



Chương 7: Định nghĩa dữ liệu với ngôn ngữ SQL

Tham khảo tài liệu [A]:

Chapter 6 : SQL: Data Definition



Nội dung chương 7

- ❖ Các kiểu dữ liệu của SQL chuẩn.
- ❖ Mục đích của việc tăng cường tính toàn vẹn của SQL.
- ❖ Cách định nghĩa các ràng buộc toàn vẹn dùng SQL.
- ❖ Cách viết lệnh CREATE và ALTER TABLE có tăng cường tính toàn vẹn dữ liệu.



Nội dung chương 7

- ❖ Mục đích của VIEW
- ❖ Tạo và xóa VIEW dùng SQL.
- ❖ Cách DBMS thực hiện các thao tác trên VIEW.
- ❖ Điều kiện để VIEW cập nhật dữ liệu được
- ❖ Ưu và nhược điểm của VIEW
- ❖ Mô hình giao tác theo chuẩn ISO
- ❖ Dùng lệnh GRANT và REVOKE



Các kiểu dữ liệu của SQL chuẩn ISO

Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	



Đặc tính tăng cường tính toàn vẹn dữ liệu

❖ Xét các loại ràng buộc toàn vẹn:

- Ràng buộc bắt buộc nhập liệu
- Ràng buộc về miền thuộc tính
- Ràng buộc về thực thể
- Ràng buộc ASSERTION



Đặc tính tăng cường tính toàn vẹn dữ liệu

Ràng buộc bắt buộc nhập liệu:

position VARCHAR(10) NOT NULL

Ràng buộc về miền thuộc tính:

(a) CHECK

sex **CHAR** **NOT NULL**

CHECK (sex IN ('M', 'F'))



Đặc tính tăng cường tính toàn vẹn dữ liệu

Ràng buộc về miền thuộc tính:

(a) CHECK

Diemtoan DECIMAL(4,2)

CONSTRAINT chk_diemtoan

CHECK(diemtoan>=0 AND diemtoan<=10)



Đặc tính tăng cường tính toàn vẹn dữ liệu

Ràng buộc về miền thuộc tính:

(a) CHECK

CONSTRAINT chk_lop

CHECK (NamNhậpHọc<=YEAR(GETDATE()) AND
HéDaoTao IN ('chính quy', 'liên thông'))



Đặc tính tăng cường tính toàn vẹn dữ liệu

(b) CREATE DOMAIN

CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]

Ví dụ:

CREATE DOMAIN SexType AS CHAR
 CHECK (VALUE IN ('M', 'F'));
sex SexType NOT NULL



Đặc tính tăng cường tính toàn vẹn dữ liệu

- ❖ *searchCondition* có thể tìm trong một bảng dữ liệu :

```
CREATE DOMAIN BranchNo AS CHAR(4)
CHECK (VALUE IN (SELECT branchNo
                  FROM Branch));
```

- ❖ Miền có thể được xóa bằng DROP DOMAIN:

```
DROP DOMAIN DomainName
[RESTRICT | CASCADE]
```



Ràng buộc về thực thể

- ❖ Khóa chính của một bảng phải chứa một giá trị duy nhất và khác NULL
- ❖ Chuẩn ISO hỗ trợ mệnh đề FOREIGN KEY trong câu lệnh CREATE và ALTER TABLE

PRIMARY KEY(staffNo)

PRIMARY KEY(clientNo, propertyNo)

- ❖ Chỉ dùng một mệnh đề PRIMARY KEY cho mỗi bảng. Đối với các thuộc tính khác, muốn cấm việc trùng dữ liệu thì dùng lệnh UNIQUE:

UNIQUE.telNo

UNIQUE (name,address)



Ràng buộc khoá ngoại

- ❖ Khóa ngoại (FK) là 1 hay nhiều cột có tính chất liên kết mỗi hàng ở bảng con chứa FK đến 1 hàng có giá trị tương ứng trong bảng cha.
- ❖ Vậy nếu FK chứa một giá trị, thì giá trị đó phải tham khảo đến một hàng hiện hữu ở bảng cha của FK.
- ❖ Chuẩn ISO hỗ trợ định nghĩa FOREIGN KEY trong CREATE và ALTER TABLE:

FOREIGN KEY(branchNo) REFERENCES Branch



Ràng buộc trong tham khảo dữ liệu

Ràng buộc tham chiếu trong SQL

- * Thuộc tính được tham chiếu đến (Referenced attribute) phải là PRIMARY KEY hoặc UNIQUE

vd: Student.ID, Student.(name,address),
Campus.location

- * Thuộc tính tham chiếu (Referencing attribute) được gọi là FOREIGN KEY - Khoá ngoài.

vd: Apply.ID, Apply.location



Ràng buộc trong tham khảo dữ liệu

- ❖ Bất kỳ lệnh INSERT/UPDATE nào làm cho dữ liệu của FK không giống với giá trị nào của FK ở bảng cha đều không thực hiện được.
- ❖ Thao tác update/delete trên bảng cha có liên quan đến FK bị phụ thuộc vào mệnh đề ON UPDATE và ON DELETE:
 - CASCADE
 - SET DEFAULT
 - SET NULL,
 - NO ACTION



Ràng buộc trong tham khảo dữ liệu

CASCADE: xóa dây chuyền

SET NULL: Khi xóa hàng dữ liệu trong bảng cha thì tại các giá trị tương ứng của FK ở bảng con sẽ được gán về NULL (Nếu giá trị của FK là khác NULL)

SET DEFAULT: Khi xóa hàng dữ liệu trong bảng cha thì tại các giá trị tương ứng của FK ở bảng con sẽ được gán về mặc định nào đó.

NO ACTION: Không cho xóa ở bảng cha (mặc định)



Ràng buộc trong tham khảo dữ liệu

**FOREIGN KEY (staffNo) REFERENCES Staff
ON DELETE SET NULL**

**FOREIGN KEY (ownerNo) REFERENCES Owner
ON UPDATE CASCADE**



Ràng buộc ASSERTION

- ❖ Có thể dùng CHECK/UNIQUE trong câu lệnh CREATE và ALTER TABLE.
- ❖ Hay:

**CREATE ASSERTION AssertionName
CHECK (searchCondition)**

- ❖ Giống như sử dụng mệnh đề CHECK



Ràng buộc ASSERTION

❖ GENERAL ASSERTION

CREATE ASSERTION HighVals

CHECK(4.0 < (SELECT avg(GPA) FROM Student)
AND 1200 < (SELECT avg(SAT) FROM Student))



Ràng buộc từ ASSERTION

- ❖ Ví dụ: Một sinh viên với GPA < 4.0 chỉ được đăng ký vào campus có rank (cấp bậc) > 3.

