóa, mà giá trị ol.B >2 thì nó sẽ in ra thông báo 'Notify Customer Relations'

1. Instead of trigger là gì?

Instead of trigger là một loại trigger đặc biệt, nó cho phép bạn bỏ qua câu lệnh INSERT, UPDATE hoặc DELETE trên một table hoặc view. Ví dụ bạn muốn khi có hành động delete trên table product thì bạn không xóa sản phẩm mà sẽ chạy một câu lệnh UPDATE status của product đó sang chế độ ẩn. Mặc dù bạn có thể không sử dung lệnh DELETE mà UPDATE trực tiếp cũng được, tuy nhiên để đảm bảo mọi câu truy vấn đều đúng thì nên sử dụngtrigger.

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 trigger AA trong bảng Q với biến cố delete. Định nghĩa biến @old với @old được gán bằng thuộc tính B của bảng ol. Nếu @old > 2 thì in ra dòng 'Notify Customer Relations'

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?

Từ khóa “after” nghĩa là trigger AA trong bảng Q được tạo sau khi xảy ra biến cố delete.

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?

Từ khóa “for” nghĩa là trigger AA trong bảng Q được tạo ngay khi xảy ra biến cố delete.

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 trigger BB trong bảng Q1 với biến cố insert. Định nghĩa biến @newA và @newB với @newA được gán bằng thuộc tính A của bảnginsert n và @newB được gán bằng thuộc tính B của bảng insert n. Sauđó thêm vào bảng Q2 2 giá trị @newA và @newB.

1. View là gì?

Dạng xem là bảng ảo có nội dung (cột và hàng) được xác định bởi một truy vấn lấy dữ liệu trong một hoặc nhiều bảng (được gọi là bảng cơ sở) hoặc các dạng xem khác trong cơ sở dữ liệu. Tuy nhiên, View không tồn tại như một cấu trúc lưu trữ dữ liệu trong csdl.

1. Mục đích của sử dụng view

Các View thường được sử dụng với các mục đích sau đây:

- Sử dụng View để tập trung trên dữ liệu được xác định.

- Sử dụng View để đơn giản hóa thao tác dữ liệu.

- Sử dụng View để tùy biến dữ liệu.

- Sử dụng View để trích xuất dữ liệu tạo báo cáo.

- Sử dụng View để bảo mật dữ liệu, che giấu thông tin.

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

- Lệnh 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\_nameWHERE condition;

1. Phân loại View?

- Standard View: view được tạo bao gồm các cột của các bảng hoặc các view khác

- Indexed View: view được tạo và được đặt chỉ mục Unique Clustered

- Partitiioned view: view được tạo bao gồm các dữ liệu được phân cụm ngang từ một hoặc nhiều bảng

1. Ý nghĩa của câu lệnh

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

Tạo View cc gồm cột A từ bảng Q1 và cột ln chứa giá trị lớnnhất của cột B trong bảng Q1 khi gộp bởi cột 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 PbanADD CONSTRAINT QL CHECK(MaNQL = 1)

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.

Sửa cấu trúc bảng NV bằng cách thêm một ràng buộc có tên là chk\_luong\_min kiểm tra lương có >10000 hay không

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) là quy tắt, quy luật của DBMS về việc INSERT, DELETE, UPDATE dữ liệu giữa hai bảng có mối quan hệ ràng buộckhóa ngoại với nhau, để đảm bảo tính đúng đắn của dữ liệu. Các thuộc tính đượctham chiếu của quan hệ thứ hai phải được khai báo DUY NHẤT hoặc KHÓACHÍNH cho quan hệ của chúng. Ràng buộc khóa ngoại trong bảng cơ sở dữ liệu làtrường phù hợp với trường khóa chính trong bảng khác. Trường khóa ngoại đượcsử dụng để tạo ra mối quan hệ với trường khóa chính của bảng chính.

- Nó quan trọng vì: Điều này ngăn cản bạn chèn thêm bất kỳ dữ liệu không hợp lệtrong trường khóa ngoại vì trường khóa ngoại chỉ chấp nhận những giá trị đã cótrong trường khóa chính.

- Tính toàn vẹn tham chiếu là sự nhất quán về CSDL giữa 2 quan hệ tham chiếuvới nhau, đảm bảo tính đúng đắn của dữ liệu trong quá trình insert, delete, updatecủa các bảng có ràng buộc khóa ngoại.

**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?

Vd: dept.managerid là khóa ngoại tham chiếu đến emp.eid. Khi xóa 1 bộ trong quan hệ dept thì không có ràng buộc nào để tuân thủ.

