Chương 2

NGÔN NGỮ THAO TÁC DỮ LIỆU

SQL được xem là công cụ hữu hiệu để thực hiện các yêu cầu truy vấn và thao tác trên dữ liệu. Trong chương này, ta sẽ bàn luận đến nhóm các câu lệnh trong SQL được sử dụng cho mục đích này. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language) bao gồm các câu lệnh sau:

- SELECT: Sử dụng để truy xuất dữ liệu từ môt hoặc nhiều bảng
- INSERT: Bổ sung dữ liệu
- UPDATE: Cập nhật dữ liệu
- DELETE: Xoá dữ liệu

Trong số các câu lệnh này, có thể nói SELECT là câu lệnh tương đối phức tạp và được sử dụng nhiều trong cơ sở dữ liệu. Với câu lệnh này, ta không chỉ thực hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thực hiện được các yêu cầu thống kê dữ liệu phức tạp. Cũng chính vì vậy, phần đầu của chương này sẽ tập trung tương đối nhiều đến câu lệnh SELECT. Các câu lệnh INSERT, UPDATE và DELETE được bàn luận đến ở cuối chương.

2.1 Truy xuất dữ liệu với câu lệnh SELECT

Câu lệnh SELECT được sử dụng để truy xuất dữ liệu từ các dòng và các cột của một hay nhiều bảng, khung nhìn. Câu lệnh này có thể dùng để thực hiện phép chọn (tức là truy xuất một tập con các dòng trong một hay nhiều bảng), phép chiếu (tức là truy xuất một tập con các cột trong một hay nhiều bảng) và phép nối (tức là liên kết các dòng trong hai hay nhiều bảng để truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thực hiện các thao tác truy vấn và thống kê dữ liệu phức tạp khác.

Cú pháp chung của câu lệnh SELECT có dạng:

```
SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn
[INTO   tên_bảng_mới]
FROM  danh_sách_bảng/khung_nhìn
[WHERE     điều_kiện]
[GROUP BY  danh_sách_cột]
[HAVING điều_kiện]
[ORDER BY    cột_sắp_xếp]
[COMPUTE     danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Điều cần lưu ý đầu tiên đối với câu lệnh này là các thành phần trong câu lệnh SELECT nếu được sử dụng phải tuân theo đúng thứ tự như trong cú pháp. Nếu không, câu lệnh sẽ được xem là không hợp lệ.

Câu lệnh SELECT được sử dụng để tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng được hiển thị dưới dạng bảng, tức là một tập hợp các dòng và các cột.

Ví dụ 2.1: Kết quả của câu lệnh sau đây cho biết mã lớp, tên lớp và hệ đào tạo của các lớp hiện có

```
SELECT malop,tenlop,hedaotao
FROM lop
```

malop	tenlop	hedaotao
HH1501	Hóa học K15	Chính quy
HTTT1601	Hệ thống thông tin K16	Chính quy
KTPM1601	Kỹ thuật phần mềm K16	Chính quy
NN1601	Nhật bản học K16	Chính quy
NN1702	Hàn Quốc học K17	Chính quy
TC1501	Toán cơ K15	Chính quy
TC1601	Toán cơ K16	Chính quy
VL1501	Vật lý K15	Chính quy

2.1.1 Mệnh đề FROM

Mệnh đề FROM trong câu lệnh SELECT được sử dụng nhằm chỉ định các bảng và khung nhìn cần truy xuất dữ liệu. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn, tên của các bảng và khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ 2.2: Câu lệnh dưới đây hiển thị danh sách các khoa trong trường

```
SELECT * FROM khoa
```

kết quả câu lệnh như sau:

makhoa	tenkhoa	dienthoai
DHT01	Khoa Toán cơ – Tin học	0961762912
DHT02	Khoa Công nghệ thông tin	0961764551
DHT03	Khoa Vật lý	0961760989
DHT04	Khoa Hóa học	0961766650
DHT05	Khoa Ngoại ngữ	0961715627
DHT06	Khoa Sinh học	0961719905
DHT07	Khoa Kinh tế	0961754399
DHT08	Khoa Luật	0961711554

Ta có thể sử dụng các bí danh cho các bảng hay khung nhìn trong câu lệnh SELECT. Bí danh được gán trong mệnh đề FROM bằng cách chỉ định bí danh ngay sau tên bảng.

Ví dụ 2.3: câu lệnh sau gán bí danh là *a* cho bảng *khoa*

```
SELECT * FROM khoa a
```

2.1.2 Danh sách chọn trong câu lệnh SELECT

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức được chỉ định ngay sau từ khóa SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

a. Chọn tất cả các cột trong bảng

Khi cần hiển thị tất cả các trường trong các bảng, sử dụng ký tự * trong danh sách chọn thay vì phải liệt kê danh sách tất cả các cột. Trong trường hợp này, các cột được hiển thị trong kết quả truy vấn sẽ tuân theo thứ tự mà chúng đã được tạo ra khi bảng được định nghĩa.

Ví dụ 2.4: Câu lệnh

```
SELECT * FROM lop
```

cho kết quả như sau:

malop	tenlop	khoa	hedaotao	namnhaphoc	siso	makhoa
HH1501	Hóa học K15	15	Chính quy	2021	5	DHT04
HTTT1601	Hệ thống thông tin K16	16	Chính quy	2022	6	DHT02
KTPM1601	Kỹ thuật phần mềm K16	16	Chính quy	2022	8	DHT02
NN1601	Nhật bản học K16	16	Chính quy	2022	6	DHT05
NN1702	Hàn Quốc học K17	17	Chính quy	2023	8	DHT05
TC1501	Toán cơ K15	15	Chính quy	2021	6	DHT01
TC1601	Toán cơ K16	16	Chính quy	2022	6	DHT01
VL1501	Vật lý K15	15	Chính quy	2021	5	DHT03

b. Tên cột trong danh sách chọn

Trong trường hợp cần chỉ định cụ thể các cột cần hiển thị trong kết quả truy vấn, ta chỉ định danh sách các tên cột trong danh sách chọn. Thứ tự của các cột trong kết quả truy vấn tuân theo thứ tự của các trường trong danh sách chọn.

Ví dụ 2.5: Câu lệnh

```
SELECT malop,tenlop,namnhaphoc,khoa
FROM lop
```

cho biết mã lớp, tên lớp, năm nhập học và khóa của các lớp và có kết quả như sau:

malop	tenlop	namnhaphoc	khoa
HH1501	Hóa học K15	2021	15
HTTT1601	Hệ thống thông tin K16	2022	16
KTPM1601	Kỹ thuật phần mềm K16	2022	16
NN1601	Nhật bản học K16	2022	16
NN1702	Hàn Quốc học K17	2023	17
TC1501	Toán cơ K15	2021	15
TC1601	Toán cơ K16	2022	16
VL1501	Vật lý K15	2021	15

Lưu ý: Nếu truy vấn được thực hiện trên nhiều bảng/khung nhìn và trong các bảng/khung nhìn có các trường trùng tên thì tên của những trường này nếu xuất hiện trong danh sách chọn phải được viết dưới dạng:

```
tên_bảng.tên_trường
```

Ví dụ 2.6:

```
SELECT malop, tenlop, lop.makhoa, tenkhoa
FROM lop, khoa
WHERE lop.makhoa = khoa.makhoa
```

c. Thay đổi tiêu đề các cột

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các trường tương ứng trong bảng. Tuy nhiên, để các tiêu đề trở nên thân thiện hơn, ta có thể đổi tên các tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó, ta sử dụng cách viết:

```
tiêu_đề_cột = tên_trường
```
hoặc
```
tên_trường AS tiêu_đề_cột
```
hoặc
```
tên_trường  tiêu_đề_cột
```

Ví dụ 2.7: Câu lệnh dưới đây:

```
SELECT 'Mã lớp'= malop,tenlop 'Tên lớp',khoa AS 'Khóa'
FROM lop
```

cho biết mã lớp, tên lớp và khóa học của các lớp trong trường. Kết quả của câu lệnh như sau:

Mã lớp	Tên lớp	Khóa
HH1501	Hóa học K15	15
HTTT1601	Hệ thống thông tin K16	16
KTPM1601	Kỹ thuật phần mềm K16	16
NN1601	Nhật bản học K16	16
NN1702	Hàn Quốc học K17	17
TC1501	Toán cơ K15	15
TC1601	Toán cơ K16	16
VL1501	Vật lý K15	15

d. Sử dụng cấu trúc CASE trong danh sách chọn

Cấu trúc CASE được sử dụng trong danh sách chọn nhằm thay đổi kết quả của truy vấn tuỳ thuộc vào các trường hợp khác nhau. Cấu trúc này có cú pháp như sau:

```
CASE biểu_thức
   WHEN biểu_thức_kiểm_tra THEN kết_quả
   [ ... ]
   [ELSE kết_quả_của_else]
END
```

hoặc:

```
CASE
   WHEN điều_kiện THEN kết_quả
   [ ... ]
   [ELSE kết_quả_của_else]
END
```

Ví dụ 2.8: Để hiển thị mã, họ tên và giới tính (nam hoặc nữ) của các sinh viên, ta sử dụng câu lệnh

```
SELECT masv,hodem,ten,
       CASE gioitinh
          WHEN 1 THEN 'Nam'
          ELSE N'Nữ'
       END AS gioitinh
FROM sinhvien
```

hoặc:

```
SELECT masv,hodem,ten,
       CASE
          WHEN gioitinh=1 THEN 'Nam'
          ELSE N'Nữ'
       END AS gioitinh
FROM sinhvien
```

Kết quả của hai câu lệnh trên đều có dạng như sau:

masv	hodem	ten	gioitinh
22150001	Nguyễn Tự	Ánh	Nữ
22150002	Lưu Quang	Bảo	Nam
22150003	Lê Văn	Quyền	Nam
22150004	Vương Trung	Quân	Nam
22150005	Lê Thị Ngọc	Trinh	Nữ
22150006	Nguyễn Thị Thúy	Kiều	Nữ
22151001	Nguyễn Nhật	An	Nữ
22151002	Phạm Thế	Bảo	Nam
22151003	Trương Ngọc	Hào	Nam
22151004	Phạm Nhật	Quang	Nam

e. Hằng và biểu thức trong danh sách chọn

Ngoài danh sách trường, trong danh sách chọn của câu lệnh SELECT còn có thể sử dụng các biểu thức. Mỗi một biểu thức trong danh sách chọn trở thành một cột trong kết quả truy vấn.

Ví dụ 2.9: Câu lệnh dưới đây cho biết tên và số tiết của các môn học

```
SELECT tenmonhoc,sodvht*15 AS sotiet
FROM monhoc
```

tenmonhoc	sotiet
Hóa đại cương	45
Tiếng Anh 1	60
Tiếng Anh 2	60
Tiếng Anh 3	60
Tin học đại cương	45
Ngôn ngữ C	75
Lý thuyết hệ điều hành	75
Cấu trúc dữ liệu và giải thuật	75
Cơ sở dữ liệu	75
Toán cao cấp	30
Giải tích 1	30
Vật lý đại cương	45

Nếu trong danh sách chọn có sự xuất hiện của giá trị hằng thì giá trị này sẽ xuất hiện trong một cột của kết quả truy vấn ở tất cả các dòng

Ví dụ 2.10: Câu lệnh

```
SELECT tenmonhoc,'Số tiết: ',sodvht*15 AS sotiet
FROM monhoc
```

cho kết quả như sau:

tenmonhoc	(No column name)	sotiet
Hóa đại cương	Số tiết:	45
Tiếng Anh 1	Số tiết:	60
Tiếng Anh 2	Số tiết:	60
Tiếng Anh 3	Số tiết:	60
Tin học đại cương	Số tiết:	45
Ngôn ngữ C	Số tiết:	75
Lý thuyết hệ điều hành	Số tiết:	75
Cấu trúc dữ liệu và giải thuật	Số tiết:	75
Cơ sở dữ liệu	Số tiết:	75
Toán cao cấp	Số tiết:	30
Giải tích 1	Số tiết:	30
Vật lý đại cương	Số tiết:	45

f. Loại bỏ các dòng dữ liệu trùng nhau trong kết quả truy vấn

Trong kết quả của truy vấn có thể xuất hiện các dòng dữ liệu trùng nhau. Để loại bỏ bớt các dòng này, ta chỉ định thêm từ khóa DISTINCT ngay sau từ khóa SELECT.

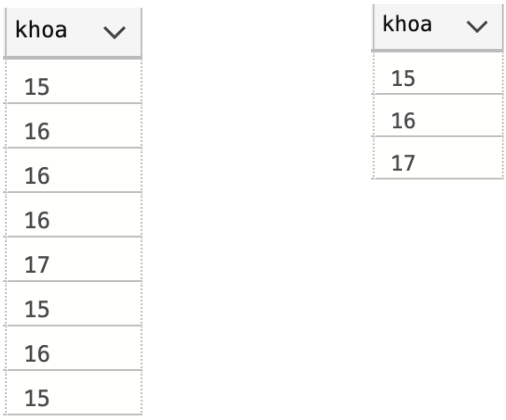
Ví dụ 2.11: Hai câu lệnh dưới đây

```
SELECT khoa FROM lop
```

và:

```
SELECT DISTINCT khoa FROM lop
```

có kết quả lần lượt như sau:



khoa
15
16
16
16
17
15
16
15

khoa
15
16
17

g. Giới hạn số lượng dòng trong kết quả truy vấn

Kết quả của truy vấn được hiển thị thường sẽ là tất cả các dòng dữ liệu truy vấn được. Trong trường hợp cần hạn chế số lượng các dòng xuất hiện trong kết quả truy vấn, ta chỉ định thêm mệnh đề TOP ngay trước danh sách chọn của câu lệnh SELECT.

Ví dụ 2.12: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của 5 sinh viên đầu tiên trong danh sách

```
SELECT TOP 5 hodem,ten,ngaysinh
FROM sinhvien
```

Ngoài cách chỉ định cụ số lượng dòng cần hiển thị trong kết quả truy vấn, ta có thể chỉ định số lượng các dòng cần hiển thị theo tỷ lệ phần trăm bằng cách sử dụng thêm từ khóa PERCENT như ở ví dụ dưới đây.

Ví dụ 2.13: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của 10% số lượng sinh viên hiện có trong bảng SINHVIEN

```
SELECT TOP 10 PERCENT hodem,ten,ngaysinh
FROM sinhvien
```

2.1.3 Chỉ định điều kiện truy vấn dữ liệu

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một biểu thức logic và chỉ những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

Ví dụ 2.14: Câu lệnh dưới đây hiển thị danh sách các môn học có số đơn vị học trình lớn hơn 3

```
SELECT * FROM  monhoc
WHERE sodvht>3
```

Kết quả của câu lệnh này như sau:

mamonhoc	tenmonhoc	sodvht
TA001	Tiếng Anh 1	4
TA002	Tiếng Anh 2	4
TA003	Tiếng Anh 3	4
TI002	Ngôn ngữ C	5
TI003	Lý thuyết hệ điều hành	5
TI004	Cấu trúc dữ liệu và giải thuật	5
TI005	Cơ sở dữ liệu	5

Trong mệnh đề WHERE thường sử dụng:

- Các toán tử kết hợp điều kiện (AND, OR)
- Các toán tử so sánh
- Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)
- Danh sách
- Kiểm tra khuôn dạng dữ liệu
- Các giá trị NULL

a. Các toán tử so sánh

Toán tử	ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ 2.15: Câu lệnh:

```
SELECT masv,hodem,ten,ngaysinh
FROM sinhvien
WHERE (ten='Anh') AND (YEAR(GETDATE())-YEAR(ngaysinh)>=20)
```

cho biết mã, họ tên và ngày sinh của các sinh viên có tên là *Anh* và có tuổi lớn hơn hoặc bằng 20.

masv	hodem	ten	ngaysinh
22160001	Ngô Thị Nhật	Anh	2002-11-27
22160002	Nguyễn Thị Ngọc	Anh	2001-03-18
22160003	Nguyễn Tuấn	Anh	2002-10-20

b. Kiểm tra giới hạn của dữ liệu

Để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử BETWEEN (NOT BETWEEN) như sau:

Cách sử dụng	Ý nghĩa
giá_trị BETWEEN a AND b	a ≤ giá_trị ≤ b
giá_trị NOT BETWEEN a AND b	(giá_trị < a) AND (giá_trị>b)

Ví dụ 2.16: Câu lệnh dưới đây cho biết họ tên và tuổi của các sinh viên có tên là *Anh* và có tuổi nằm trong khoảng từ 20 đến 22