CREATE ASSERTION RestrictApps

CHECK(NOT EXISTS

```
(SELECT * FROM Student, Apply, Campus  
WHERE Student.ID = Apply.ID  
AND Apply.location = Campus.location  
AND Student.GPA <4.0  
AND Campus.rank <3))
```



Ràng buộc ASSERTION

```
CREATE ASSERTION StaffNotHandlingTooMuch  
CHECK (NOT EXISTS (SELECT staffNo  
                   FROM PropertyForRent  
                   GROUP BY staffNo  
                   HAVING COUNT(*) > 100))
```



Định nghĩa dữ liệu

- ❖ SQL DDL cho phép tạo và xóa schema, domain, table, view, và index

- ❖ Các câu lệnh SQL DDL chính là:

CREATE SCHEMA

DROP SCHEMA

CREATE/ALTER DOMAIN

DROP DOMAIN

CREATE/ALTER TABLE

DROP TABLE

CREATE VIEW

DROP VIEW

- ❖ Nhiều DBMS cho phép:

CREATE INDEX

DROP INDEX



Định nghĩa dữ liệu

- ❖ Các quan hệ và các đối tượng CSDL khác tồn tại trong một môi trường (*environment*)
- ❖ Mỗi môi trường chứa 1 hay nhiều *catalog*, mỗi catalog bao gồm tập hợp các lược đồ (schema)
- ❖ Lược đồ có tên theo ý nghĩa là tập hợp các đối tượng CSDL có liên quan nhau
- ❖ Các đối tượng trong một lược đồ có thể là: tables, views, domains, assertions, collations, translations, và các tập ký tự. Một lược đồ của một tác giả.

**CREATE SCHEMA [Name |
AUTHORIZATION CreatorId]**

DROP SCHEMA Name [RESTRICT | CASCADE]

- ❖ Với RESTRICT (default), schema phải rỗng, nếu không rỗng sẽ không xóa được
- ❖ Với CASCADE sẽ xóa dây chuyền tất cả các đối tượng có liên quan đến schema



CREATE TABLE

CREATE TABLE TableName
(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] [...,]}
{[FOREIGN KEY (listOfFKColumns)
REFERENCES ParentTableName
[(listOfCKColumns)],
[ON UPDATE referentialAction]
[ON DELETE referentialAction]] [...]}
{[CHECK searchCondition)] [...] })



CREATE TABLE

- ❖ Tạo một bảng với các cột, mỗi cột có kiểu dữ liệu riêng
- ❖ Với NOT NULL, cột không chấp nhận giá trị NULL
- ❖ Mỗi cột có thể đặt một giá trị DEFAULT
- ❖ Khóa chính phải đặt NOT NULL
- ❖ FOREIGN KEY có chỉ ra chế độ thao tác xóa/sửa ở cột có liên quan bảng cha



Ví dụ 7.1 - CREATE TABLE

- ❖ **CREATE DOMAIN OwnerNumber AS VARCHAR(5)**
 CHECK (VALUE IN (SELECT ownerNo FROM Owner));
- ❖ **CREATE DOMAIN StaffNumber AS VARCHAR(5)**
 CHECK (VALUE IN (SELECT staffNo FROM Staff));
- ❖ **CREATE DOMAIN PNumber AS VARCHAR(5);**
- ❖ **CREATE DOMAIN PRooms AS SMALLINT;**
 CHECK(VALUE BETWEEN 1 AND 15);
- ❖ **CREATE DOMAIN PRent AS DECIMAL(6,2)**
 CHECK(VALUE BETWEEN 0 AND 9999.99);



Ví dụ 7.1 - CREATE TABLE

```
CREATE TABLE PropertyForRent (
    propertyNo PNumber          NOT NULL, ....
    rooms       PRooms            NOT NULL DEFAULT 4,
    rent        PRent             NOT NULL, DEFAULT 600,
    ownerNo     OwnerNumber      NOT NULL,
    staffNo     StaffNumber
                Constraint StaffNotHandlingTooMuch ....
    branchNo   BranchNumber     NOT NULL,
PRIMARY KEY (propertyNo),
FOREIGN KEY (staffNo) REFERENCES Staff
ON DELETE SET NULL ON UPDATE CASCADE ....);
```

- ❖ Thêm một cột mới cho một bảng
- ❖ Xóa một cột của một bảng
- ❖ Thêm một ràng buộc cho bảng
- ❖ Xóa một ràng buộc của bảng
- ❖ Đặt giá trị default cho một cột
- ❖ Xóa việc đặt giá trị default cho một cột



Ví dụ 7.2(a) - ALTER TABLE

Bảng Staff: xóa giá trị default 'Assistant' cho cột position và đặt giá trị default cho cột sex là Female ('F').

ALTER TABLE Staff

ALTER position DROP DEFAULT;

ALTER TABLE Staff

ALTER sex SET DEFAULT 'F';



Ví dụ 7.2(b) - ALTER TABLE

Xóa ràng buộc không cho một nhân viên quản lý hơn 100 tài sản; thêm vào bảng Client một cột mới

ALTER TABLE PropertyForRent

DROP CONSTRAINT StaffNotHandlingTooMuch;

ALTER TABLE Client

ADD prefNoRooms PRooms;

