

CHƯƠNG 3: NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU	2
3.1 Tạo bảng dữ liệu	2
3.1.1 Ràng buộc CHECK.....	5
3.1.2 Ràng buộc PRIMARY KEY	7
3.1.3 Ràng buộc UNIQUE.....	9
3.1.4 Ràng buộc FOREIGN KEY	9
3.2 Sửa đổi định nghĩa bảng	12
3.3 Xoá bảng	14
3.4 Khung nhìn	15
3.4.1 Tạo khung nhìn	17
3.4.2 Cập nhật, bổ sung và xoá dữ liệu thông qua khung nhìn	19
3.4.3 Sửa đổi khung nhìn	22
3.4.4 Xoá khung nhìn.....	23

Chương 3

NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU

Các câu lệnh SQL đã đề cập đến trong chương 3 được sử dụng nhằm thực hiện các thao tác bổ sung, cập nhật, loại bỏ và xem dữ liệu. Nhóm các câu lệnh này được gọi là ngôn ngữ thao tác dữ liệu (DML). Trong chương này, chúng ta sẽ tìm hiểu nhóm các câu lệnh được sử dụng để định nghĩa và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... và được gọi là ngôn ngữ định nghĩa dữ liệu (DDL).

Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

- CREATE: định nghĩa và tạo mới đối tượng CSDL.
- ALTER: thay đổi định nghĩa của đối tượng CSDL.
- DROP: Xóa đối tượng CSDL đã có.

3.1 Tạo bảng dữ liệu

Như đã nói đến ở chương 1, bảng dữ liệu là cấu trúc có vai trò quan trọng nhất trong cơ sở dữ liệu quan hệ. Toàn bộ dữ liệu của cơ sở dữ liệu được tổ chức trong các bảng, những bảng này có thể là những bảng hệ thống được tạo ra khi tạo lập cơ sở dữ liệu, và cũng có thể là những bảng do người sử dụng định nghĩa.

masv	hodem	ten	ngaysinh	gioitinh	noisinh	malop
22150001	Nguyễn Tự	Ánh	2002-12-29	Nữ	HCM	TC1501
22150002	Lưu Quang	Bảo	2002-08-15	Nam	Đồng Tháp	TC1501
22150003	Lê Văn	Quyền	2001-10-18	Nam	Vĩnh Long	TC1501
22150004	Vương Trung	Quân	2002-11-12	Nam	HCM	TC1501
22150005	Lê Thị Ngọc	Trình	2002-10-18	Nữ	Long An	TC1501
22150006	Nguyễn Thị Thúy	Kiều	2002-06-29	Nữ	Nam Định	TC1501
22151001	Nguyễn Nhật	An	2002-07-19	Nữ	Tây Ninh	VL1501
22151002	Phạm Thế	Bảo	2002-12-09	Nam	HCM	VL1501
22151003	Trương Ngọc	Hào	2001-12-11	Nam	Tiền Giang	VL1501
22151004	Phạm Nhật	Quang	2002-09-18	Nam	HCM	VL1501

Hình 3.1 Bảng trong cơ sở dữ liệu quan hệ

Trong các bảng, dữ liệu được tổ chức dưới dạng các dòng và cột. Mỗi một dòng là một bản ghi duy nhất trong bảng và mỗi một cột là một trường. Các bảng trong cơ sở

dữ liệu được sử dụng để biểu diễn thông tin, lưu giữ dữ liệu về các đối tượng trong thế giới thực và/hoặc mối quan hệ giữa các đối tượng. Bảng trong hình 3.1 bao gồm 10 bản ghi và 7 trường là MSSV, HODEM, TEN, NGAYSINH, GIOITINH, NOISINH và MALOP.

Câu lệnh CREATE TABLE được sử dụng để định nghĩa một bảng dữ liệu mới trong cơ sở dữ liệu. Khi định nghĩa một bảng dữ liệu mới, ta cần phải xác định được các yêu cầu sau đây:

- Bảng mới được tạo ra sử dụng với mục đích gì và có vai trò như thế nào trong cơ sở dữ liệu.
- Cấu trúc của bảng bao gồm những trường (cột) nào, mỗi một trường có ý nghĩa như thế nào trong việc biểu diễn dữ liệu, kiểu dữ liệu của mỗi trường là gì và trường đó có cho phép nhận giá trị NULL hay không.
- Những trường nào sẽ tham gia vào khóa chính của bảng. Bảng có quan hệ với những bảng khác hay không và nếu có thì quan hệ như thế nào.
- Trên các trường của bảng có tồn tại những ràng buộc về khuôn dạng, điều kiện hợp lệ của dữ liệu hay không; nếu có thì sử dụng ở đâu và như thế nào.

Câu lệnh CREATE TABLE có cú pháp như sau

```
CREATE TABLE tên_bảng
(
    tên_cột    thuộc_tính_cột    các_ràng_buộc
    [, ...
    , tên_cột_n thuộc_tính_cột_n    các_ràng_buộc_cột_n
    [, các_ràng_buộc_trên_bảng]
)
```

Trong đó:

<i>tên_bảng</i>	Tên của bảng cần tạo. Tên phải tuân theo qui tắc định danh và không được vượt quá 128 ký tự.
<i>tên_cột</i>	Là tên của cột (trường) cần định nghĩa, tên cột phải tuân theo quy tắc định danh và không được trùng nhau trong mỗi một bảng. Mỗi một bảng phải có ít nhất một cột. Nếu bảng có nhiều cột thì định nghĩa của các cột (tên cột, thuộc tính và các ràng buộc) phải phân cách nhau bởi dấu phẩy.
<i>thuộc_tính_cột</i>	Các thuộc tính bao gồm: <ul style="list-style-type: none"> • Kiểu dữ liệu của cột. Đây là thuộc tính bắt buộc phải có đối với mỗi cột. • Giá trị mặc định của cột: là giá trị được tự động gán cho cột nếu như người sử dụng không nhập dữ liệu cho cột một cách tường minh. Mỗi một cột chỉ có thể có nhiều nhất một giá trị mặc định.

- Cột có tính chất IDENTITY hay không? tức là giá trị của cột có được tự động tăng mỗi khi có bản ghi mới được bổ sung hay không. Tính chất này chỉ có thể sử dụng đối với các trường kiểu số.
- Cột có chấp nhận giá trị NULL hay không.