```
SELECT hodem,ten,year(getdate())-year(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten='Anh' AND
      YEAR(GETDATE())-YEAR(ngaysinh) BETWEEN 20 AND 22
```

c. Danh sách (IN và NOT IN)

Từ khóa IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu lệnh SELECT là một danh sách các giá trị. Sau IN (hoặc NOT IN) có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ 2.17: Để biết danh sách các môn học có số đơn vị học trình là 2, 4 hoặc 5, thay vì sử dụng câu lệnh

```
SELECT * FROM monhoc
WHERE sodvht=2 OR sodvht=4 OR sodvht=5
```

ta có thể sử dụng câu lệnh

```
SELECT * FROM monhoc
WHERE sodvht IN (2,4,5)
```

d. Toán tử LIKE và các ký tự đại diện

Từ khóa LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm. Chúng thường được kết hợp với các ký tự đại diện sau đây:

Ký tự đại diện	ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Ký tự đơn bất kỳ
[]	Ký tự đơn bất kỳ trong giới hạn được chỉ định (ví dụ [a-f]) hay một tập (ví dụ [abcdef])
[^]	Ký tự đơn bất kỳ không nằm trong giới hạn được chỉ định (ví dụ [^a-f] hay một tập (ví dụ [^abcdef]).

Ví dụ 2.18: Câu lệnh dưới đây

```
SELECT hodem,ten FROM sinhvien
WHERE hodem LIKE 'Lê%'
```

cho biết họ tên của các sinh viên có họ là *Lê* và có kết quả như sau:

hodem	ten
Lê Văn	Quyền
Lê Thị Ngọc	Trinh
Lê Văn	Quyến
Lê Chí	Cường
Lê Thành	Đạt
Lê Phương	Thảo
Lê Thanh	Phong

Câu lệnh:

```
SELECT hodem,ten FROM sinhvien
WHERE hodem LIKE N'Nguyễn%' AND ten LIKE '[AB]%'
```

Có kết quả là:

hodem	ten
Nguyễn Nhật	An
Nguyễn Thị Ngọc	Anh
Nguyễn Tuấn	Anh
Nguyễn Thanh	Bình

e. Giá trị NULL

Dữ liệu trong một cột cho phép NULL sẽ nhận giá trị NULL trong các trường hợp sau:

- Nếu không có dữ liệu được nhập cho cột và không có mặc định cho cột hay kiểu dữ liệu trên cột đó.
- Người sử dụng trực tiếp đưa giá trị NULL vào cho cột đó.
- Một cột có kiểu dữ liệu là kiểu số sẽ chứa giá trị NULL nếu giá trị được chỉ định gây tràn số.

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không, ta sử dụng cách viết:

```
WHERE tên_cột IS NULL
```

hoặc:

```
WHERE tên_cột IS NOT NULL
```

2.1.4 Tạo mới bảng dữ liệu từ kết quả của câu lệnh SELECT

Câu lệnh SELECT ... INTO có tác dụng tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của truy vấn. Bảng mới được tạo ra sẽ có số cột bằng số cột được chỉ định trong danh sách chọn và số dòng sẽ là số dòng kết quả của truy vấn

Ví dụ 2.19: Câu lệnh dưới đây truy vấn dữ liệu từ bảng SINHVIEN và tạo một bảng TUOISV bao gồm các trường HODEM, TEN và TUOI

```
SELECT hodem,ten,YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
INTO tuoisv
FROM sinhvien
```

Lưu ý: Nếu trong danh sách chọn có các biểu thức thì những biểu thức này phải được đặt tiêu đề.

2.1.5 Sắp xếp kết quả truy vấn

Mặc định, các dòng dữ liệu trong kết quả của câu truy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục). Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT; Sau ORDER BY là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng.

Ví dụ 2.20: Câu lệnh dưới đây hiển thị danh sách các môn học và sắp xếp theo chiều giảm dần của số đơn vị học trình

```
SELECT * FROM monhoc
ORDER BY sodvht DESC
```

mamonhoc	tenmonhoc	sodvht
TI002	Ngôn ngữ C	5
TI003	Lý thuyết hệ điều hành	5
TI004	Cấu trúc dữ liệu và giải thuật	5
TI005	Cơ sở dữ liệu	5
TA001	Tiếng Anh 1	4
TA002	Tiếng Anh 2	4
TA003	Tiếng Anh 3	4
TI001	Tin học đại cương	3
HO001	Hóa đại cương	3
VL001	Vật lý đại cương	3
TO001	Toán cao cấp	2
TO002	Giải tích 1	2

Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo thứ tự từ trái qua phải.

Ví dụ 2.21: Câu lệnh

```
SELECT hodem,ten,gioitinh,
       YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten=N'Anh'
ORDER BY gioitinh,tuoi
```

có kết quả là:

hodem	ten	gioitinh	tuoi
Ngô Thị Nhật	Anh	0	21
Nguyễn Thị Ngọc	Anh	0	22
Nguyễn Tuấn	Anh	1	21

Thay vì chỉ định tên cột sau ORDER BY, ta có thể chỉ định số thứ tự của cột cần được sắp xếp. Câu lệnh ở ví dụ trên có thể được viết lại như sau:

```
SELECT hodem,ten,gioitinh,
       YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten=N'Anh'
ORDER BY 3, 4
```

2.1.6 Phép hợp

Phép hợp được sử dụng trong trường hợp ta cần gộp kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất. SQL cung cấp toán tử UNION để thực hiện phép hợp. Cú pháp như sau

```
Câu_lệnh_1
UNION [ALL] Câu_lệnh_2
[UNION [ALL] Câu_lệnh_3]
...
[UNION [ALL] Câu_lệnh_n]
[ORDER BY  cột_sắp_xếp]
[COMPUTE  danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Trong đó

Câu_lệnh_1 có dạng

```
SELECT  danh_sách_cột
[INTO tên_bảng_mới]
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
```

và *Câu_lệnh_i* (i = 2,..,n):

```
SELECT  danh_sách_cột
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
```

Ví dụ 2.22: Giả sử ta có hai bảng Table1 và Table2 lần lượt như sau:

A	B	C
a	1	10
b	2	20
c	3	30
d	4	40
a	5	50
b	6	60

D	E
a	1
b	2
d	3
e	4

câu lệnh

```
SELECT A,B FROM Table1
UNION
SELECT D,E FROM table2
```

Cho kết quả như sau:

A	B
a	1
a	5
b	2
b	6
c	3
d	3
d	4
e	4

Mặc định, nếu trong các truy vấn thành phần của phép hợp xuất hiện những dòng dữ liệu giống nhau thì trong kết quả truy vấn chỉ giữ lại một dòng. Nếu muốn giữ lại các dòng này, ta phải sử dụng thêm từ khóa ALL trong truy vấn thành phần.

Ví dụ 2.23: Câu lệnh

```
SELECT A,B FROM Table1
UNION ALL
SELECT D,E FROM table2
```

Cho kết quả như sau

A	B
a	1
b	2
c	3
d	4
a	5
b	6
a	1
b	2
d	3
e	4

Khi sử dụng toán tử UNION để thực hiện phép hợp, ta cần chú ý các nguyên tắc sau:

- Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.
- Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong bản thân mỗi truy vấn thành phần phải cùng kiểu dữ liệu.
- Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau. Nguyên nhân là do phép hợp so sánh các cột từng cột một theo thứ tự được cho trong mỗi truy vấn.
- Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).
- Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

- Truy vấn thành phần đầu tiên có thể có INTO để tạo mới một bảng từ kết quả của chính phép hợp.
- Mệnh đề ORDER BY dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng ở trong bất kỳ truy vấn thành phần nào.
- Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác động lên kết quả chung của phép hợp.
- Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.
- Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

2.1.7 Phép nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép nối. Một câu lệnh nối kết hợp các dòng dữ liệu trong các bảng khác nhau lại theo một hoặc nhiều điều kiện nào đó và hiển thị chúng trong kết quả truy vấn.