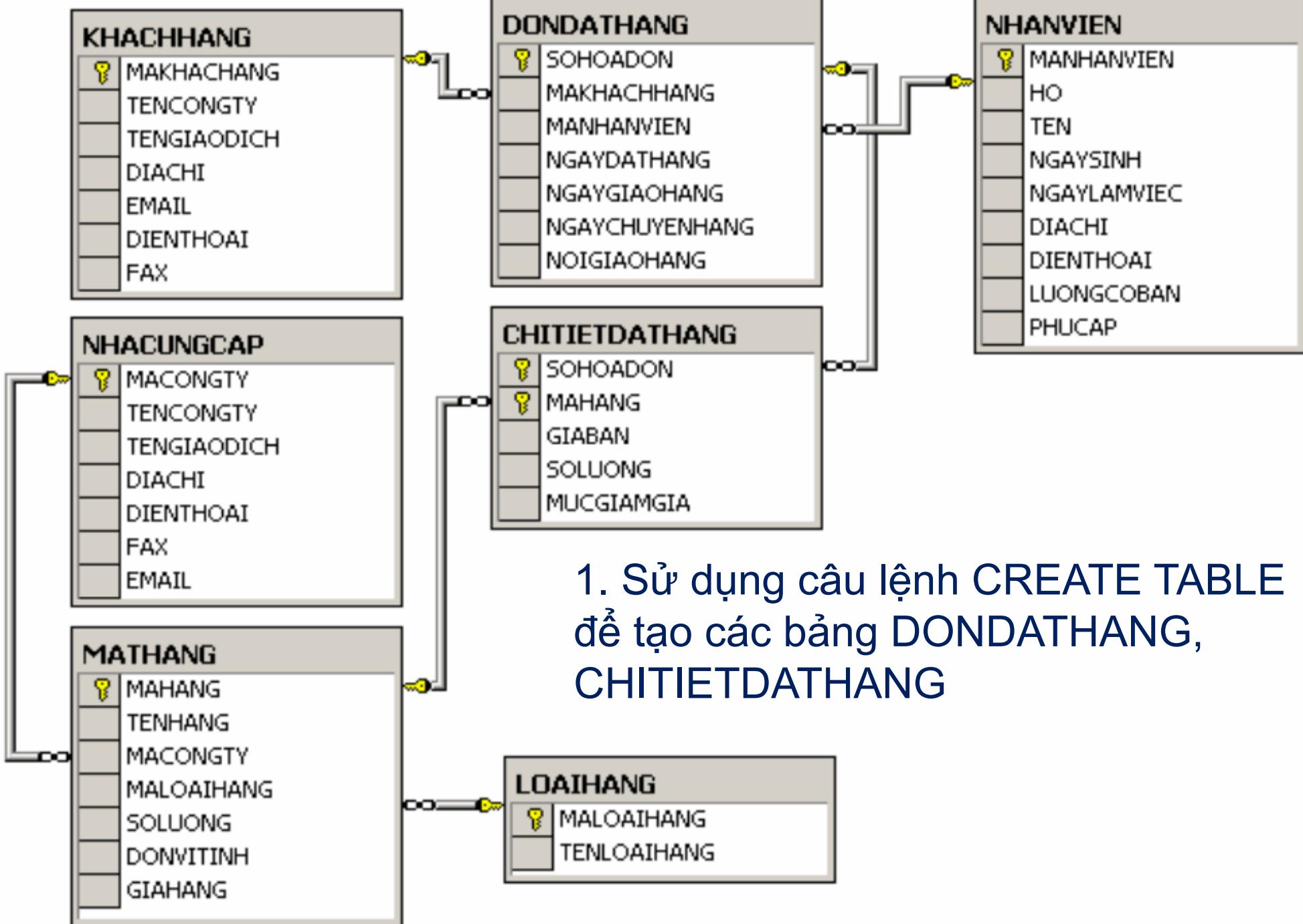


DROP TABLE

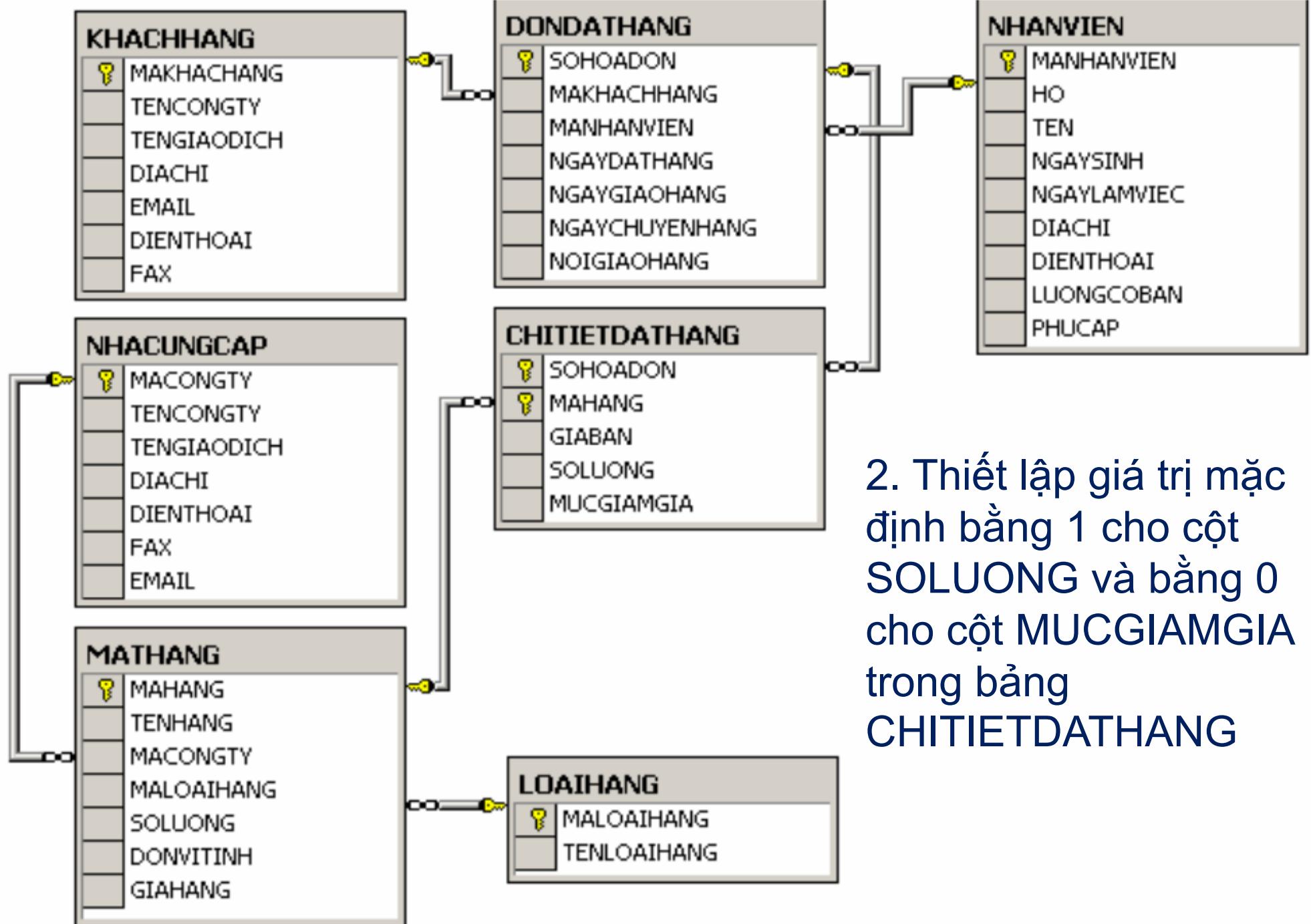
DROP TABLE TableName [RESTRICT | CASCADE]

ví dụ: **DROP TABLE PropertyForRent;**

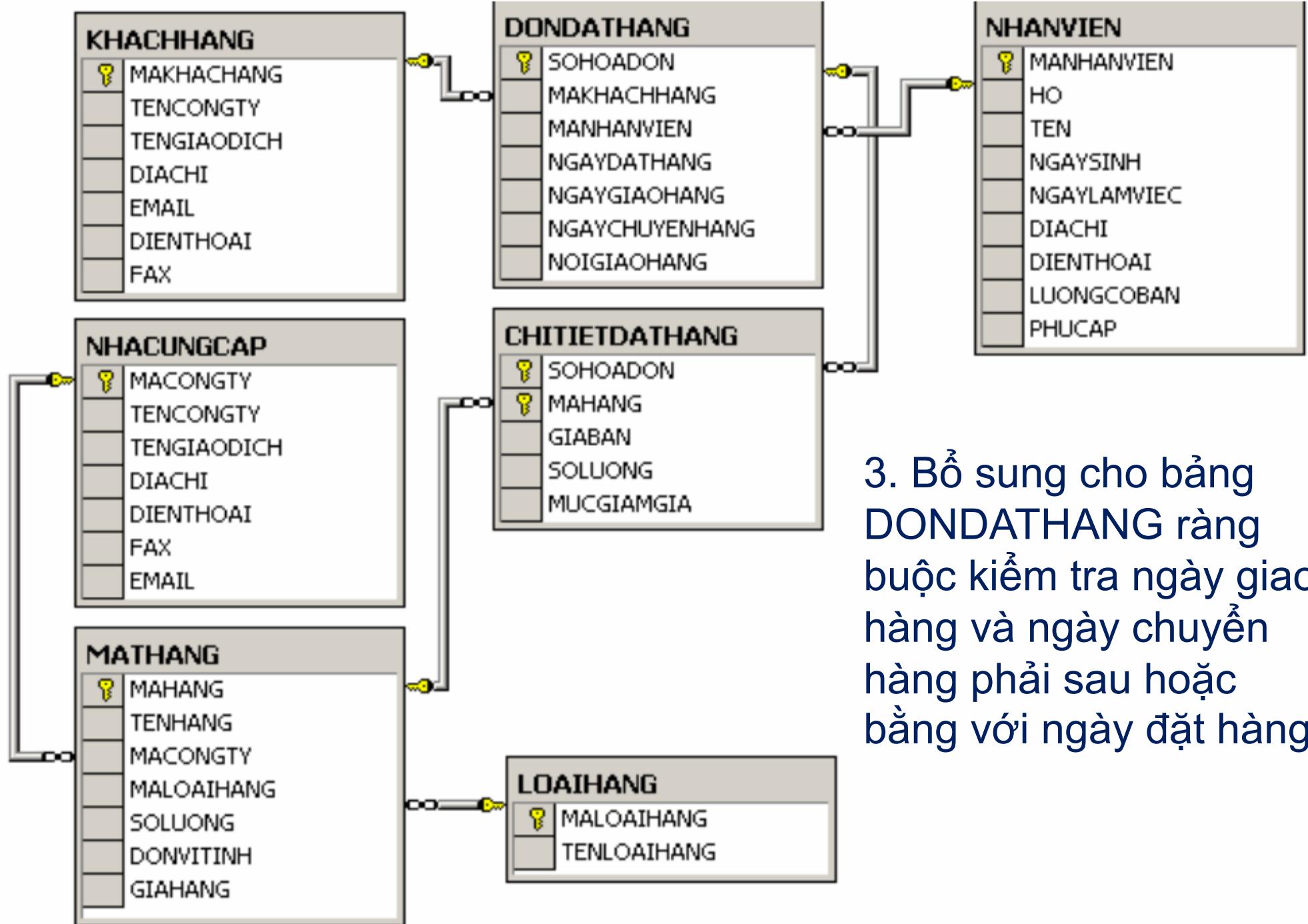
- ❖ Xóa bảng và dữ liệu trong bảng
- ❖ Với RESTRICT, nếu các đối tượng khác có dùng bảng thì không thể xóa bảng
- ❖ Với CASCADE: xóa dây chuyền