Ví dụ 3.1: Khai báo dưới đây định nghĩa cột STT có kiểu dữ liệu là *int* và cột có tính chất IDENTITY:

```
stt INT IDENTITY
```

hay định nghĩa cột *NGAY* có kiểu *datetime* và không cho phép chấp nhận giá trị *NULL*:

```
ngay DATETIME NOT NULL
```

và định nghĩa cột *SOLUONG* kiểu *int* và có giá trị mặc định là 0:

```
soluong INT DEFAULT (0)
```

các_ràng_buộc

Các ràng buộc được sử dụng trên mỗi cột hoặc trên bảng nhằm các mục đích sau:

- Quy định khuôn dạng hay giá trị dữ liệu được cho phép trên cột (chẳng hạn qui định tuổi của một học sinh phải lớn hơn 6 và nhỏ hơn 20, số điện thoại phải là một chuỗi bao gồm 6 chữ số,...). Những ràng buộc kiểu này được gọi là ràng buộc CHECK.
- Đảm bảo tính toàn vẹn dữ liệu trong một bảng và toàn vẹn tham chiếu giữa các bảng trong cơ sở dữ liệu. Những loại ràng buộc này nhằm đảm bảo tính đúng của dữ liệu như: số chứng minh nhân dân của mỗi một người phải duy nhất, nếu sinh viên học một lớp nào đó thì lớp đó phải tồn tại,... Liên quan đến những loại ràng buộc này bao gồm các ràng buộc PRIMARYKEY (khóa chính), UNIQUE (khóa dự tuyển) và FOREIGN KEY (khóa ngoài).

Các loại ràng buộc này sẽ được trình bày chi tiết hơn ở phần sau.

Ví dụ 3.2: Câu lệnh dưới đây định nghĩa bảng NHANVIEN với các trường MANV (mã nhân viên), HOTEN (họ và tên), NGAYSINH (ngày sinh của nhân viên), DIENTHOAI (điện thoại) và HSLUONG (hệ số lương)

```
CREATE TABLE nhanvien
(
    manv      NVARCHAR(10) NOT NULL,
    hoten     NVARCHAR(50) NOT NULL,
    ngaysinh  DATETIME      NULL,
    dienthoai NVARCHAR(10) NULL,
    hsluong   DECIMAL(3,2) DEFAULT (1.92)
)
```

Trong câu lệnh trên, trường MANV và HOTEN của bảng NHANVIEN không được NULL (tức là bắt buộc phải có dữ liệu), trường NGAYSINH và DIENTHOAI sẽ nhận giá trị NULL nếu ta không nhập dữ liệu cho chúng còn trường HSLUONG sẽ nhận giá trị mặc định là 1.92 nếu không được nhập dữ liệu.

Nếu ta thực hiện các câu lệnh dưới đây sau khi thực hiện câu lệnh trên để bổ sung dữ liệu cho bảng NHANVIEN

```
INSERT INTO nhanvien
VALUES('NV01','Le Van A','2/4/75','886963',2.14)
INSERT INTO nhanvien(manv,hoten)
VALUES('NV02','Mai Thi B')
INSERT INTO nhanvien(manv,hoten,dienthoai)
VALUES('NV03','Tran Thi C','849290')
```

Ta sẽ có được dữ liệu trong bảng NHANVIEN như sau:

MANV	HOTEN	NGAYSINH	DIENTHOAI	HSLUONG
NV01	Le Van A	1975-02-04 00:00:00.000	886963	2.14
NV02	Mai Thi B	NULL	NULL	1.92
NV03	Tran Thi C	NULL	849290	1.92

3.1.1 Ràng buộc CHECK

Ràng buộc CHECK được sử dụng nhằm chỉ định điều kiện hợp lệ đối với dữ liệu. Mỗi khi có sự thay đổi dữ liệu trên bảng (INSERT, UPDATE), những ràng buộc này sẽ được sử dụng nhằm kiểm tra xem dữ liệu mới có hợp lệ hay không.

Ràng buộc CHECK được khai báo theo cú pháp như sau:

```
[CONSTRAINT tên_ràng_buộc]
CHECK (điều_kiện)
```

Trong đó, điều_kiện là một biểu thức logic tác động lên cột nhằm quy định giá trị hoặc khuôn dạng dữ liệu được cho phép. Trên mỗi một bảng cũng như trên mỗi một cột có thể có nhiều ràng buộc CHECK.

Ví dụ 3.3: Câu lệnh dưới đây tạo bảng DIEMTOTNGHIEP trong đó quy định giá trị của cột DIEMVAN và DIEMTOAN phải lớn hơn hoặc bằng 0 và nhỏ hơn hoặc bằng 10.

```
CREATE TABLE diemtotnghiep
(
    hoten          NVARCHAR(30) NOT NULL,
    ngaysinh       DATETIME,
    diemvan        DECIMAL(4,2)
                  CONSTRAINT chk_diemvan
                  CHECK(diemvan>=0 AND diemvan<=10),
    diemtoan       DECIMAL(4,2)
                  CONSTRAINT chk_diemtoan
                  CHECK(diemtoan>=0 AND diemtoan<=10),
)
```

Như vậy, với định nghĩa như trên của bảng DIEMTOTNGHIEP, các câu lệnh dưới đây là hợp lệ:

```
INSERT INTO diemtotnghiep(hoten,diemvan,diemtoan)
VALUES('Le Thanh Hoang',9.5,2.5)
INSERT INTO diemtotnghiep(hoten,diemvan)
VALUES('Hoang Thi Mai',2.5)
```

còn câu lệnh dưới đây là không hợp lệ:

```
INSERT INTO diemtotnghiep(hoten,diemvan,diemtoan)
VALUES('Tran Van Hanh',6,10.5)
```

do cột DIEMTOAN nhận giá trị 10.5 không thỏa mãn điều kiện của ràng buộc.

Trong ví dụ trên, các ràng buộc được chỉ định ở phần khai báo của mỗi cột. Thay vì chỉ định ràng buộc trên mỗi cột, ta có thể chỉ định các ràng buộc ở mức bảng bằng cách khai báo các ràng buộc sau khi đã khai báo xong các cột trong bảng.