1. 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 int primary key,

ename nvarchar(50),

age int,

salary float

);

CREATE TABLE Works (

eid int primary key,

did int primary key,

pct\_time int

);

CREATE TABLE Dept (

did int primary key,

ename nvarchar(50),

budger float,

managerid int FOREIGN KEY REFERENCES Emp(eid)

);

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

managerid int not null FOREIGN KEY 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 EmpVALUES (101, John Doe, 32, 15000)

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

update Empset 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 ename='Toy';

Nếu Toy là trưởng phòng thì không xóa được vì ảnh hướng đến khóa ngoài.  
**Exercise 3.** Consider the following relations:



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

sid : - là khóa chính trong bảng Students, nhưng lại xuất hiện trong bảng Enrolled đóng vai trò là khóa ngoại kết nối 2 bảng

fid : - là khóa chính trong bảng Faculty, nhưng lại xuất hiện trong bảng Teaches đóng vàitrò là khóa ngoại kết nối 2 bảng

cid : là khóa chính trong bảng Course, nhưng lại xuất hiện trong bảng Teaches và bảng Enrolled đóng vai trò là khóa ngoại

rno : là khóa chính trong bảng Rooms, nhưng lại xuất hiện trong bảng Meets\_In đóng vai trò là khóa ngoại kết nối giữa 2 bảng

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.

Vd: Không có môn nào vượt quá 10 tín chỉ

1. Create the above database with their constraints.

CREATE DATABASE School;

use School;

CREATE TABLE Students (

sid nvarchar(10) primary key,

name nvarchar(max),

login varchar(50),

age int,

gpa float

);

CREATE TABLE Faculty (

fid nvarchar(10) primary key,

fname nvarchar(max),

sal float

);

CREATE TABLE Courses (

cid nvarchar(10) primary key,

cname nvarchar(50),

credits int,

check(credits < 10)

);

CREATE TABLE Rooms (

rno int primary key,

raddress nvarchar(200),

capacity int

);

CREATE TABLE Enrolled (

studentid nvarchar(10) FOREIGN KEY REFERENCES

Students(studentid),

cid nvarchar(10) FOREIGN KEY REFERENCES Courses(cid),

grade varchar(10)

);

CREATE TABLE Teaches (

fid nvarchar(10) FOREIGN KEY REFERENCES Faculty(fid),

cid nvarchar(10) FOREIGN KEY REFERENCES Courses(cid),

);

CREATE TABLE Meets\_In (

cid nvarchar(10) FOREIGN KEY REFERENCES Courses(cid),

rno int FOREIGN KEY REFERENCES Rooms(rno),

time varchar(20)

);

**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 Check(salary > 10000);

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

CREATE FUNCTION checkAgeManager(@managerid int)

RETURNS int AS

BEGIN

DECLARE @age int;

SELECT @age = E.age FROM Emp E WHERE E.eid = @managerid; IF (@age IS

NULL) RETURN 0;

IF (@age > 30) RETURN 1;

RETURN 0;

END;

ALTER TABLE Dept ADD CONSTRAINT checkAgeManagerCS CHECK

(dbo.checkAgeManager(managerid) = 1)

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.

CREATE TRIGGER CheckSalaryBeforeDelete

BEFORE DELETE ON Emp

FOR EACH ROW

BEGIN

DECLARE manager\_salary REAL;

-- Lấy lương của quản lý của phòng ban mà nhân viên này làm việc

SELECT m.salary INTO manager\_salary

FROM Emp m

JOIN Dept d ON m.eid = d.managerid

JOIN Works w ON d.did = w.did

WHERE w.eid = OLD.eid;

-- Kiểm tra nếu lương của nhân viên lớn hơn lương của quản lý

IF OLD.salary > manager\_salary THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Can’t delete';

END IF;

END;

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

* Điểm mạnh của Trigger:

- Tính toán cập nhật dữ liệu tự động

- Kiểm tra dữ liệu nhập-Kiểm tra ràng buộc phức tạp

- Bẫy lỗi dể hiểu

* 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.

* Sự tương phản giữa trigger với các ràng buộc toàn vẹn khác được hỗ trợ bởi SQL

- Trigger có thể tạo ra 1 bảng ảo mà khi thao tác trên CSDL nó sẽ tự động cập nhật

- Chỉ là một phần mở rộng của việc kiểm tra tính hợp lệ của dữ liệu chứ không thay thế được hoàn toàn công việc này.

- Hoạt động ngầm ở trong CSDL, không hiển thị ở tầng giao diện.Do đó, khó chỉ ra được điều gì xảy ra ở tầng CSDL.

**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.

Create trigger TG\_SALARY\_LESS on Emp after insert, delete as

Declare @new int, @old int, @ssn int;

Select @new = ne.salary, @old = ol.salary, @ssn = ol.ename

From inserted ne, deleted ol;

Where ne.ename = ol.ename

If(@new >= 1000)

Begin

Update Emp

Set salary = @new

Where ename = @ssn

End

1. Every manager must be also be an employee.

Create trigger MANAGER on Dept after insert, delete as

Declare @new int, @old int, @ssn int;

Select @new = ne.managerid, @old = ol.managerid, @ssn =

Emp.eid

From inserted ne, delete ol, Emp;

Where ne.managerid = ol.managerid

If(@new = @ssn)

Begin

Update Dept

Set managerid = @new

Where managerid = @old

End

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

Create trigger TIME\_SUM on Works after insert, delete as

Declare @new int, @old int, @ssn int;

Select @new = ne.pct\_time, @old = ol.pct\_time, @ssn = ol.eid

From insert ne, delete ol;

Where ne.eid = ol.eid

If(@new < 100)

Begin

Update Works

Set pct\_time = @new

Where eid = @ssn

End

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

manages.