Xét hai bảng sau đây:

Bảng KHOA

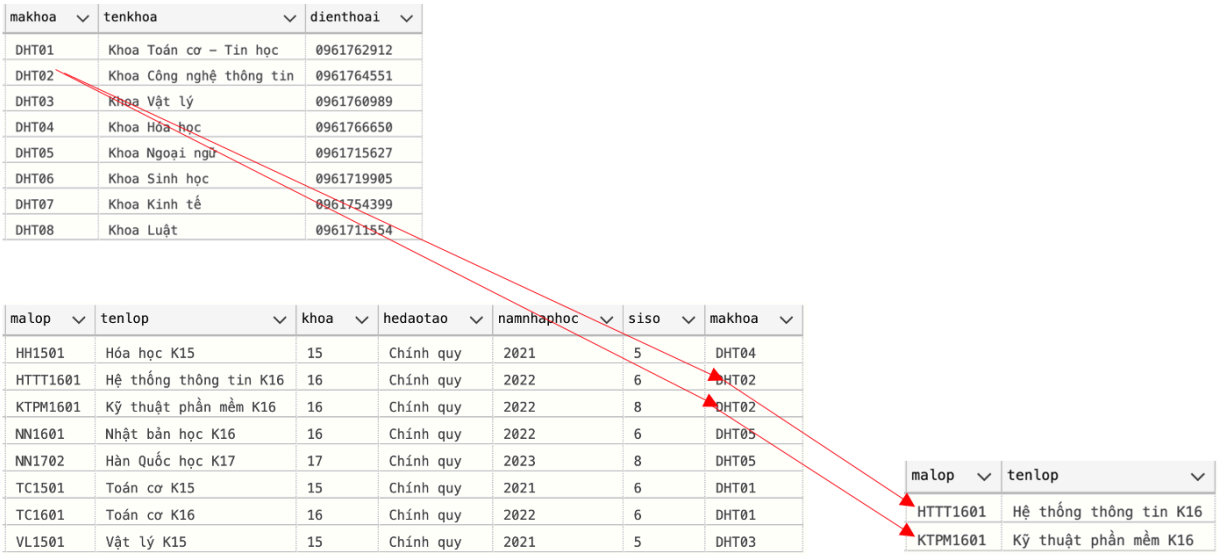
makhoa	tenkhoa	dienthoai
DHT01	Khoa Toán cơ – Tin học	0961762912
DHT02	Khoa Công nghệ thông tin	0961764551
DHT03	Khoa Vật lý	0961760989
DHT04	Khoa Hóa học	0961766650
DHT05	Khoa Ngoại ngữ	0961715627
DHT06	Khoa Sinh học	0961719905
DHT07	Khoa Kinh tế	0961754399
DHT08	Khoa Luật	0961711554

Bảng LOP

malop	tenlop	khoa	hedaotao	namnhaphoc	siso	makhoa
HH1501	Hóa học K15	15	Chính quy	2021	5	DHT04
HTTT1601	Hệ thống thông tin K16	16	Chính quy	2022	6	DHT02
KTPM1601	Kỹ thuật phần mềm K16	16	Chính quy	2022	8	DHT02
NN1601	Nhật bản học K16	16	Chính quy	2022	6	DHT05
NN1702	Hàn Quốc học K17	17	Chính quy	2023	8	DHT05
TC1501	Toán cơ K15	15	Chính quy	2021	6	DHT01
TC1601	Toán cơ K16	16	Chính quy	2022	6	DHT01
VL1501	Vật lý K15	15	Chính quy	2021	5	DHT03

Giả sử ta cần biết mã lớp và tên lớp của các lớp thuộc *Khoa Công nghệ Thông tin*, ta phải làm như sau:

- Chọn ra dòng trong bảng KHOA có tên khoa là *Khoa Công nghệ Thông tin*, từ đó xác định được mã khoa (MAKHOA) là *DHT02*.
- Tìm kiếm trong bảng LOP những dòng có giá trị trường MAKHOA là *DHT02* (tức là bằng MAKHOA tương ứng trong bảng KHOA) và đưa nhữngdòng này vào kết quả truy vấn.



makhoa	tenkhoa	dienthoai
DHT01	Khoa Toán cơ – Tin học	0961762912
DHT02	Khoa Công nghệ thông tin	0961764551
DHT03	Khoa Vật lý	0961760989
DHT04	Khoa Hóa học	0961766650
DHT05	Khoa Ngoại ngữ	0961715627
DHT06	Khoa Sinh học	0961719905
DHT07	Khoa Kinh tế	0961754399
DHT08	Khoa Luật	0961711554

malop	tenlop	khoa	hedaotao	namnhaphoc	siso	makhoa
HH1501	Hóa học K15	15	Chính quy	2021	5	DHT04
HTTT1601	Hệ thống thông tin K16	16	Chính quy	2022	6	DHT02
KTPM1601	Kỹ thuật phần mềm K16	16	Chính quy	2022	8	DHT02
NN1601	Nhật bản học K16	16	Chính quy	2022	6	DHT05
NN1702	Hàn Quốc học K17	17	Chính quy	2023	8	DHT05
TC1501	Toán cơ K15	15	Chính quy	2021	6	DHT01
TC1601	Toán cơ K16	16	Chính quy	2022	6	DHT01
VL1501	Vật lý K15	15	Chính quy	2021	5	DHT03

malop	tenlop
HTTT1601	Hệ thống thông tin K16
KTPM1601	Kỹ thuật phần mềm K16

Như vậy, để thực hiện được yêu cầu truy vấn dữ liệu trên, ta phải thực hiện phép nối giữa hai bảng KHOA và LOP với điều kiện nối là MAKHOA của KHOA bằng với MAKHOA của LOP. Câu lệnh sẽ được viết như sau:

```
SELECT malop,tenlop
FROM khoa,lop
WHERE khoa.makhoa = lop.makhoa AND
      tenkhoa='Khoa Công nghệ Thông tin'
```

2.1.7.1 Sử dụng phép nối

Phép nối là cơ sở để thực hiện các yêu cầu truy vấn dữ liệu liên quan đến nhiều bảng. Một câu lệnh nối thực hiện lấy các dòng dữ liệu trong các bảng tham gia truy vấn, so sánh giá trị của các dòng này trên một hoặc nhiều cột được chỉ định trong điều kiện nối và kết hợp các dòng thoả mãn điều kiện thành những dòng trong kết quả truy vấn.

Để thực hiện được một phép nối, cần phải xác định được những yếu tố sau:

- Những cột nào cần hiển thị trong kết quả truy vấn
- Những bảng nào có tham gia vào truy vấn
- Điều kiện để thực hiện phép nối giữa các bảng dữ liệu là gì

Trong các yếu tố kể trên, việc xác định chính xác điều kiện để thực hiện phép nối giữa các bảng đóng vai trò quan trọng nhất. Trong đa số các trường hợp, điều kiện của phép nối được xác định nhờ vào mối quan hệ giữa các bảng cần phải truy xuất dữ liệu. Thông thường, đó là điều kiện bằng nhau giữa khóa chính và khóa ngoài của hai bảng có mối quan hệ với nhau. Như vậy, để có thể đưa ra một câu lệnh nối thực hiện chính xác yêu cầu truy vấn dữ liệu đòi hỏi phải hiểu được mối quan hệ cũng như ý nghĩa của chúng giữa các bảng dữ liệu.

Danh sách chọn trong phép nối

Một câu lệnh nối cũng được bắt đầu với từ khóa SELECT. Các cột được chỉ định tên sau từ khóa SELECT là các cột được hiển thị trong kết quả truy vấn. Việc sử dụng tên các cột trong danh sách chọn có thể là:

- Tên của một số cột nào đó trong các bảng có tham gia vào truy vấn. Nếu tên cột trong các bảng trùng tên nhau thì tên cột phải được viết dưới dạng

  ```
  tên_bảng.tên_cột
  ```

- Dấu sao (*) được sử dụng trong danh sách chọn khi cần hiển thị tất cả các cột của các bảng tham gia truy vấn.
- Trong trường hợp cần hiển thị tất cả các cột của một bảng nào đó, ta sử dụng cách viết:

  ```
  tên_bảng.*
  ```

Mệnh đề FROM trong phép nối

Sau mệnh đề FROM của câu lệnh nối là danh sách tên các bảng (hay khung nhìn) tham gia vào truy vấn. Nếu ta sử dụng dấu * trong danh sách chọn thì thứ tự của các bảng liệt kê sau FROM sẽ ảnh hưởng đến thứ tự các cột được hiển thị trong kết quả truy vấn.