Ví dụ 3.4: Câu lệnh

```
CREATE TABLE lop
(
    malop          NVARCHAR(10)          NOT NULL ,
    tenlop         NVARCHAR(30)          NOT NULL ,
    khoa          SMALLINT              NULL ,
    hedaotao       NVARCHAR(25)          NULL
```

```

        CONSTRAINT chk_lop_hedaotao
        CHECK (hedaotao IN ('chính quy','tài chức')),
namnhaphoc INT NULL
        CONSTRAINT chk_lop_namnhaphoc
        CHECK (namnhaphoc<=YEAR(GETDATE())) ,
makhoa NVARCHAR(5)
)

```

có thể được viết lại như sau:

```

CREATE TABLE lop
(
malop NVARCHAR(10) NOT NULL ,
tenlop NVARCHAR(30) NOT NULL ,
khoa SMALLINT NULL ,
hedaotao NVARCHAR(25) NULL,
namnhaphoc INT NULL ,
makhoa NVARCHAR(5) ,
CONSTRAINT chk_lop
CHECK (namnhaphoc<=YEAR(GETDATE())) AND
        hedaotao IN ('chính quy','tài chức'))
)

```

3.1.2 Ràng buộc PRIMARY KEY

Ràng buộc PRIMARY KEY được sử dụng để định nghĩa khóa chính của bảng. Khóa chính của một bảng là một hoặc một tập nhiều cột mà giá trị của chúng là duy nhất trong bảng. Hay nói cách khác, giá trị của khóa chính sẽ giúp cho ta xác định được duy nhất một dòng (bản ghi) trong bảng dữ liệu. Mỗi một bảng chỉ có thể có duy nhất một khóa chính và bản thân khóa chính không chấp nhận giá trị NULL. Ràng buộc PRIMARY KEY là cơ sở cho việc đảm bảo tính toàn vẹn thực thể cũng như toàn vẹn tham chiếu.

Để khai báo một ràng buộc PRIMARY KEY, ta sử dụng cú pháp như sau:

```

[CONSTRAINT tên_ràng_buộc]
PRIMARY KEY [(danh_sách_cột)]

```

Nếu khóa chính của bảng chỉ bao gồm đúng một cột và ràng buộc PRIMARY KEY được chỉ định ở mức cột, ta không cần thiết phải chỉ định danh sách cột sau từ khóa PRIMARY KEY. Tuy nhiên, nếu việc khai báo khóa chính được tiến hành ở mức bảng(sử dụng khi số lượng các cột tham gia vào khóa là từ hai trở lên) thì bắt buộc phải chỉ định danh sách cột ngay sau từ khóa PRIMARY KEY và tên các cột được phân cách nhau bởi dấu phẩy.

Ví dụ 3.5: Câu lệnh dưới đây định nghĩa bảng SINHVIEN với khóa chính là MASV

```
CREATE TABLE sinhvien
(
    masv          NVARCHAR(10)
                CONSTRAINT pk_sinhvien_masv PRIMARY KEY,
    hodem         NVARCHAR(25)    NOT NULL ,
    ten           NVARCHAR(10)    NOT NULL ,
    ngaysinh      DATETIME,
    gioitinh      BIT,
    noisinh       NVARCHAR(255),
    malop         NVARCHAR(10)
)
```

Với bảng vừa được tạo bởi câu lệnh ở trên, nếu ta thực hiện câu lệnh:

```
INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)
VALUES ('2217009','Lê Hoàng Phương','Anh',0,'KTPM1601')
```

một bản ghi mới sẽ được bổ sung vào bảng này. Nhưng nếu ta thực hiện tiếp câu lệnh:

```
INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)
VALUES ('2217009','Lê Huy','Đan',1,'KTPM1601')
```

thì câu lệnh này sẽ bị lỗi do trùng giá trị khóa với bản ghi đã có.

Ví dụ 3.6: Câu lệnh dưới đây tạo bảng DIEMTHI với khóa chính là tập bao gồm hai cột MAMONHOC và MASV

```
CREATE TABLE diemthi
(
    mamonhoc      NVARCHAR(10)    NOT NULL ,
    masv          NVARCHAR(10)    NOT NULL ,
    diemlan1      NUMERIC(4, 2),
    diemlan2      NUMERIC(4, 2),
    CONSTRAINT pk_diemthi PRIMARY KEY(mamonhoc,masv)
)
```

Lưu ý:

- Mỗi một bảng chỉ có thể có nhiều nhất một ràng buộc PRIMARY KEY.
- Một khóa chính có thể bao gồm nhiều cột nhưng không vượt quá 16 cột.

3.1.3 Ràng buộc UNIQUE

Trên một bảng chỉ có thể có nhiều nhất một khóa chính nhưng có thể có nhiều cột hoặc tập các cột có tính chất như khóa chính, tức là giá trị của chúng là duy nhất trong bảng. Tập một hoặc nhiều cột có giá trị duy nhất và không được chọn làm khóa chính được gọi là khóa phụ (khóa dự tuyển) của bảng. Như vậy, một bảng chỉ có nhiều nhất một khóa chính nhưng có thể có nhiều khóa phụ.

Ràng buộc UNIQUE được sử dụng trong câu lệnh CREATE TABLE để định nghĩa khóa phụ cho bảng và được khai báo theo cú pháp sau đây:

```
[CONSTRAINT tên_ràng_buộc]  
UNIQUE [ (danh_sách_cột) ]
```

Ví dụ 3.7: Giả sử ta cần định nghĩa bảng LOP với khóa chính là cột MALOP nhưng đồng thời lại không cho phép các lớp khác nhau được trùng tên lớp với nhau, ta sử dụng câu lệnh như sau:

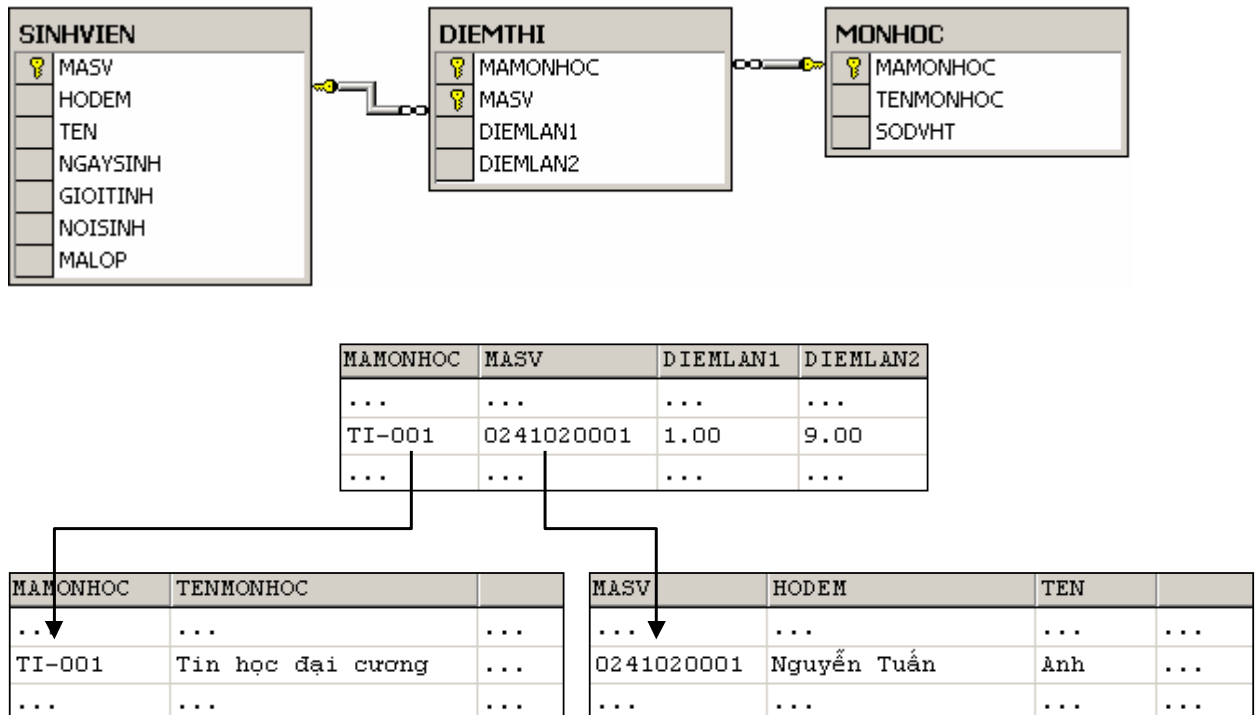
```
CREATE TABLE lop  
(  
    malop          NVARCHAR(10)          NOT NULL,  
    tenlop         NVARCHAR(30)          NOT NULL,  
    khoa          SMALLINT              NULL,  
    hedaotao       NVARCHAR(25)          NULL,  
    namnhaphoc     INT                   NULL,  
    makhoa         NVARCHAR(5) ,  
    CONSTRAINT pk_lop PRIMARY KEY (malop),  
    CONSTRAINT unique_lop_tenlop UNIQUE(tenlop)  
)
```

3.1.4 Ràng buộc FOREIGN KEY

Các bảng trong một cơ sở dữ liệu có mối quan hệ với nhau. Những mối quan hệ này biểu diễn cho sự quan hệ giữa các đối tượng trong thế giới thực. Về mặt dữ liệu, những mối quan hệ được đảm bảo thông qua việc đòi hỏi sự có mặt của một giá trị dữ liệu trong bảng này phải phụ thuộc vào sự tồn tại của giá trị dữ liệu đó ở trong một bảng khác.

Ràng buộc FOREIGN KEY được sử dụng trong định nghĩa bảng dữ liệu nhằm tạo nên mối quan hệ giữa các bảng trong một cơ sở dữ liệu. Một hay một tập các cột trong một bảng được gọi là khóa ngoại, tức là có ràng buộc FOREIGN KEY, nếu giá trị của nó được xác định từ khóa chính (PRIMARY KEY) hoặc khóa phụ (UNIQUE) của một bảng dữ liệu khác.

Hình dưới đây cho ta thấy được mối quan hệ giữa 3 bảng DIEMTHI, SINHVIEN và MONHOC. Trong bảng DIEMTHI, MASV là khóa ngoài tham chiếu đến cột MASV của bảng SINHVIEN và MAMONHOC là khóa ngoài tham chiếu đến cột MAMONHOC của bảng MONHOC.



Hình 3.2 Mối quan hệ giữa các bảng

Với mối quan hệ được tạo ra như hình trên, hệ quản trị cơ sở dữ liệu sẽ kiểm tra tính hợp lệ của mỗi bản ghi trong bảng DIEMTHI mỗi khi được bổ sung hay cập nhật. Một bản ghi bất kỳ trong bảng DIEMTHI chỉ hợp lệ (đảm bảo ràng buộc FOREIGN KEY) nếu giá trị của cột MASV phải tồn tại trong một bản ghi nào đó của bảng SINHVIEN và giá trị của cột MAMONHOC phải tồn tại trong một bản ghi nào đó của bảng MONHOC.

Ràng buộc FOREIGN KEY được định nghĩa theo cú pháp dưới đây:

```
[CONSTRAINT tên_ràng_buộc]
FOREIGN KEY [(danh_sách_cột)]
REFERENCES tên_bảng_tham_chiếu(danh_sách_cột_tham_chiếu)
[ON DELETE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
[ON UPDATE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
```

Việc định nghĩa một ràng buộc FOREIGN KEY bao gồm các yếu tố sau:

- Tên cột hoặc danh sách cột của bảng được định nghĩa tham gia vào khóa ngoài.
- Tên của bảng được tham chiếu bởi khóa ngoài và danh sách các cột được tham chiếu đến trong bảng tham chiếu.
- Cách thức xử lý đối với các bản ghi trong bảng được định nghĩa trong trường hợp các bản ghi được tham chiếu trong bảng tham chiếu bị xóa (ON DELETE) hay cập nhật (ON UPDATE). SQL chuẩn đưa ra 4 cách xử lý:
 - CASCADE: Tự động xóa (cập nhật) nếu bản ghi được tham chiếu bị xóa (cập nhật).
 - NO ACTION: (Mặc định) Nếu bản ghi trong bảng tham chiếu đang được tham chiếu bởi một bản ghi bất kỳ trong bảng được định nghĩa thì bản ghi đó không được phép xóa hoặc cập nhật (đối với cột được tham chiếu).
 - SET NULL: Cập nhật lại khóa ngoài của bản ghi thành giá trị NULL (nếu cột cho phép nhận giá trị NULL).
 - SET DEFAULT: Cập nhật lại khóa ngoài của bản ghi nhận giá trị mặc định (nếu cột có qui định giá trị mặc định).

Ví dụ 3.8: Câu lệnh dưới đây định nghĩa bảng DIEMTHI với hai khóa ngoài trên cột MASV và cột MAMONHOC (giả sử hai bảng SINHVIEN và MONHOC đã được định nghĩa)

```
CREATE TABLE diemthi
(
  mamonhoc  NVARCHAR(10)    NOT NULL ,
  masv      NVARCHAR(10)    NOT NULL ,
  diemlan1  NUMERIC(4, 2),
  diemlan2  NUMERIC(4, 2),
  CONSTRAINT pk_diemthi PRIMARY KEY(mamonhoc,masv),
  CONSTRAINT fk_diemthi_mamonhoc
             FOREIGN KEY(mamonhoc)
             REFERENCES monhoc(mamonhoc)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
  CONSTRAINT fk_diemthi_masv
             FOREIGN KEY(masv)
             REFERENCES sinhvien(masv)
             ON DELETE CASCADE
             ON UPDATE CASCADE
)
```

Lưu ý:

- Cột được tham chiếu trong bảng tham chiếu phải là khóa chính (hoặc là khóa phụ).
- Cột được tham chiếu phải có cùng kiểu dữ liệu và độ dài với cột tương ứng trong khóa ngoài.
- Bảng tham chiếu phải được định nghĩa trước. Do đó, nếu các bảng có mối quan hệ vòng, ta có thể không thể định nghĩa ràng buộc FOREIGN KEY ngay trong câu lệnh CREATE TABLE mà phải định nghĩa thông qua lệnh ALTER TABLE.