Create trigger MANAGER\_SALARY on Emp after insert, delete as

Declare @new int, @old int, @ssn int, @e\_id int, @e\_salary

int

Select @new = ne.salary, @old = ol.salary, @ssn = ol.eid,

@e\_id = Emp.eid, @e\_salary = Emp.salary

From insert ne, delete ol, Emp;

Where ne.eid = ol.eid

If(Dept.managerid = @ssn and Dept.did = Works.did and

Works.eid = @e\_id and @new > @e\_salary)

Begin

Update Emp

Set salary = @new

Where eid = @ssn

End

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

be at least as much.

Create trigger MANAGER\_SALARY\_INCREASED on Emp after insert,

delete as

Declare @new\_e int, @old\_e int, @ssn\_e int, @new\_m int,

@old\_m int, @ssn\_m int;

Select @new\_e = ne1.salary, @old\_e = ol1.salary, @ssn\_e =

ol1.eid, @ssn\_m = ol2.eid

From insert ne1, delete ol1, insert ne2, delete ol2;

Where ne1.eid = ol1.eid and ne2.eid = ol2.eid

If(Dept.did = Works.did and Works.eid = @ssn\_e and @new\_e >

@old\_e)

Begin

Update Emp

Set salary = @new\_e

Where eid = @ssn\_e

Set salary = @new\_m + @new\_e - @old\_e

Where eid = @ssn\_m

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 MANAGER\_SALARY\_INCREASED on Emp, Dept after

insert, delete as

Declare @new\_e int, @old\_e int, @ssn\_e int, @new\_m int,

@old\_m int, @ssn\_m int, @new\_b int, @old\_b int, @ssn\_b int;

Select @new\_e = ne1.salary, @old\_e = ol1.salary, @ssn\_e =

ol1.eid, @ssn\_m = ol2.eid, @new\_b = ne3.budget, @old\_b =

ol3.budget, @ssn\_b = ol3.did

From insert ne1, delete ol1, insert ne2, delete ol2, insert

ne3, delete ol3;

Where ne1.eid = ol1.eid and ne2.eid = ol2.eid and ne3.did =

ol3.did

If(Dept.did = Works.did and Works.eid = @ssn\_e and @new\_e >

@old\_e)

Begin

Update Emp

Set salary = @new\_e

Where eid = @ssn\_e

Set salary = @new\_m + @new\_e - @old\_e

Where eid = @ssn\_m

Update Dept

Set budget = @new\_b

Where did = @ssn\_b

End

**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

Đầu tiên, hệ thống sẽ mở rộng view SeniorEmp thành câu lệnh SELECT tương ứng mà view định nghĩa:

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

FROM Emp E

WHERE E.age > 50;

Sau đó, hệ thống sẽ kết hợp điều kiện trong truy vấn gốc với điều kiện trong view. Kết quả là một truy vấn SQL như sau:

SELECT E.ename

FROM Emp E

WHERE E.age > 50 AND E.salary > 100000

Cuối cùng, hệ thống sẽ thực thi truy vấn kết hợp này trên bảng Emp. Nó sẽ tìm các nhân viên có age > 50 và salary > 100000, sau đó trả về ename của những nhân viên đó.

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

CREATE VIEW EmpSalaries AS

SELECT eid, salary

FROM Emp;

View EmpSalaries chỉ chứa thông tin về eid và salary của các nhân viên từ bảng Emp.

Nếu có bất kỳ thay đổi nào đối với salary trong bảng Emp, thì những thay đổi đó sẽ tự động được phản ánh trong view EmpSalaries vì view này trực tiếp phụ thuộc vào bảng Emp mà không có phép tính hay điều kiện phức tạp nào khác.

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 SeniorEmpNames AS

SELECT ename

FROM Emp

WHERE age > 50;

View SeniorEmpNames chỉ chứa tên các nhân viên có tuổi trên 50.

Nếu bạn cố gắng cập nhật dữ liệu trong view này, chẳng hạn như thay đổi ename, hệ thống không thể tự động xác định cách thay đổi trong view sẽ ảnh hưởng đến bảng gốc Emp mà không có hướng dẫn rõ ràng.

Điều này là do age không có trong view, nên hệ thống không biết có thể cần phải cập nhật giá trị age để giữ cho điều kiện age > 50 đúng. Hơn nữa, view không chứa khóa chính (eid), làm cho việc xác định dòng cụ thể trong Emp cần cập nhật trở nên khó khăn.