Mệnh đề WHERE trong phép nối

Khi hai hay nhiều bảng được nối với nhau, ta phải chỉ định điều kiện để thực hiện phép nối ngay sau mệnh đề WHERE. Điều kiện nối được biểu diễn dưới dạng biểu thức logic so sánh giá trị dữ liệu giữa các cột của các bảng tham gia truy vấn.

Các toán tử so sánh dưới đây được sử dụng để xác định điều kiện nối

Phép toán	Ý nghĩa
=	Bằng
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn

<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ 2.24: Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: mã sinh viên, họ và tên, mã lớp, tên lớp và tên khoa

```
SELECT masv,hodem,ten,sinhvien.malop,tenlop,tenkhoa
FROM sinhvien,lop,khoa
WHERE sinhvien.malop = lop.malop AND
      lop.makhoa=khoa.makhoa
```

Trong câu lệnh trên, các bảng tham gia vào truy vấn bao gồm SINHVIEN, LOP và KHOA. Điều kiện để thực hiện phép nối giữa các bảng bao gồm hai điều kiện:

```
sinhvien.malop = lop.malop
```

và

```
lop.malop = khoa.malop
```

Điều kiện nối giữa các bảng trong câu lệnh trên là điều kiện bằng giữa khóa ngoài và khóa chính của các bảng có mối quan hệ với nhau. Hay nói cách khác, điều kiện của phép nối được xác định dựa vào mối quan hệ giữa các bảng trong cơ sở dữ liệu.

2.1.7.2 Các loại phép nối

Phép nối bằng và phép nối tự nhiên

Một *phép nối bằng* (equi-join) là một phép nối trong đó giá trị của các cột được sử dụng để nối được so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia nối đều được đưa ra trong kết quả.

Ví dụ 2.25: Câu lệnh dưới đây thực hiện phép nối bằng giữa hai bảng LOP và KHOA

```
SELECT *
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

Trong kết quả của câu lệnh trên, cột *makhoa* (mã khoa) xuất hiện hai lần trong kết quả phép nối (cột *makhoa* của bảng *khoa* và cột *makhoa* của bảng *lop*) và như vậy là không cần thiết. Ta có thể loại bỏ bớt đi những cột trùng tên trong kết quả truy vấn bằng cách chỉ định danh sách cột cần được hiển thị trong danh sách chọn của câu lệnh.

Một dạng đặc biệt của phép nối bằng được sử dụng nhiều là *phép nối tự nhiên* (natural-join). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khóa ngoài và khóa chính của hai bảng; Và trong danh sách chọn của câu lệnh chỉ giữ lại một cột trong hai cột tham gia vào điều kiện của phép nối.

Ví dụ 2.26: Để thực hiện phép nối tự nhiên, ví dụ 2.25 được viết lại như sau:

```
SELECT malop,tenlop,khoa,hedaotao,namnhaphoc,
       siso,lop.makhoa,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

hoặc viết dưới dạng ngắn gọn hơn:

```
SELECT lop.*,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

Phép nối với các điều kiện bổ sung

Trong các câu lệnh nối, ngoài điều kiện của phép nối được chỉ định trong mệnh đề WHERE còn có thể chỉ định các điều kiện tìm kiếm dữ liệu khác (điều kiện chọn). Các điều kiện này được kết hợp với điều kiện nối thông qua toán tử AND.

Ví dụ 2.27: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của các sinh viên *Khoa Công nghệ Thông tin*

```
SELECT hodem,ten,ngaysinh
FROM sinhvien,lop,khoa
WHERE tenkhoa=N'Khoa Công nghệ Thông tin'
       AND sinhvien.malop = lop.malop AND
       lop.makhoa = khoa.makhoa
```

Phép tự nối và các bí danh

Phép tự nối là phép nối mà trong đó điều kiện nối được chỉ định liên quan đến các cột của cùng một bảng. Trong trường hợp này, sẽ có sự xuất hiện tên của cùng một bảng nhiều lần trong mệnh đề FROM và do đó các bảng cần phải được đặt bí danh.

Ví dụ 2.28: Để biết được họ tên và ngày sinh của các sinh viên có cùng ngày sinh với sinh viên *Ngô Thị Nhật Anh*, ta phải thực hiện phép tự nối ngay trên chính bảng *sinhvien*. Trong câu lệnh nối, bảng *sinhvien* xuất hiện trong mệnh đề FROM với bí danh là *a* và *b*. Bảng *sinhvien* với bí danh là *a* sử dụng để chọn ra sinh viên có họ tên là *Ngô Thị Nhật Anh* và bảng *sinhvien* với bí danh là *b* sử dụng để xác định các sinh viên trùng ngày sinh với sinh viên *Ngô Thị Nhật Anh*. Câu lệnh được viết như sau:

```
SELECT b.hodem,b.ten,b.ngaysinh
FROM sinhvien a, sinhvien b
WHERE a.hodem=N'Ngô Thị Nhật Anh' AND a.ten='Anh' AND
      a.ngaysinh=b.ngaysinh AND a.masv<>b.masv
```

Phép nối không dựa trên tiêu chuẩn bằng

Trong phép nối này, điều kiện để thực hiện phép nối giữa các bảng dữ liệu không phải là điều kiện so sánh bằng giữa các cột. Loại phép nối này trong thực tế thường ít được sử dụng.

Phép nối ngoài (outer-join)

Trong các phép nối đã đề cập ở trên, chỉ những dòng có giá trị trong các cột được chỉ định thoả mãn điều kiện kết nối mới được hiển thị trong kết quả truy vấn, và được gọi là phép nối trong (inner join) Theo một nghĩa nào đó, những phép nối này loại bỏ thông tin chứa trong những dòng không thoả mãn điều kiện nối. Tuy nhiên, đôi khi ta cũng cần giữ lại những thông tin này bằng cách cho phép những dòng không thoả mãn điều kiện nối có mặt trong kết quả của phép nối. Để làm điều này, ta có thể sử dụng *phép nối ngoài*.

SQL cung cấp các loại phép nối ngoài sau đây:

- **Phép nối ngoài trái** (ký hiệu: *=): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên trái trong điều kiện nối cho dù những dòng này không thoả mãn điều kiện của phép nối.
- **Phép nối ngoài phải** (ký hiệu: =*): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên phải trong điều kiện nối cho dù những dòng này không thoả điều kiện của phép nối.

Ví dụ 2.29: Giả sử ta có hai bảng DONVI và NHANVIEN như sau:

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Câu lệnh:

```
SELECT *
FROM nhanvien,donvi
WHERE nhanvien.madv=donvi.madv
```

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai

Nếu thực hiện phép nối ngoài trái giữa bảng NHANVIEN và bảng DONVI:

```
SELECT *
FROM nhanvien,donvi
WHERE nhanvien.madv*=donvi.madv
```

kết quả của câu lệnh sẽ là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Và kết quả của phép nối ngoài phải:

```
select *
from nhanvien,donvi
where nhanvien.madv=*donvi.madv
```

như sau:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

Phép nối và các giá trị NULL

Nếu trong các cột của các bảng tham gia vào điều kiện của phép nối có các giá trị NULL thì các giá trị NULL được xem như là không bằng nhau.

Ví dụ 2.30: Giả sử ta có hai bảng TABLE1 và TABLE2 như sau:

A	B
1	b1
NULL	b2
4	b3

C	D
NULL	d1
4	d2

Câu lệnh:

```
SELECT *
FROM table1, table2
WHERE A *= C
```

Có kết quả là:

A	B	C	D
1	b1	NULL	NULL
NULL	b2	NULL	NULL
4	b3	4	d2

2.1.7.4 Sử dụng phép nối trong SQL2

Ở phần trước đã đề cập đến phương pháp sử dụng phép nối trong và phép nối ngoài trong truy vấn SQL. Như đã trình bày, điều kiện của phép nối trong câu lệnh được chỉ định trong mệnh đề WHERE thông qua các biểu thức so sánh giữa các bảng tham gia truy vấn.

Chuẩn SQL2 (SQL-92) đưa ra một cách khác để biểu diễn cho phép nối, trong cách biểu diễn này, điều kiện của phép nối không được chỉ định trong mệnh đề WHERE mà được chỉ định ngay trong mệnh đề FROM của câu lệnh. Cách sử dụng phép nối này cho phép ta biểu diễn phép nối cũng như điều kiện nối được rõ ràng, đặc biệt là trong trường hợp phép nối được thực hiện trên ba bảng trở lên.

Phép nối trong

Điều kiện để thực hiện phép nối trong được chỉ định trong mệnh đề FROM theo cú pháp như sau:

```
tên_bảng_1 [INNER] JOIN tên_bảng_2 ON điều_kiện_nối
```

Ví dụ 2.31: Để hiển thị họ tên và ngày sinh của các sinh viên lớp *Toán cơ K15*, thay vì sử dụng câu lệnh:

```
SELECT hodem,ten,ngaysinh
FROM sinhvien,lop
WHERE tenlop=N'Toán cơ K15' AND
      sinhvien.malop=lop.malop
```

ta có thể sử dụng câu lệnh như sau:

```
SELECT hodem,ten,ngaysinh
FROM sinhvien INNER JOIN lop
     ON sinhvien.malop=lop.malop
WHERE tenlop= N'Toán cơ K15'
```

Phép nối ngoài

SQL2 cung cấp các phép nối ngoài sau đây:

- Phép nối ngoài trái (LEFT OUTER JOIN)
- Phép nối ngoài phải (RIGHT OUTER JOIN)
- Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Cũng tương tự như phép nối trong, điều kiện của phép nối ngoài cũng được chỉ định ngay trong mệnh đề FROM theo cú pháp:

```
tên_bảng_1 LEFT|RIGHT|FULL [OUTER] JOIN tên_bảng_2
            ON điều_kiện_nối
```

Ví dụ 2.32: Giả sử ta có hai bảng dữ liệu như sau:

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Phép nối ngoài trái giữa hai bảng NHANVIEN và DONVI được biểu diễn bởi câu lệnh:

```
SELECT *
FROM nhanvien LEFT OUTER JOIN donvi
     ON nhanvien.madv=donvi.madv
```

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Câu lệnh:

```
SELECT *
FROM nhanvien RIGHT OUTER JOIN donvi
     ON nhanvien.madv=donvi.madv
```

thực hiện phép nối ngoài phải giữa hai bảng NHANVIEN và DONVI, và có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

Nếu phép nối ngoài trái (tương ứng phải) hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của bảng bên trái (tương ứng phải) trong phép nối thì phép nối ngoài đầy đủ hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của cả hai bảng tham gia vào phép nối.

Ví dụ 2.33: Với hai bảng NHANVIEN và DONVI như ở trên, câu lệnh

```
SELECT *
FROM nhanvien FULL OUTER JOIN donvi
     ON nhanvien.madv=donvi.madv
```

cho kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL
NULL	NULL	4	Kinh doanh
NULL	NULL	3	Ke toan

Thực hiện phép nối trên nhiều bảng

Một đặc điểm nổi bật của SQL2 là cho phép biểu diễn phép nối trên nhiều bảng dữ liệu một cách rõ ràng. Thứ tự thực hiện phép nối giữa các bảng được xác định theo nghĩa kết quả của phép nối này được sử dụng trong một phép nối khác.

Ví dụ 2.34: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của các sinh viên thuộc *Khoa Công nghệ Thông tin*

```
SELECT hodem,ten,ngaysinh
FROM (sinhvien INNER JOIN lop
                ON sinhvien.malop=lop.malop)
      INNER JOIN khoa ON lop.makhoa=khoa.makhoa
WHERE tenkhoa=N'Khoa công nghệ thông tin'
```

Trong câu lệnh trên, thứ tự thực hiện phép nối giữa các bảng được chỉ định rõ ràng: phép nối giữa hai bảng *sinhvien* và *lop* được thực hiện trước và kết quả của phép nối này lại tiếp tục được nối với bảng *khoa*.

2.1.8 Thống kê dữ liệu với GROUP BY

Ngoài khả năng thực hiện các yêu cầu truy vấn dữ liệu thông thường (chiếu, chọn, nối,…) như đã đề cập như ở các phần trước, câu lệnh SELECT còn cho phép thực hiện các thao tác truy vấn và tính toán thống kê trên dữ liệu như: c*ho biết tổng số tiết dạy của mỗi giáo viên, điểm trung bình các môn học của mỗi sinh viên*,…

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu, và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình,...

Các hàm gộp được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm gộp dưới đây:

Hàm gộp	**Chức năng**
SUM([ALL | DISTINCT] *biểu_thức*)	Tính tổng các giá trị.
AVG([ALL | DISTINCT] *biểu_thức*)	Tính trung bình của các giá trị
COUNT([ALL | DISTINCT] *biểu_thức*)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn.
MAX(*biểu_thức*)	Tính giá trị lớn nhất
MIN(*biểu_thức*)	Tính giá trị nhỏ nhất

Trong đó:

- Hàm SUM và AVG chỉ làm việc với các biểu thức số.
- Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.
- Hàm COUNT(*) không bỏ qua các giá trị NULL.

Mặc định, các hàm gộp thực hiện tính toán thống kê trên toàn bộ dữ liệu. Trong trường hợp cần loại bỏ bớt các giá trị trùng nhau (chỉ giữ lại một giá trị), ta chỉ định thêm từ khóa DISTINCT ở trước biểu thức là đối số của hàm.

Thống kê trên toàn bộ dữ liệu

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm gộp trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bất kỳ một tên cột hay biểu thức nào ngoài các hàm gộp.

Ví dụ 2.35: Để thống kê trung bình điểm lần 1 của tất cả các môn học, ta sử dụng câu lệnh như sau:

```
SELECT AVG(diemlan1)
FROM diemthi
```

còn câu lệnh dưới đây cho biết tuổi lớn nhất, tuổi nhỏ nhất và độ tuổi trung bình của tất cả các sinh viên sinh tại *HCM*:

```
SELECT MAX(YEAR(GETDATE())-YEAR(ngaysinh)),
       MIN(YEAR(GETDATE())-YEAR(ngaysinh)),
       AVG(YEAR(GETDATE())-YEAR(ngaysinh))
FROM sinhvien
WHERE noisinh='HCM'
```

Thống kê dữ liệu trên các nhóm

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu, ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu vào trong các nhóm. Các hàm gộp được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ 2.36: Câu lệnh dưới đây cho biết sĩ số (số lượng sinh viên) của mỗi lớp

```
SELECT lop.malop,tenlop,COUNT(masv) AS siso
FROM lop,sinhvien
WHERE lop.malop=sinhvien.malop
GROUP BY lop.malop,tenlop
```

và có kết quả là

malop	tenlop	siso
HH1501	Hóa học K15	5
HTTT1601	Hệ thống thông tin K16	6
KTPM1601	Kỹ thuật phần mềm K16	8
NN1601	Nhật bản học K16	6
NN1702	Hàn Quốc học K17	8
TC1501	Toán cơ K15	6
TC1601	Toán cơ K16	6
VL1501	Vật lý K15	5

còn câu lệnh:

```
SELECT sinhvien.masv,hodem,ten,
       sum(diemlan1*sodvht)/sum(sodvht)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND
      diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv,hodem,ten
```

cho biết trung bình điểm thi lần 1 các môn học của các sinh viên.

Lưu ý: Trong trường hợp danh sách chọn của câu lệnh SELECT có cả các hàm gộp và những biểu thức không phải là hàm gộp thì những biểu thức này phải có mặt đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Ví dụ 2.37: Dưới đây là một câu lệnh sai