3.2 Sửa đổi định nghĩa bảng

Một bảng sau khi đã được định nghĩa bằng câu lệnh CREATE TABLE có thể được sửa đổi thông qua câu lệnh ALTER TABLE. Câu lệnh này cho phép chúng ta thực hiện được các thao tác sau:

- Bổ sung một cột vào bảng.
- Xoá một cột khỏi bảng.
- Thay đổi định nghĩa của một cột trong bảng.
- Xoá bỏ hoặc bổ sung các ràng buộc cho bảng

Cú pháp của câu lệnh ALTER TABLE như sau:

```
ALTER TABLE tên_bảng
  ADD định_nghĩa_cột |
  ALTER COLUMN tên_cột kiểu_dữ_liệu [NULL | NOT NULL] |
  DROP COLUMN tên_cột |
  ADD CONSTRAINT tên_ràng_buộc định_nghĩa_ràng_buộc |
  DROP CONSTRAINT tên_ràng_buộc
```

Ví dụ 3.9: Các ví dụ dưới đây minh họa cho ta cách sử dụng câu lệnh ALTER TABLE trong các trường hợp.

Giả sử ta có hai bảng DONVI và NHANVIEN với định nghĩa như sau:

```
CREATE TABLE donvi
(
  madv          INT          NOT NULL   PRIMARY KEY,
  tendv         NVARCHAR(30) NOT NULL
)
```

```

CREATE TABLE nhanvien
(
manv          NVARCHAR(10)    NOT NULL,
hoten         NVARCHAR(30)    NOT NULL,
ngaysinh      DATETIME,
diachi        CHAR(30)        NOT NULL
)

```

Bổ sung vào bảng NHANVIEN cột DIENTHOAI với ràng buộc CHECK nhằm quy định điện thoại của nhân viên là một chuỗi 6 chữ số:

```

ALTER TABLE nhanvien
ADD
    dienthoai NVARCHAR(6)
    CONSTRAINT chk_nhanvien_dienthoai
    CHECK (dienthoai LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]')

```

Bổ sung thêm cột MADV vào bảng NHANVIEN:

```

ALTER TABLE nhanvien
ADD
    madv INT NULL

```

Định nghĩa lại kiểu dữ liệu của cột DIACHI trong bảng NHANVIEN và cho phép cột này chấp nhận giá trị NULL:

```

ALTER TABLE nhanvien
ALTER COLUMN diachi NVARCHAR(100) NULL

```

Xoá cột ngày sinh khỏi bảng NHANVIEN:

```

ALTER TABLE nhanvien
DROP COLUMN ngaysinh

```

Định nghĩa khóa chính (ràng buộc PRIMARY KEY) cho bảng NHANVIEN là cột MANV:

```

ALTER TABLE nhanvien
ADD
    CONSTRAINT pk_nhanvien PRIMARY KEY (manv)

```

Định nghĩa khóa ngoài cho bảng NHANVIEN trên cột MADV tham chiếu đến cột MADV của bảng DONVI:

```

ALTER TABLE nhanvien
ADD
    CONSTRAINT fk_nhanvien_madv
    FOREIGN KEY (madv) REFERENCES donvi (madv)

```

ON DELETE CASCADE

ON UPDATE CASCADE

Xoá bỏ ràng buộc kiểm tra số điện thoại của nhân viên

```
ALTER TABLE nhanvien
```

```
DROP CONSTRAINT CHK_NHANVIEN_DIENHOTOAI
```

Lưu ý:

- Nếu bổ sung thêm một cột vào bảng và trong bảng đã có ít nhất một bản ghi thì cột mới cần bổ sung phải cho phép chấp nhận giá trị NULL hoặc phải có giá trị mặc định.
- Muốn xoá một cột đang được ràng buộc bởi một ràng buộc hoặc đang được tham chiếu bởi một khóa ngoài, ta phải xoá ràng buộc hoặc khóa ngoài trước sao cho trên cột không còn bất kỳ một ràng buộc và không còn được tham chiếu bởi bất kỳ khóa ngoài nào.
- Nếu bổ sung thêm ràng buộc cho một bảng đã có dữ liệu và ràng buộc cần bổ sung không được thoả mãn bởi các bản ghi đã có trong bảng thì câu lệnh ALTER TABLE không thực hiện được.

3.3 Xoá bảng

Khi một bảng không còn cần thiết, ta có thể xoá nó ra khỏi cơ sở dữ liệu bằng câu lệnh DROP TABLE. Câu lệnh này cũng đồng thời xoá tất cả những ràng buộc, chỉ mục, trigger liên quan đến bảng đó.

Câu lệnh có cú pháp như sau:

```
DROP TABLE tên_bảng
```

Trong các hệ quản trị cơ sở dữ liệu, khi đã xoá một bảng bằng lệnh DROP TABLE, ta không thể khôi phục lại bảng cũng như dữ liệu của nó. Do đó, cần phải cẩn thận khi sử dụng câu lệnh này.

Câu lệnh DROP TABLE không thể thực hiện được nếu bảng cần xoá đang được tham chiếu bởi một ràng buộc FOREIGN KEY. Trong trường hợp này, ràng buộc FOREIGN KEY đang tham chiếu hoặc bảng đang tham chiếu đến bảng cần xoá phải được xoá trước.

Khi một bảng bị xoá, tất cả các ràng buộc, chỉ mục và trigger liên quan đến bảng cũng đồng thời bị xoá theo. Do đó, nếu ta tạo lại bảng thì cũng phải tạo lại các đối tượng này.