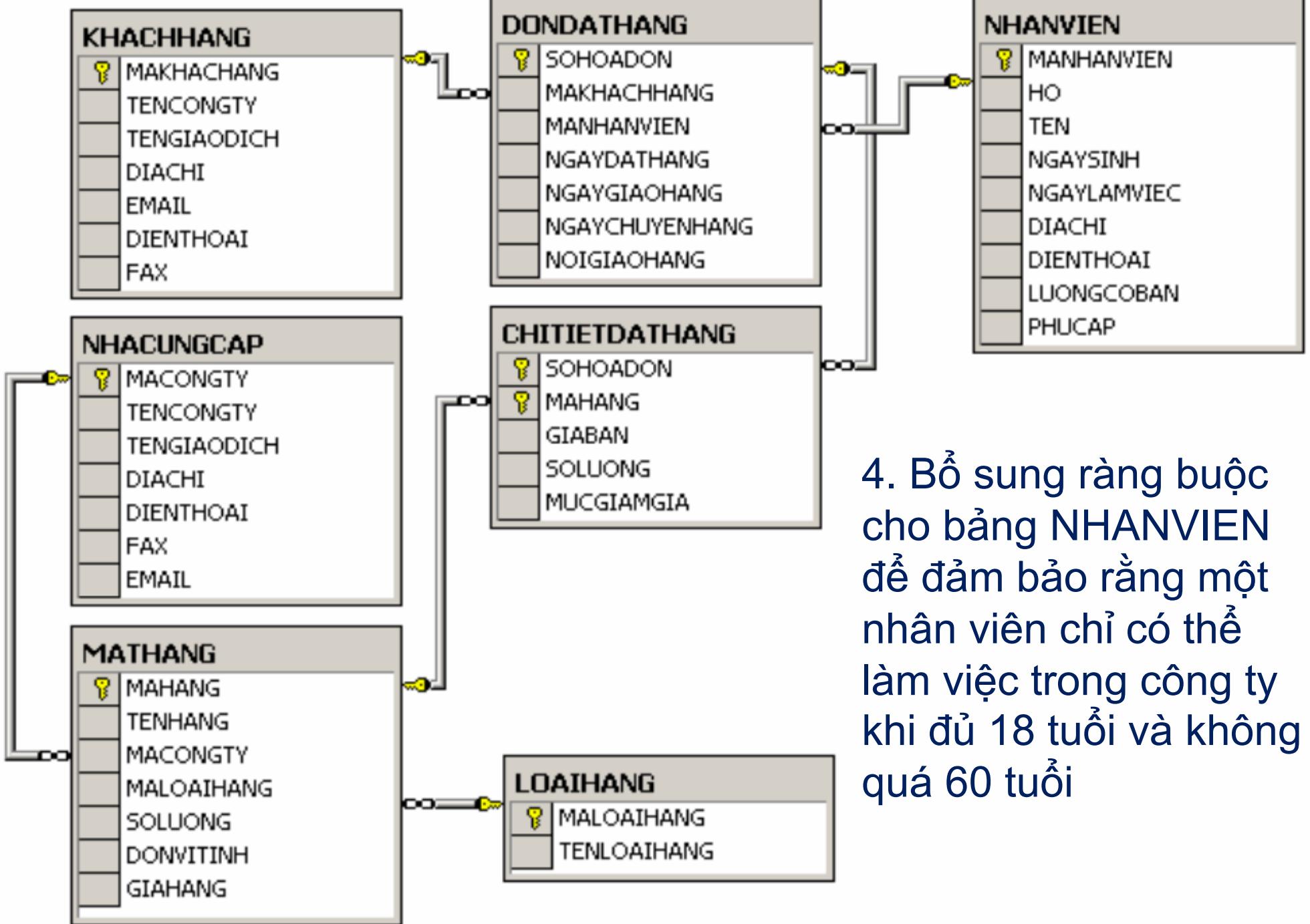
1. Sử dụng câu lệnh CREATE TABLE
để tạo các bảng DONDATHANG,
CHITIETDATHANG



2. Thiết lập giá trị mặc định bằng 1 cho cột SOLUONG và bằng 0 cho cột MUCGIAMGIA trong bảng CHITIETDATHANG



3. Bổ sung cho bảng DONDATHANG ràng buộc kiểm tra ngày giao hàng và ngày chuyển hàng phải sau hoặc bằng với ngày đặt hàng



4. Bổ sung ràng buộc
cho bảng NHANVIEN
để đảm bảo rằng một
nhân viên chỉ có thể
làm việc trong công ty
khi đủ 18 tuổi và không
quá 60 tuổi



BÀI TẬP

Xét bảng:

TranDau(MaTD, NgayTD, GioBD, GioKT)

Viết lệnh tạo bảng và có các ràng buộc sau:

- Giờ bắt đầu trận đấu phải nhỏ hơn giờ kết thúc trận đấu.**
- Hai trận đấu diễn ra trong 1 ngày thì không được bắt đầu chung một giờ bắt đầu**



View

- ❖ Nội dung của một view được định nghĩa như là một truy vấn trên một hay nhiều quan hệ nền
- ❖ Với view phân giải (view resolution), bất kỳ thao tác nào trên view sẽ chuyển thành các thao tác trên các quan hệ nền của nó
- ❖ Với view vật chất (view materialization), view được lưu như là một bảng tạm. Khi nào các quan hệ nền được cập nhật thì view nên cập nhật theo.