```
SELECT lop.malop,tenlop,COUNT(masv)
FROM lop,sinhvien
WHERE lop.malop=sinhvien.malop
GROUP BY lop.malop
```

do thiếu trường TENLOP sau mệnh đề GROUP BY.

Chỉ định điều kiện đối với hàm gộp

Mệnh đề HAVING được sử dụng nhằm chỉ định điều kiện đối với các giá trị thống kê được sản sinh từ các hàm gộp tương tự như cách thức mệnh đề WHERE thiết lập các điều kiện cho câu lệnh SELECT. Mệnh đề HAVING thường không thực sự có nghĩa nếu như không sử dụng kết hợp với mệnh đề GROUP BY. Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được có các hàm gộp trong khi HAVING lại cho phép sử dụng các hàm gộp trong điều kiện của mình.

Ví dụ 2.38: Để biết trung bình điểm thi lần 1 của các sinh viên có điểm trung bình lớn hơn hoặc bằng 5, ta sử dụng câu lệnh như sau:

```
SELECT sinhvien.masv,hodem,ten,
       SUM(diemlan1*sodvht)/sum(sodvht)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND
      diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv,hodem,ten
HAVING sum(diemlan1*sodvht)/sum(sodvht)>=5
```

2.1.9 Truy vấn con (Subquery)

Truy vấn con là một câu lệnh SELECT được lồng vào bên trong một câu lệnh SELECT, INSERT, UPDATE, DELETE hoặc bên trong một truy vấn con khác. Loại truy vấn này được sử dụng để biểu diễn cho những truy vấn trong đó điều kiện truy vấn dữ liệu cần phải sử dụng đến kết quả của một truy vấn khác.

Cú pháp của truy vấn con như sau:

```
(SELECT [ALL | DISTINCT] danh_sách_chọn
FROM  danh_sách_bảng
[WHERE     điều_kiện]
[GROUP BY  danh_sách_cột]
[HAVING điều_kiện])
```

Khi sử dụng truy vấn con cần lưu ý một số quy tắc sau:

- Một truy vấn con phải được viết trong cặp dấu ngoặc. Trong hầu hết các trường hợp, một truy vấn con thường phải có kết quả là một cột (tức là chỉ có duy nhất một cột trong danh sách chọn).
- Mệnh đề ORDER BY không được phép sử dụng trong truy vấn con.
- Các tên cột xuất hiện trong truy vấn con có thể là các cột của các bảng trong truy vấn ngoài.
- Một truy vấn con thường được sử dụng làm điều kiện trong mệnh đề WHERE hoặc HAVING của một truy vấn khác.
- Nếu truy vấn con trả về đúng một giá trị, nó có thể sử dụng như là một thành phần bên trong một biểu thức (ví dụ phép so sánh bằng).

Phép so sánh đối với với kết quả truy vấn con

Kết quả của truy vấn con có thể được sử dụng đề thực hiện phép so sánh số học với một biểu thức của truy vấn cha. Trong trường hợp này, truy vấn con có dạng:

```
WHERE biểu_thức phép_toán_số_học [ANY|ALL]
      (truy_vấn_con)
```

Trong đó phép toán số học có thể sử dụng bao gồm: =, <>, >, <, >=, <=; Và truy vấn con phải có kết quả bao gồm đúng một cột.

Ví dụ 2.43: Câu lệnh dưới đây cho biết danh sách các môn học có số đơn vị học trình lớn hơn hoặc bằng số đơn vị học trình của môn học có mã là *TI001*

```
SELECT *
FROM monhoc
WHERE sodvht>=(SELECT sodvht FROM monhoc
                  WHERE mamonhoc='TI001')
```

Nếu truy vấn con trả về nhiều hơn một giá trị, việc sử dụng phép so sánh như trên sẽ không hợp lệ. Trong trường hợp này, sau phép toán so sánh phải sử dụng thêm lượng từ ALL hoặc ANY. Lượng từ ALL được sử dụng khi cần so sánh giá trị của biểu thức với tất cả các giá trị trả về trong kết quả của truy vấn con; ngược lai, phép so sánh với lượng từ ANY có kết quả đúng khi chỉ cần một giá trị bất kỳ nào đó trong kết quả của truy vấn con thoả mãn điều kiện.

Ví dụ 2.44: Câu lệnh dưới đây cho biết họ tên của những sinh viên lớp *Toán cơ K15* sinh trước tất cả các sinh viên của lớp *Hóa học K15*

```
SELECT hodem,ten
FROM sinhvien JOIN lop ON sinhvien.malop=lop.malop
WHERE tenlop=N'Toán cơ K15' AND
ngaysinh<ALL(SELECT ngaysinh
             FROM sinhvien JOIN lop
             ON sinhvien.malop=lop.malop WHERE
             lop.tenlop=N'Hoá học K15')
```

và câu lệnh:

```
SELECT hodem,ten
FROM sinhvien JOIN lop on sinhvien.malop=lop.malop
WHERE tenlop=N'Toán cơ K15' AND
   year(ngaysinh)= ANY(SELECT year(ngaysinh)
                       FROM sinhvien JOIN lop
                       ON sinhvien.malop=lop.malop
                       WHERE lop.tenlop=N'Hoá học K15')
```

cho biết họ tên của những sinh viên lớp *Toán cơ K15* có năm sinh trùng với năm sinh của bất kỳ một sinh viên nào đó của lớp *Hóa học K15.*

Sử dụng truy vấn con với toán tử IN

Khi cần thực hiện phép kiểm tra giá trị của một biểu thức có xuất hiện (không xuất hiện) trong tập các giá trị của truy vấn con hay không, ta có thể sử dụng toán tử IN(NOT IN) như sau:

```
WHERE biểu_thức [NOT] IN (truy_vấn_con)
```

Ví dụ 2.45: Để hiển thị họ tên của những sinh viên lớp *Toán cơ K15* có năm sinh bằng với năm sinh của một sinh viên nào đó của lớp *Hóa học K15*, thay vì sử dụng câu lệnh như ở ví dụ trên, ta có thể sử dụng câu lệnh như sau:

```
SELECT hodem,ten
FROM sinhvien JOIN lop on sinhvien.malop=lop.malop
WHERE tenlop=N'Toán cơ K15' AND
      year(ngaysinh)IN(SELECT year(ngaysinh)
                  FROM sinhvien JOIN lop
                  ON sinhvien.malop=lop.malop WHERE
                  lop.tenlop='Hoá học K15')
```

Sử dụng lượng từ EXISTS với truy vấn con

Lượng từ EXISTS được sử dụng kết hợp với truy vấn con dưới dạng:

```
WHERE [NOT] EXISTS (truy_vấn_con)
```

để kiểm tra xem một truy vấn con có trả về dòng kết quả nào hay không. Lượng từ EXISTS (tương ứng NOT EXISTS) trả về giá trị True (tương ứng False) nếu kết quả của truy vấn con có ít nhất một dòng (tương ứng không có dòng nào). Điều khác biệt của việc sử dụng EXISTS với hai cách đã nêu ở trên là trong danh sách chọn của truy vấn con có thể có nhiều hơn hai cột.

Ví dụ 2.46: Câu lệnh dưới đây cho biết họ tên của những sinh viên hiện chưa có điểm thi của bất kỳ một môn học nào

```
SELECT hodem,ten
FROM sinhvien
WHERE NOT EXISTS(SELECT masv FROM diemthi
                 WHERE diemthi.masv=sinhvien.masv)
```

Sử dụng truy vấn con với mệnh đề HAVING

Một truy vấn con có thể được sử dụng trong mệnh đề HAVING của một truy vấn khác. Trong trường hợp này, kết quả của truy vấn con được sử dụng để tạo nên điều kiện đối với các hàm gộp.

Ví dụ 2.47: Câu lệnh dưới đây cho biết mã, tên và trung bình điểm lần 1 của các môn học có trung bình lớn hơn trung bình điểm lần 1 của tất cả các môn học