Ví dụ 3.10: Giả sử cột MADV trong bảng DONVI đang được tham chiếu bởi khóa ngoài *fk_nhanvien_madv* trong bảng NHANVIEN. Để xoá bảng DONVI ra khỏi cơ sở dữ liệu, ta thực hiện hai câu lệnh sau:

Xoá bỏ ràng buộc *fk_nhanvien_madv* khỏi bảng NHANVIEN:

```
ALTER TABLE nhanvien
DROP CONSTRAINT fk_nhanvien_madv
```

Xoá bảng DONVI:

```
DROP TABLE donvi
```

3.4 Khung nhìn

Các bảng trong cơ sở dữ liệu đóng vai trò là các đối tượng tổ chức và lưu trữ dữ liệu. Như vậy, ta có thể quan sát được dữ liệu trong cơ sở dữ liệu bằng cách thực hiện các truy vấn trên bảng dữ liệu. Ngoài ra, SQL còn cho phép chúng ta quan sát được dữ liệu thông qua việc định nghĩa các khung nhìn.

Một khung nhìn (view) có thể được xem như là một bảng “ảo” trong cơ sở dữ liệu có nội dung được định nghĩa thông qua một truy vấn (câu lệnh SELECT). Như vậy, một khung nhìn trông giống như một bảng với một tên khung nhìn và là một tập bao gồm các dòng và các cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. Thực chất dữ liệu quan sát được trong khung nhìn được lấy từ các bảng thông qua câu lệnh truy vấn dữ liệu.

Hình 3.3 dưới đây minh hoạ cho ta thấy khung nhìn có tên DSSV được định nghĩa thông qua câu lệnh SELECT truy vấn dữ liệu trên hai bảng SINHVIEN và LOP:

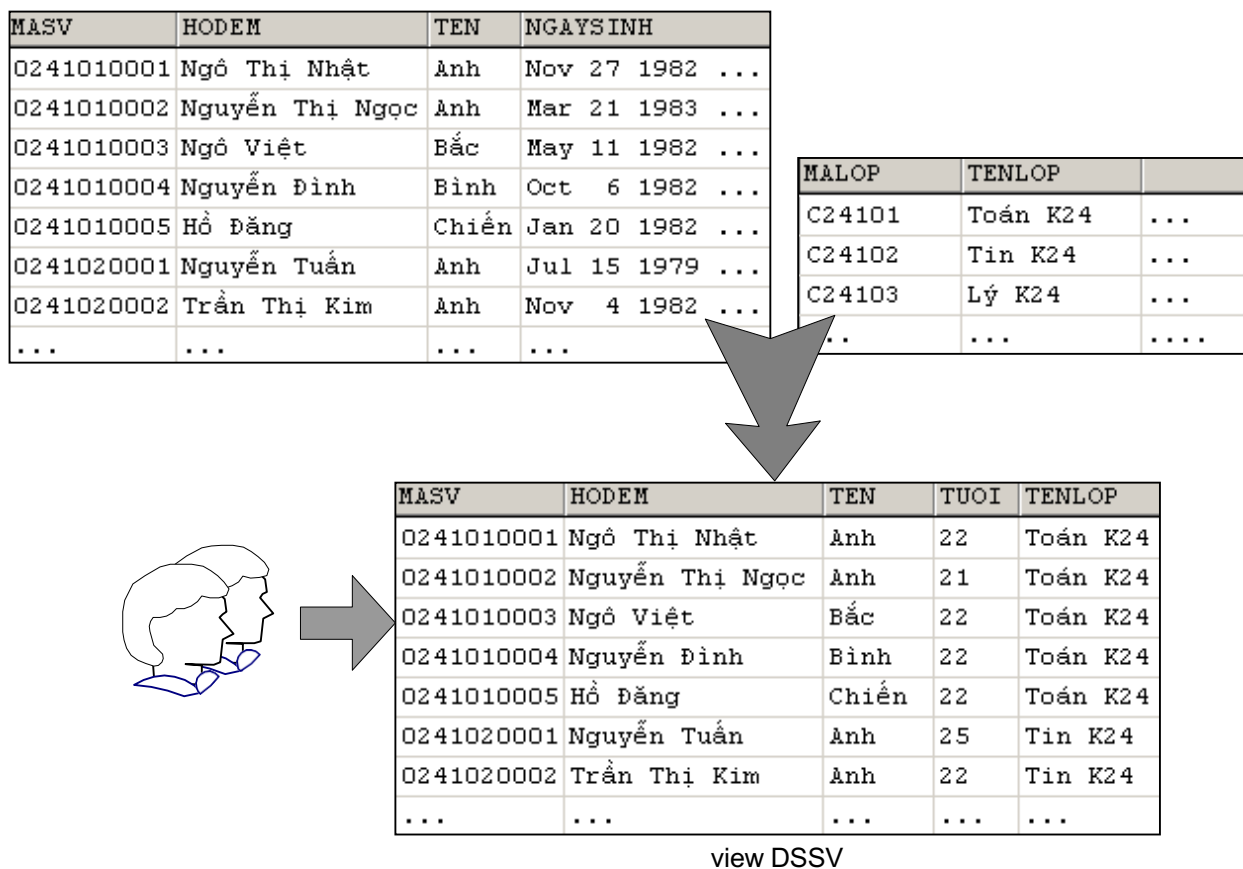
```
SELECT masv, hodem, ten,
       DATEDIFF(YY, ngaysinh, GETDATE()) AS tuoi, tenlop
FROM sinhvien, lop
WHERE sinhvien.malop=lop.malop
```

Khi khung nhìn DSSV đã được định nghĩa, ta có thể sử dụng câu lệnh SELECT để truy vấn dữ liệu từ khung nhìn như đối với các bảng. Khi trong câu truy vấn xuất hiện khung nhìn, hệ quản trị cơ sở dữ liệu sẽ dựa vào định nghĩa của khung nhìn để chuyển yêu cầu truy vấn dữ liệu liên quan đến khung nhìn thành yêu cầu tương tự trên các bảng cơ sở và việc truy vấn dữ liệu được thực hiện bởi yêu cầu tương đương trên các bảng.

Việc sử dụng khung nhìn trong cơ sở dữ liệu đem lại các lợi ích sau đây:

- **Bảo mật dữ liệu:** Người sử dụng được cấp phát quyền trên các khung nhìn với những phần dữ liệu mà người sử dụng được phép. Điều này hạn chế được phần nào việc người sử dụng truy cập trực tiếp dữ liệu.