Lệnh CREATE VIEW

CREATE VIEW ViewName [(newColumnName [,...])]

AS subselect

[WITH [CASCADED | LOCAL] CHECK OPTION]

- ❖ Có thể gán tên cho mỗi cột của view.
- ❖ Nếu có ghi danh sách tên các cột thì số lượng các tên cột phải bằng số lượng cột trong kết quả của *subselect*.
- ❖ Nếu không có ghi danh sách tên các cột thì các cột lấy tên từ kết quả của *subselect*.



Lệnh CREATE VIEW

- ❖ Nên liệt kê các tên của view để tránh sự nhầm lẫn.
- ❖ Phần *subselect* được gọi là phần truy vấn định nghĩa view (defining query).
- ❖ Phần WITH CHECK OPTION bảo đảm rằng nếu hàng dữ liệu nào không thỏa phần WHERE thì sẽ không được thêm vào bảng dữ liệu nền.



Ví dụ 7.3 - Tạo View theo chiều ngang

Tạo view để giám đốc chi nhánh B003 chỉ thấy chi tiết về các nhân viên làm việc tại chi nhánh này.

```
CREATE VIEW Manager3Staff  
AS    SELECT *  
      FROM Staff
```

Table 6.3 Data for view Manager3Staff.

staffNo	fName	IName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003



Ví dụ 7.4 - Tạo View theo chiều dọc

Tạo view chi tiết nhân viên tại chi nhánh B003 không có cột salaries.

```
CREATE VIEW Staff3  
AS SELECT staffNo, fName, lName, position, sex  
FROM Staff  
WHERE branchNo = 'B003';
```

Table 6.4 Data for view Staff3.

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F



Ví dụ 7.5 - Tạo View có nhóm, có kết dữ liệu

Tạo view nhân viên quản lý tài sản, bao gồm: mã chi nhánh đang làm việc, mã nhân viên, và mã tài sản mà nhân viên đang quản lý.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
AS SELECT s.branchNo, s.staffNo, COUNT(*)
        FROM Staff s, PropertyForRent p
        WHERE s.staffNo = p.staffNo
        GROUP BY s.branchNo, s.staffNo;
```



Ví dụ 7.5 - Tạo View có nhóm, có kết dữ liệu

Table 6.5 Data for view StaffPropCnt.

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1



Lệnh DROP VIEW

DROP VIEW ViewName [RESTRICT | CASCADE]

- ❖ Để xóa view trong cơ sở dữ liệu .
- ❖ Ví dụ:

DROP VIEW Manager3Staff;

- ❖ Với CASCADE: xóa dây chuyền
- ❖ Với RESTRICT (mặc định): không cho xóa view nếu các đối tượng khác có sử dụng view



Sự phân giải view (view phân giải)

Đếm số lượng tài sản được quản lý bởi từng nhân viên ở chi nhánh B003.

```
SELECT staffNo, cnt  
FROM StaffPropCnt  
WHERE branchNo = 'B003'  
ORDER BY staffNo;
```



Sự phân giải view (view phân giải)

(a) Các tên cột của view sau SELECT được phân giải thành các tên tương ứng trong truy vấn định nghĩa:

```
SELECT s.staffNo As staffNo, COUNT(*) As cnt
```

(b) Tên view sau FROM được thay thế bằng danh sách sau phần FROM của truy vấn định nghĩa view:

```
FROM Staff s, PropertyForRent p
```



Sự phân giải view (view phân giải)

(c) Cuối cùng, truy vấn sau sẽ thực thi để lấy kết quả:

```
SELECT s.staffNo, COUNT(*)  
FROM staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo AND  
      branchNo = 'B003'  
GROUP BY s.branchNo, s.staffNo  
ORDER BY s.staffNo;
```



Giới hạn cho view

SQL đặt ra nhiều giới hạn trong việc tạo và sử dụng view.

(a) Nếu view có cột sử dụng hàm thống kê:

- Cột này chỉ có thể xuất hiện trong SELECT và ORDER BY của câu truy vấn view này.
- Cột này không dùng trong WHERE và cũng không nằm trong một hàm thống kê trong câu truy vấn view này.



Giới hạn cho view

- ❖ **Ví dụ, truy vấn sau là sai:**

```
SELECT COUNT(cnt)  
FROM StaffPropCnt;
```

- ❖ **Tương tự, truy vấn sau là sai :**

```
SELECT *  
FROM StaffPropCnt  
WHERE cnt > 2;
```



Giới hạn cho view

- (b) View có nhóm không được kết với các view khác hay kết với các bảng dữ liệu nền.
- ❖ Ví dụ, view tên StaffPropCnt là một view có nhóm, do đó không được kết view này với các bảng hay các view khác.



Việc cập nhật view

- ❖ Cập nhật dữ liệu từ các bảng nền sẽ kéo theo sự cập nhật dữ liệu lên các view có liên quan
- ❖ Tuy nhiên chiều ngược lại là không phải bao giờ cũng thực hiện được.



Việc cập nhật view

- ❖ Xét view tên StaffPropCnt, nếu chúng ta thử thêm một hàng dữ liệu: tại chi nhánh B003, nhân viên SG5 quản lý 2 tài sản:

```
INSERT INTO StaffPropCnt  
VALUES ('B003', 'SG5', 2);
```

- ❖ Việc này dẫn tới việc bắt buộc phải nhập 2 hàng dữ liệu trong bảng PropertyForRent (2 tài sản do SG5 quản lý), tuy nhiên do không có thông tin về 2 mã tài sản nên không thể thêm 2 hàng dữ liệu này được.



Việc cập nhật view

- ❖ SQL chuẩn ISO qui định một view cập nhật được nếu:
 - Không dùng DISTINCT
 - Sau SELECT là các tên cột, và không dùng các cột hơn một lần
 - Sau FROM chỉ có một bảng
 - Không có lệnh SELECT lồng nhau dùng các bảng khác
 - Không có GROUP BY và HAVING
 - Dữ liệu thêm vào phải thỏa các ràng buộc toàn vẹn của bảng dữ liệu



Dùng WITH CHECK OPTION

- ❖ Các hàng trong có mặt trong view vì thỏa điều kiện WHERE.
- ❖ Nếu một hàng không còn thỏa điều kiện WHERE sẽ biến mất khỏi view.
- ❖ Các hàng mới có thể sẽ xuất hiện trong view khi có sự chèn/thêm dữ liệu (nếu thỏa điều kiện WHERE)
- ❖ Các hàng mới có hoặc bị mất đi khỏi view gọi là các hàng dịch chuyển vào ra (*migrating rows*)
- ❖ WITH CHECK OPTION cấm các hàng dịch chuyển ra khỏi view.



Dùng WITH CHECK OPTION

- ❖ LOCAL/CASCDED
- ❖ Nếu dùng LOCAL: không cho các hàng ra khỏi (biến khỏi) view trừ khi các hàng đó đã biến khỏi các bảng hay view khác mà view này đang tham khảo.
- ❖ Nếu dùng CASCDED (default): không cho các hàng ra khỏi (biến khỏi) view.



Ví dụ 7.6 - Dùng WITH CHECK OPTION

CREATE VIEW Manager3Staff