- **Đơn giản hoá các thao tác truy vấn dữ liệu:** Một khung nhìn đóng vai trò như là một đối tượng tập hợp dữ liệu từ nhiều bảng khác nhau vào trong một “bảng”. Nhờ vào đó, người sử dụng có thể thực hiện các yêu cầu truy vấn dữ liệu một cách đơn giản từ khung nhìn thay vì phải đưa ra những câu truy vấn phức tạp.
- **Tập trung và đơn giản hoá dữ liệu:** Thông qua khung nhìn ta có thể cung cấp cho người sử dụng những cấu trúc đơn giản, dễ hiểu hơn về dữ liệu trong cơ sở dữ liệu đồng thời giúp cho người sử dụng tập trung hơn trên những phần dữ liệu cần thiết.
- **Độc lập dữ liệu:** Một khung nhìn có thể cho phép người sử dụng có được cái nhìn về dữ liệu độc lập với cấu trúc của các bảng trong cơ sở dữ liệu cho dù các bảng cơ sở có bị thay đổi phần nào về cấu trúc.



Hình 3.3 Khung nhìn DSSV với dữ liệu được lấy từ bảng SINHVIEN và LOP

Tuy nhiên, việc sử dụng khung nhìn cũng tồn tại một số nhược điểm sau:

- Do hệ quản trị cơ sở dữ liệu thực hiện việc chuyển đổi các truy vấn trên khung nhìn thành những truy vấn trên các bảng cơ sở nên nếu một khung

nhìn được định nghĩa bởi một truy vấn phức tạp thì sẽ dẫn đến chi phí về mặt thời gian khi thực hiện truy vấn liên quan đến khung nhìn sẽ lớn.

- Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được; hay nói cách khác là dữ liệu trong khung nhìn là chỉ đọc.

3.4.1 Tạo khung nhìn

Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

```
CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)]
AS
    câu_lệnh_SELECT
```

Ví dụ 3.11: Câu lệnh dưới đây tạo khung nhìn có tên DSSV từ câu lệnh SELECT truy vấn dữ liệu từ hai bảng SINHVIEN và LOP

```
CREATE VIEW dssv
AS
    SELECT masv, hodem, ten,
           DATEDIFF(YY, ngaysinh, GETDATE()) AS tuoi, tenlop
    FROM sinhvien, lop
    WHERE sinhvien.malop=lop.malop
```

và nếu thực hiện câu lệnh:

```
SELECT * FROM dssv
```

ta có được kết quả như sau:

MASV	HODEM	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
0241020003	Võ Đức	Ấn	22	Tin K24
0241020004	Nguyễn Công	Bình	25	Tin K24
0241020005	Nguyễn Thanh	Bình	22	Tin K24
...

Nếu trong câu lệnh CREATE VIEW, ta không chỉ định danh sách các tên cột cho khung nhìn, tên các cột trong khung nhìn sẽ chính là tiêu đề các cột trong kết quả của câu lệnh SELECT. Trong trường hợp tên các cột của khung nhìn được chỉ định, chúng phải có cùng số lượng với số lượng cột trong kết quả của câu truy vấn.

Ví dụ 3.12: Câu lệnh dưới đây tạo khung nhìn từ câu truy vấn tương tự như ví dụ trên nhưng có đặt tên cho các cột trong khung nhìn:

```
CREATE VIEW dssv(ma,ho,ten,tuoi,lop)
AS
SELECT masv,hodem,ten,
       DATEDIFF(YY,ngaysinh,GETDATE()),tenlop
FROM sinhvien,lop
WHERE sinhvien.malop=lop.malop
```

và câu lệnh:

```
SELECT * FROM dssv
```

trong trường hợp này có kết quả như sau:

MA	HO	TEN	TUOI	LOP
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
0241020003	Võ Đức	Ấn	22	Tin K24
0241020004	Nguyễn Công	Bình	25	Tin K24
0241020005	Nguyễn Thanh	Bình	22	Tin K24
...

Khi tạo khung nhìn với câu lệnh CREATE VIEW, ta cần phải lưu ý một số nguyên tắc sau:

- Tên khung nhìn và tên cột trong khung nhìn, cũng giống như bảng, phải tuân theo qui tắc định danh.
- Không thể qui định ràng buộc và tạo chỉ mục cho khung nhìn.
- Câu lệnh SELECT với mệnh đề COMPUTE ... BY không được sử dụng để định nghĩa khung nhìn.
- Phải đặt tên cho các cột của khung nhìn trong các trường hợp sau đây:

- Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức là không phải là một tên cột trong bảng cơ sở) và cột đó không được đặt tiêu đề.
- Tồn tại hai cột trong kết quả của câu lệnh SELECT có cùng tiêu đề cột.

Ví dụ 3.13: Câu lệnh dưới đây là câu lệnh sai do cột thứ 4 không xác định được tên cột

```
CREATE VIEW tuoisinhvien
AS
SELECT masv, hodem, ten, DATEDIFF(YY, ngaysinh, GETDATE())
FROM sinhvien
```

3.4.2 Cập nhật, bổ sung và xoá dữ liệu thông qua khung nhìn

Đối với một số khung nhìn, ta có thể tiến hành thực hiện các thao tác cập nhật, bổ sung và xoá dữ liệu. Thực chất, những thao tác này sẽ được chuyển thành những thao tác tương tự trên các bảng cơ sở và có tác động đến những bảng cơ sở.

Về mặt lý thuyết, để có thể thực hiện thao tác bổ sung, cập nhật và xoá, một khung nhìn trước tiên phải thoả mãn các điều kiện sau đây:

- Trong câu lệnh SELECT định nghĩa khung nhìn không được sử dụng từ khóa DISTINCT, TOP, GROUP BY và UNION.
- Các thành phần xuất hiện trong danh sách chọn của câu lệnh SELECT phải là các cột trong các bảng cơ sở. Trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp.

Ngoài những điều kiện trên, các thao tác thay đổi đến dữ liệu thông qua khung nhìn còn phải đảm bảo thoả mãn các ràng buộc trên các bảng cơ sở, tức là vẫn đảm bảo tính toàn vẹn dữ liệu. Ví dụ dưới đây sẽ minh hoạ cho ta thấy việc thực hiện các thao tác bổ sung, cập nhật và xoá dữ liệu thông qua khung nhìn.

Ví dụ 3.14: Xét định nghĩa hai bảng DONVI và NHANVIEN như sau:

```
CREATE TABLE donvi
(
    madv          INT          PRIMARY KEY,
    tendv         NVARCHAR(30) NOT NULL,
    dienthoai     NVARCHAR(10) NULL,
)

CREATE TABLE nhanvien
(
```

```

manv      NVARCHAR(10)  PRIMARY KEY,
hoten     NVARCHAR(30)  NOT NULL,
ngaysinh  DATETIME      NULL,
diachi    NVARCHAR(50)  NULL,
madv      INT           FOREIGN KEY
                        REFERENCES donvi (madv)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
)

```

Giả sử trong hai bảng này đã có dữ liệu như sau:

MADV	TENDV	DIENTHOAI
1	P. Kinh doanh	822321
2	P. Tiếp thị	822012

Bảng DONVI

MANV	HOTEN	NGAYSINH	DIACHI	MADV
NV01	Tran Van A	1975-02-03 00:00:00	77 Tran Phu	1
NV02	Mai Thi B	1977-05-04 00:00:00	34 Nguyen Hue	2
NV03	Nguyen Van C	NULL	NULL	2

Bảng NHANVIEN

Câu lệnh dưới đây định nghĩa khung nhìn NV1 cung cấp các thông tin về mã nhân viên, họ tên và mã đơn vị nhân viên làm việc:

```

CREATE VIEW nv1
AS
    SELECT manv,hoten,madv FROM nhanvien

```


Nếu ta thực hiện câu lệnh

```
INSERT INTO nv1 VALUES ('NV04','Le Thi D',1)
```

Một bản ghi mới sẽ được bổ sung vào bảng NHANVIEN và dữ liệu trong bảng này sẽ là:

MANV	HOTEN	NGAYSINH	DIACHI	MADV
NV01	Tran Van A	1975-02-03 00:00:00	77 Tran Phu	1
NV02	Mai Thi B	1977-05-04 00:00:00	34 Nguyen Hue	2
NV03	Nguyen Van C	NULL	NULL	2
NV04	Le Thi D	NULL	NULL	1

Bản ghi mới



Thông qua khung nhìn này, ta cũng có thể thực hiện thao tác cập nhật và xoá dữ liệu. Chẳng hạn, nếu ta thực hiện câu lệnh:

```
DELETE FROM nv1 WHERE manv='NV04'
```

Thì bản ghi tương ứng với nhân viên có mã NV04 sẽ bị xoá khỏi bảng NHANVIEN.

Nếu trong danh sách chọn của câu lệnh SELECT có sự xuất hiện của biểu thức tính toán đơn giản, thao tác bổ sung dữ liệu thông qua khung nhìn không thể thực hiện được. Tuy nhiên, trong trường hợp này thao tác cập nhật và xoá dữ liệu vẫn có thể có khả năng thực hiện được (hiển nhiên không thể cập nhật dữ liệu đối với một cột có được từ một biểu thức tính toán).

Ví dụ 3.15: Xét khung nhìn NV2 được định nghĩa như sau:

```
CREATE VIEW nv2
AS
SELECT manv,hoten,YEAR(ngaysinh) AS namsinh,madv
FROM nhanvien
```

Đối với khung nhìn NV2, ta không thể thực hiện thao tác bổ sung dữ liệu nhưng có thể cập nhật hoặc xoá dữ liệu trên bảng thông qua khung nhìn này. Câu lệnh dưới đây là không thể thực hiện được trên khung nhìn NV2

```
INSERT INTO nv2 (manv,hoten,madv)
VALUES ('NV05','Le Van E',1)
```

Nhưng câu lệnh:

```
UPDATE nv2 SET hoten='Le Thi X' WHERE manv='NV04'
```

hoặc câu lệnh

```
DELETE FROM nv2 WHERE manv='NV04'
```

lại có thể thực hiện được và có tác động đối với dữ liệu trong bảng NHANVIEN.

Trong trường hợp khung nhìn được tạo ra từ một phép nối (trong hoặc ngoài) trên nhiều bảng, ta có thể thực hiện được thao tác bổ sung hoặc cập nhật dữ liệu nếu thao tác này chỉ có tác động đến đúng một bảng cơ sở (câu lệnh DELETE không thể thực hiện được trong trường hợp này).

Ví dụ 3.16: Với khung nhìn được định nghĩa như sau:

```
CREATE VIEW nv3
AS
SELECT manv,hoten,ngaysinh, diachi,nhanvien.madv
AS noilamviec,donvi.madv,tendv,dienthoai
```

```
FROM nhanvien FULL OUTER JOIN donviON
      nhanvien.madv=donvi.madv
```

Câu lệnh:

```
INSERT INTO nv3 (manv,hoten,noilamviec)
VALUES ('NV05','Le Van E',1)
```

sẽ bổ sung thêm vào bảng NHANVIEN một bản ghi mới. Hoặc câu lệnh:

```
INSERT INTO nv3 (madv,tendv) VALUES (3,'P. Ke toan')
```

bổ sung thêm vào bảng DONVI một bản ghi do cả hai câu lệnh này chỉ có tác động đến đúng một bảng cơ sở.

Câu lệnh dưới đây không thể thực hiện được do có tác động một lúc đến hai bảng cơ sở:

```
INSERT INTO nv3 (manv,hoten,noilamviec,madv,tendv)
VALUES ('NV05','Le Van E',1,3,'P. Ke toan')
```

3.4.3 Sửa đổi khung nhìn

Câu lệnh ALTER VIEW được sử dụng để định nghĩa lại khung nhìn hiện có nhưng không làm thay đổi các quyền đã được cấp phát cho người sử dụng trước đó. Câu lệnh này sử dụng tương tự như câu lệnh CREATE VIEW và có cú pháp như sau:

```
ALTER VIEW tên_khung_nhìn [(danh_sách_tên_cột)]
AS
      Câu_lệnh_SELECT
```

Ví dụ 3.17: Ta định nghĩa khung nhìn như sau:

```
CREATE VIEW viewlop
AS
      SELECT malop,tenlop,tenkhoa
      FROM lop INNER JOIN khoa ON lop.makhoa=khoa.makhoa
      WHERE tenkhoa='Khoa Vật lý'
```

và có thể định nghĩa lại khung nhìn trên bằng câu lệnh:

```
ALTER VIEW view_lop
AS
      SELECT malop,tenlop,hedaotao
      FROM lop INNER JOIN khoa ON lop.makhoa=khoa.makhoa
      WHERE tenkhoa='Khoa Công nghệ thông tin'
```

3.4.4 Xoá khung nhìn

Khi một khung nhìn không còn sử dụng, ta có thể xoá nó ra khỏi cơ sở dữ liệu thông qua câu lệnh:

```
DROP VIEW tên_khung_nhìn
```

Nếu một khung nhìn bị xoá, toàn bộ những quyền đã cấp phát cho người sử dụng trên khung nhìn cũng đồng thời bị xoá. Do đó, nếu ta tạo lại khung nhìn thì phải tiến hành cấp phát lại quyền cho người sử dụng.

Ví dụ 3.18: Câu lệnh dưới đây xoá khung nhìn VIEW_LOP ra khỏi cơ sở dữ liệu

```
DROP VIEW view_lop
```