```
AS      SELECT *
        FROM Staff
        WHERE branchNo = 'B003'
        WITH CHECK OPTION;
```

- ❖ Không thể sửa B003 thành B002 vì khi đó các hàng sẽ biến khỏi view.



Ví dụ 7.6 - Dùng WITH CHECK OPTION

- ❖ Nếu view tên Manager3Staff không lấy dl trực tiếp từ bảng Staff mà từ một view khác:

```
CREATE VIEW LowSalary  
AS SELECT * FROM Staff WHERE salary > 9000;  
CREATE VIEW HighSalary  
AS SELECT * FROM LowSalary  
      WHERE salary > 10000  
WITH LOCAL CHECK OPTION;  
CREATE VIEW Manager3Staff  
AS SELECT * FROM HighSalary  
      WHERE branchNo = 'B003';
```



Ví dụ 7.6 - WITH CHECK OPTION

UPDATE Manager3Staff

SET salary = 9500

WHERE staffNo = 'SG37';

- ❖ Update không thể thực hiện được: cho dù hàng dl có thể biến khỏi HighSalary, nhưng không thể biến khỏi LowSalary.
- ❖ Tuy nhiên, nếu update gán Salary = 8000, update sẽ thành công vì hàng dl không thuộc LowSalary.



Ví dụ 7.6 - WITH CHECK OPTION

- ❖ Nếu HighSalary dùng WITH CASCADED CHECK OPTION, thì không thể gán Salary bằng 9500 hay 8000 vì hàng dl sẽ biến khỏi HighSalary.
- ❖ Để tránh việc dì thường như vậy, các view nên dùng WITH CASCADED CHECK OPTION.

- ❖ Độc lập dữ liệu
- ❖ Tiết kiệm chi phí
- ❖ Tăng tính an toàn
- ❖ Giảm độ phức tạp
- ❖ Tiện lợi
- ❖ Mềm dẻo
- ❖ Toàn vẹn dữ liệu

Ưu điểm của View

MASV	HODDEM	TEN	NGAYSINH	...
0241010001	Ngô Thị Nhật	Anh	Nov 27 1982	...
0241010002	Nguyễn Thị Ngọc	Anh	Mar 21 1983	...
0241010003	Ngô Việt	Bắc	May 11 1982	...
0241010004	Nguyễn Đình	Bình	Oct 6 1982	...
0241010005	Hồ Đăng	Chiến	Jan 20 1982	...
0241020001	Nguyễn Tuấn	Anh	Jul 15 1979	...
0241020002	Trần Thị Kim	Anh	Nov 4 1982	...
...

Table SINHVIEN

MALOP	TENLOP	...
C24101	Toán K24	...
C24102	Tin K24	...
C24103	Lý K24	...
...

Table LOP



MASV	HODDEM	TEN	TUOI	TENLOP
0241010001	Ngô Thị Nhật	Anh	22	Toán K24
0241010002	Nguyễn Thị Ngọc	Anh	21	Toán K24
0241010003	Ngô Việt	Bắc	22	Toán K24
0241010004	Nguyễn Đình	Bình	22	Toán K24
0241010005	Hồ Đăng	Chiến	22	Toán K24
0241020001	Nguyễn Tuấn	Anh	25	Tin K24
0241020002	Trần Thị Kim	Anh	22	Tin K24
...

view DSSV





Nhược điểm của View

- ❖ **Hạn chế trong cập nhật dữ liệu**
- ❖ **Giới hạn về cấu trúc**
- ❖ **Chậm hơn**



```
CREATE VIEW view_donhang
```

```
AS
```

```
SELECT dondathang.sohoadon, makhachhang, manhanvien,  
ngaydathang, ngaygiaohang, ngaychuyenhang,  
noigiaohang, mahang,  
giaban, soluong, mucgiamgia  
FROM dondathang INNER JOIN chitietdathang  
ON dondathang.sohoadon = chitietdathang.sohoadon
```

Cơ sở dữ

- Có thể thông qua khung nhìn này để bổ sung dữ liệu cho bảng DONDATHANG được không?
- Có thể thông qua khung nhìn này để bổ sung dữ liệu cho bảng CHITIETDATHANG được không?

```
CREATE VIEW view_donhang
```

```
AS
```

```
SELECT dondathang.sohoadon, makhachhang, manhanvien,  
ngaydathang, ngaygiaoohang, ngaychuyenhang,  
noigiaoohang, mahang,  
giaban*soluong as thanhtien,  
mucgiamgia  
FROM dondathang INNER JOIN chitietdathang  
ON dondathang.sohoadon = chitietdathang.sohoadon
```

- Cơ sở
- Có thể thông qua khung nhìn này để xoá hay cập nhật dữ liệu trong bảng DONDATHANG được không?
 - Có thể thông qua khung nhìn này để cập nhật dữ liệu trong bảng CHITIETDATHANG được không?



BÀI TẬP

Xét bảng:

MatHang(MaMH, TenMH, DonGia, SoLuongTon)

HoaDon(MaHD, NgayLap)

CTHD(MaHD, MaMH, SoLuong, DonGiaBan)

Viết lệnh tạo bảng và có các ràng buộc sau:

1. Số lượng tồn trong bảng MatHang có giá trị mặc định 0
2. Tạo View v2 dùng để hiển thị tiền bán trên từng hóa đơn gồm các thông tin: MaHD, NgayLap, Tổng tiền
3. Tạo View v3 liệt kê danh sách các mặt hàng không bán được trong tháng hiện hành



Vật chất hóa view (View vật chất)

- ❖ View phân giải chậm, nhất là khi truy xuất thường xuyên.
- ❖ View vật chất lưu view như một bảng tạm khi truy xuất lần đầu tiên
- ❖ Do đó, view vật chất chạy nhanh hơn.
- ❖ Nhưng: phải xem xét đến việc cập nhật view vật chất (bảo trì) khi bảng nền của nó được cập nhật.

- ❖ Việc bảo trì view giúp cho view có thông tin hiện hành.

- ❖ Xét view sau:

```
CREATE VIEW StaffPropRent(staffNo)
AS SELECT DISTINCT staffNo
        FROM PropertyForRent
        WHERE branchNo = 'B003' AND
              rent > 400;
```

Table 6.8 Data for view StaffPropRent.

staffNo
SG37
SG14



Vật chất hóa view (View vật chất)

- ❖ Nếu chèn hàng vào PropertyForRent có rent \leq 400 thì view không đổi
- ❖ Nếu chèn hàng vào cho tài sản PG24 tại B003 với staffNo = SG19 và rent = 550, thì hàng đó sẽ xuất hiện trong view vật chất
- ❖ Nếu chèn hàng vào cho tài sản PG54 tại B003 với staffNo = SG37 và rent = 450, thì không cần thêm hàng cho view vật chất.
- ❖ Nếu xóa tài sản PG24, hàng dữ liệu này sẽ biến khỏi view
- ❖ Nếu xóa tài sản PG54, thì hàng dữ liệu cho tài sản PG37 không nên bị xóa (bởi vì còn tồn tại tài sản PG21)



Giao tác (Transactions)

- ❖ SQL định nghĩa mô hình giao tác dựa vào COMMIT và ROLLBACK.
- ❖ Giao tác là một đơn vị logic của công việc, các lệnh SQL về giao tác phải bảo đảm việc giao tác thực hiện xong công việc hoặc là phục hồi lại trạng thái trước giao tác.
- ❖ Một giao tác của SQL tự động khởi động với câu lệnh có tính khởi động giao tác (như lệnh SELECT, INSERT).
- ❖ Các thay đổi do giao tác gây ra bị dấu đi cho đến khi hoàn thành giao tác



Giao tác (Transactions)

- ❖ **Giao tác có thể kết thúc trong 4 trường hợp sau:**
 - COMMIT kết thúc thành công, dữ liệu sẽ được cập nhật
 - ROLLBACK hủy bỏ giao tác, hoàn lại trạng thái trước khi giao tác làm cho thay đổi
 - Trong lập trình với SQL, các chương trình có thể kết thúc giao tác thành công, kể cả khi chưa thực hiện lệnh COMMIT.
 - Trong lập trình với SQL, nếu chương trình bị lỗi giữa giao tác thì nên hủy giao tác.



Giao tác (Transactions)

- ❖ Các giao tác SQL không được lồng nhau.
- ❖ Lệnh SET TRANSACTION cấu hình giao tác:

SET TRANSACTION

[READ ONLY | READ WRITE] |

[ISOLATION LEVEL READ UNCOMMITTED |

**READ COMMITTED|REPEATABLE READ
|SERIALIZABLE]**



Ràng buộc toàn vẹn kiểu trung gian và trì hoãn

- ❖ Không phải lúc nào các ràng buộc cũng kiểm tra tức thời, có khi phải chờ đến lúc giao tác ủy thác (commit)
- ❖ Các ràng buộc có thể được định nghĩa là INITIALLY IMMEDIATE hay INITIALLY DEFERRED lúc đầu giao tác.
- ❖ Cũng có thể thay đổi lại kiểu ràng buộc dùng [NOT] DEFERRABLE.
- ❖ Kiểu mặc định là INITIALLY IMMEDIATE.



Ràng buộc toàn vẹn kiểu trung gian và trì hoãn

- ❖ Lệnh SET CONSTRAINTS cho giao tác hiện hành:

SET CONSTRAINTS

{ALL | constraintName [, . . .]}

{DEFERRED | IMMEDIATE}



Kiểm soát truy xuất – Danh hiệu về quyền và quyền sở hữu

- ❖ Danh hiệu về quyền là danh hiệu bình thường trong SQL dùng để nhận diện user, thường đi kèm là password.
- ❖ Dùng để xác định quyền trong một số thao tác trên một số đối tượng
- ❖ Mỗi đối tượng tạo ra trong SQL có một chủ sở hữu, như là đã định ra ở mệnh đề AUTHORIZATION của lược đồ csdl.



Đặc quyền (Privileges)

- ❖ Các thao tác user được phép thực hiện trên các view hay các quan hệ nền:

SELECT Lấy dữ liệu

INSERT Chèn hàng dữ liệu

UPDATE Sửa dữ liệu trên các hàng đã có

DELETE Xóa các hàng dữ liệu

REFERENCES Tham khảo dữ liệu để bảo đảm ràng buộc toàn vẹn

USAGE Dùng domains, collations, character sets, và translations.



Lệnh ban quyền GRANT

GRANT {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
TO {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]

- ❖ *PrivilegeList* gồm một hay nhiều quyền cách nhau bằng dấu phẩy
- ❖ **ALL PRIVILEGES** ban tất cả quyền cho user



Lệnh ban quyền GRANT

- ❖ PUBLIC cho quyền truy xuất với tất cả user (hiện tại và tương lai).
- ❖ *ObjectName* có thể là base table, view, domain, character set, collation hay translation.
- ❖ WITH GRANT OPTION cho phép thông qua quyền

Cho Manager đầy đủ quyền trên bảng Staff table:

```
GRANT ALL PRIVILEGES  
ON Staff  
TO Manager WITH GRANT OPTION;
```

Cho user Personnel và Director quyền SELECT và UPDATE trên cột salary của bảng Staff:

```
GRANT SELECT, UPDATE (salary)  
ON Staff  
TO Personnel, Director;
```



Ví dụ 7.9 – Ban quyền PUBLIC

Cho tất cả user quyền SELECT trên bảng Branch:

```
GRANT SELECT  
ON Branch  
TO PUBLIC;
```



Lệnh thu lấy quyền REVOKE

- ❖ REVOKE lấy quyền đã được ban trước đó nhờ GRANT.

```
REVOKE [GRANT OPTION FOR]
{PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
[RESTRICT | CASCADE]
```

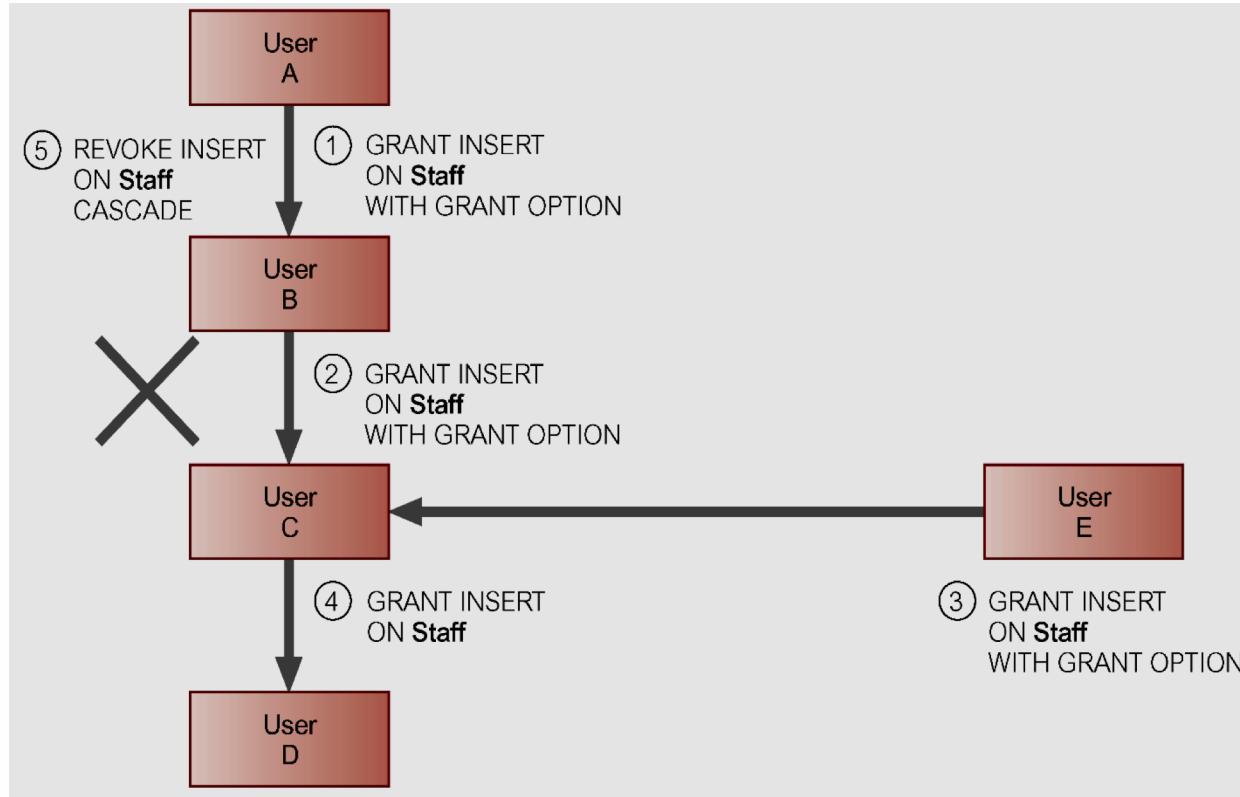
- ❖ ALL PRIVILEGES lấy lại tất cả quyền đã ban



Lệnh thu hồi quyền REVOKE

- ❖ GRANT OPTION FOR cho phép các quyền được thông qua nhờ WITH GRANT OPTION của lệnh GRANT được lấy lại một cách riêng lẻ.
- ❖ REVOKE không thực hiện được nếu trên đối tượng bị cấm, như là các view trừ khi có dùng CASCADE
- ❖ Quyền do user này ban cho user khác không bị ảnh hưởng lẫn nhau

Lệnh thu hồi quyền REVOKE





Ví dụ 7.10/11 - Lệnh lấy quyền REVOKE

Lấy quyền SELECT trên bảng Branch của tất cả user:

```
REVOKE SELECT  
ON Branch  
FROM PUBLIC;
```

Lấy tất cả quyền đã cho Director trên bảng Staff :

```
REVOKE ALL PRIVILEGES  
ON Staff  
FROM Director;
```



TRIGGER

- ❖ Trigger là một tập các câu lệnh
 - Được đặt tên và lưu trữ dưới dạng đã biên dịch
 - “Thức giấc” khi được kích hoạt bởi một (số) sự kiện nhất định

- ❖ Khi “thức giấc”, trigger sẽ xem xét thực hiện các câu lệnh tương ứng



Khai báo ràng buộc (TRIGGER)

- Cú pháp:

```
CREATE [OR REPLACE] TRIGGER tên-trigger  
ON tên Table  
{FOR | AFTER | INSTEADOF} {INSERT|DELETE|UPDATE}  
AS phát biểu SQL  
(Raiserror, rollback tran)
```



Khai báo ràng buộc (TRIGGER)

- Từ khóa REPLACE để tự động xóa và tạo mới trigger nếu trigger đó đã tồn tại. Ví dụ: REPLACE TRIGGER Tên-Trigger.
- Table_name để chỉ đến tên của table muốn tạo trigger.
- INSERT | DELETE | UPDATE ứng với sự kiện tác động lên table để trigger tự động thi hành khi sự kiện đó xảy ra.



Khai báo ràng buộc (TRIGGER)

- Đối với tham số For | After thì Trigger sẽ được gọi sau khi có thao tác Insert hoặc Update. Thứ tự thực hiện là từ Database rồi đến bảng.
- Tham số Instead of thì Trigger sẽ bỏ qua việc tác động tới CSDL, thay vào đó nó thực hiện việc lưu dữ liệu vào bảng Inserted khi có thao tác Insert, lưu dữ liệu vào bảng Deleted đối với thao tác Delete



Khai báo ràng buộc (TRIGGER)

Chú ý khi tạo trigger:

- DDL không được dùng trong phần thân của trigger.
- Không cho phép các lệnh quản lý giao tác (COMMIT, ROLLBACK, SAVEPOINT) trong phần thân của trigger.
- Nếu trigger gọi một chương trình con thì chương trình con đó không được chứa các lệnh quản lý giao tác.



Khai báo ràng buộc (TRIGGER)

- ❖ Thao tác trigger: DISABLE và ENABLE:

- Lệnh disable một trigger

ALTER TRIGGER tên-trigger DISABLE;

- Để disable tất cả các trigger liên quan đến một table cụ thể, dùng lệnh:

ALTER TABLE table_name DISABLE ALL TRIGGERS;

- Lệnh enable một trigger

ALTER TRIGGER trigger_name ENABLE;

- Để enable tất cả các trigger liên quan đến một table cụ thể, dùng lệnh:

ALTER TABLE table_name ENABLE ALL TRIGGERS;

- ❖ Cú pháp xóa trigger: DROP TRIGGER Tên-trigger;



VÍ DỤ

- ❖ Tạo trigger thông báo mỗi khi có thao tác thay đổi bảng PROJECT

```
IF OBJECT_ID('tr_PROJECT') IS NOT NULL
    DROP TRIGGER tr_PROJECT
GO

CREATE TRIGGER tr_PROJECT
ON PROJECT
AFTER INSERT, UPDATE, DELETE
AS RAISERROR('Notify PROJECT Relation',16,10)
GO

UPDATE PROJECT      Msg 50000, Level 16, State 10, Procedure tr_PROJECT, Line 5
SET DNum=3          Notify PROJECT Relation
WHERE PNumber=2     (1 row(s) affected)
GO
```



ví DỤ

- ❖ Tạo trigger kiểm tra từ chối thêm mới nhân viên dưới 18 tuổi

```
IF OBJECT_ID('tr_EMPLOYEE') IS NOT NULL
    DROP TRIGGER tr_EMPLOYEE
GO

CREATE TRIGGER tr_EMPLOYEE
ON EMPLOYEE AFTER INSERT
AS
DECLARE @birthYear INT
    SELECT @birthYear=YEAR(i.EBirthdate)
    FROM inserted i
IF YEAR(GETDATE())-@birthYear < 18
BEGIN
    RAISERROR('We cannot sign a contact with under
              18-year-old employee',16,10)
    ROLLBACK TRANSACTION
END
GO
```





VÍ DỤ

- ❖ Tạo trigger kiểm tra từ chối thêm mới nhân viên dưới 18 tuổi

```
INSERT INTO
    EMPLOYEE
VALUES
    (30121050204, N'Nguyen Thanh Chanh', 30000, 'F',
     Convert(datetime, '17/02/2008', 103),
     Convert(datetime, '15/06/2010', 103), 1, 30121050037)
GO
```

```
Msg 50000, Level 16, State 10, Procedure tr_EMPLOYEE, Line 9
We cannot sign a contact with under 18-year-old employee
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```



VÍ DỤ

- ❖ Tạo trigger kiểm tra sự thay đổi Dnum trong PROJECT

```
CREATE TRIGGER tr_PROJECT2
ON PROJECT AFTER UPDATE
AS
    DECLARE @old_deptNos INT
    DECLARE @new_deptNos INT
    SELECT @old_deptNos=DeptNo FROM deleted
    SELECT @new_deptNos=DeptNo FROM inserted
    PRINT 'tr_PROJECT2 said:'
    IF @old_deptNos = @new_deptNos
        PRINT 'No change in DeptNo'
    ELSE
        PRINT 'DeptNo (' + convert(varchar,@old_deptNos) + ')'
        PRINT 'was changed to (' + convert(varchar,@new_deptNos)+')'
    PRINT ''
GO
```

ví DỤ

```
UPDATE PROJECT  
SET DNum=3  
WHERE PNumber=2  
GO
```

```
tr_PROJECT2 said:  
DNum (5) was changed to (3)
```

```
UPDATE PROJECT  
SET DNum=3  
WHERE PNumber=2  
GO
```

```
tr_PROJECT2 said:  
No change in DNum
```