```
SELECT diemthi.mamonhoc,tenmonhoc,AVG(diemlan1)
FROM diemthi,monhoc
WHERE diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY diemthi.mamonhoc,tenmonhoc
HAVING AVG(diemlan1)>
       (SELECT AVG(diemlan1) FROM diemthi)
```

2.2 Bổ sung, cập nhật và xoá dữ liệu

Các câu lệnh thao tác dữ liệu trong SQL không những chỉ sử dụng để truy vấn dữ liệu mà còn để thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu. So với câu lệnh SELECT, việc sử dụng các câu lệnh để bổ sung, cập nhật hay xoá dữ liệu đơn giản hơn nhiều. Trong phần còn lại của chương này sẽ đề cập đến 3 câu lệnh:

- Lệnh INSERT
- Lệnh UPDATE
- Lệnh DELETE

2.2.1 Bổ sung dữ liệu

Dữ liệu trong các bảng được thể hiện dưới dạng các dòng (bản ghi). Để bổ sung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT. Hầu hết các hệ quản trị CSDL dựa trên SQL cung cấp các cách dưới đây để thực hiện thao tác bổ sung dữ liệu cho bảng:

- Bổ sung từng dòng dữ liệu với mỗi câu lệnh INSERT. Đây là các sử dụng thường gặp nhất trong giao tác SQL.
- Bổ sung nhiều dòng dữ liệu bằng cách truy xuất dữ liệu từ các bảng dữ liệu khác.

Bổ sung từng dòng dữ liệu với lệnh INSERT

Để bổ sung một dòng dữ liệu mới vào bảng, ta sử dụng câu lệnh INSERT với cú pháp như sau:

```
INSERT  INTO tên_bảng[(danh_sách_cột)]
VALUES(danh_sách_trị)
```

Trong câu lệnh INSERT, danh sách cột ngay sau tên bảng không cần thiết phải chỉ định nếu giá trị các trường của bản ghi mới được chỉ định đầy đủ trong danh sách trị. Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng với số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo đúng thứ tự của các trường như khi bảng được định nghĩa.

Ví dụ 2.48: Câu lệnh dưới đây bổ sung thêm một dòng dữ liệu vào bảng KHOA

```
INSERT INTO khoa
VALUES('DHT08','Khoa Luật',' 0961711554')
```

Trong trường hợp chỉ nhập giá trị cho một số cột trong bảng, ta phải chỉ định danh sách các cột cần nhập dữ liệu ngay sau tên bảng. Khi đó, các cột không được nhập dữ liệu sẽ nhận giá trị mặc định (nếu có) hoặc nhận giá trị NULL (nếu cột cho phép chấp nhận giá trị NULL). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị

NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

Ví dụ 2.49: Câu lệnh dưới đây bổ sung một bản ghi mới cho bảng SINHVIEN

```
INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)
VALUES('22170008',N'Lê Thanh,N'Phong',1,'NN1702')
```

câu lệnh trên còn có thể được viết như sau:

```
INSERT INTO sinhvien VALUES
VALUES('22170008',N'Lê Thanh,N'Phong',null,1,null,'NN1702')
```

Bổ sung nhiều dòng dữ liệu từ bảng khác

Câu lệnh INSERT còn được sử dụng để bổ sung nhiều dòng dữ liệu vào một bảng, các dòng dữ liệu này được lấy từ một bảng khác thông qua câu lệnh SELECT. Ở cách này, các giá trị dữ liệu được bổ sung vào bảng không được chỉ định tường minh mà thay vào đó là một câu lệnh SELECT truy vấn dữ liệu từ bảng khác.

Cú pháp câu lệnh INSERT có dạng như sau:

```
INSERT  INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT
```

Ví dụ 2.50: Giả sử ta có bảng LUUSINHVIEN bao gồm các trường HODEM, TEN, NGAYSINH. Câu lệnh dưới đây bổ sung vào bảng LUUSINHVIEN các dòng dữ liệu có được từ câu truy vấn SELECT:

```
INSERT INTO luusinhvien
SELECT  hodem,ten,ngaysinh
FROM sinhvien
WHERE noisinh like '%HCM%'
```

Khi bổ sung dữ liệu theo cách này cần lưu ý rằng kết quả của câu lệnh SELECT phải có số cột bằng với số cột được chỉ định trong bảng đích và phải tương thích về kiểu dữ liệu.

2.2.2 Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng.

Câu lệnh này có cú pháp như sau:

```
UPDATE tên_bảng
SET  tên_cột = biểu_thức
     [, ..., tên_cột_k = biểu_thức_k]
[FROM  danh_sách_bảng]
[WHERE điều_kiện]
```

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định các danh sách tên cột và biểu thức tương ứng sau từ khóa SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng để chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nếu không chỉ định, phạm vi tác động của câu lệnh được hiểu là toàn bộ các dòng trong bảng).

Ví dụ 2.51: Câu lệnh dưới đây cập nhật lại số đơn vị học trình của các môn học có số đơn vị học trình nhỏ hơn 2

```
UPDATE monhoc
SET sodvht = 3
WHERE sodvht = 2
```

Sử dụng cấu trúc CASE trong câu lệnh UPDATE

Cấu trúc CASE có thể được sử dụng trong biểu thức khi cần phải đưa ra các quyết định khác nhau về giá trị của biểu thức.

Ví dụ 2.52: Giả sử ta có bảng NHATKYPHONG sau đây

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	
202	B	5	
101	A	2	
102	C	3	

Sau khi thực hiện câu lệnh:

```
UPDATE nhatkyphong
SET tienphong=songay*CASE WHEN loaiphong='A' THEN 100
                          WHEN loaiphong='B' THEN 70
                          ELSE 50
                     END
```

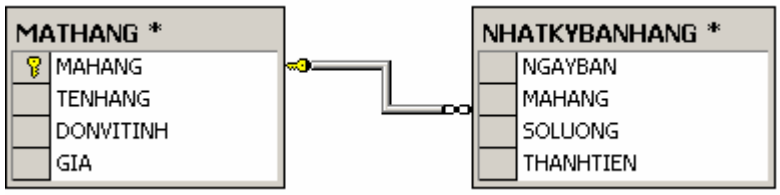
Dữ liệu trong bảng sẽ là:

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	500
202	B	5	350
101	A	2	200
102	C	3	150

Điều kiện cập nhật dữ liệu liên quan đến nhiều bảng

Mệnh đề FROM trong câu lệnh UPDATE được sử dụng khi cần chỉ định các điều kiện liên quan đến các bảng khác với bảng cần cập nhật dữ liệu. Trong trường hợp này, trong mệnh đề WHERE thường có điều kiện nối giữa các bảng.

Ví dụ 2.53: Giả sử ta có hai bảng MATHANG và NHATKYBANHANG như sau:



Câu lệnh dưới đây sẽ cập nhật giá trị trường THANHTIEN của bảng NHATKYBANHANG theo công thức THANHTIEN = SOLUONG × GIA

```
UPDATE nhatkybanhang
SET  thanhtien = soluong*gia
FROM mathang
WHERE nhatkybanhang.mahang = mathang.mahang
```

Câu lệnh UPDATE với truy vấn con

Tương tự như trong câu lệnh SELECT, truy vấn con có thể được sử dụng trong mệnh đề WHERE của câu lệnh UPDATE nhằm chỉ định điều kiện đối với các dòng dữ liệu cần cập nhật dữ liệu.

Ví dụ 2.54: Câu lệnh ở trên có thể được viết như sau:

```
UPDATE nhatkybanhang
SET  thanhtien = soluong*gia
FROM mathang
WHERE mathang.mahang =(SELECT mathang.mahang
      FROM mathang
      WHERE mathang.mahang=nhatkybanhang.mahang)
```

2.2.3 Xoá dữ liệu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE. Cú pháp:

```
DELETE FROM tên_bảng
[FROM danh_sách_bảng]
[WHERE điều_kiện]
```

Trong câu lệnh này, tên của bảng cần xoá dữ liệu được chỉ định sau DELETE FROM. Mệnh đề WHERE trong câu lệnh được sử dụng để chỉ định điều kiện đối với các dòng dữ liệu cần xoá. Nếu câu lệnh DELETE không có mệnh đề WHERE thì toàn bộ các dòng dữ liệu trong bảng đều bị xoá.

Ví dụ 2.55: Câu lệnh dưới đây xoá khỏi bảng SINHVIEN những sinh viên sinh tại HCM

```
DELETE FROM sinhvien
WHERE noisinh LIKE '%HCM%'
```

Xoá dữ liệu khi điều kiện liên quan đến nhiều bảng

Nếu điều kiện trong câu lệnh DELETE liên quan đến các bảng không phải là bảng cần xóa dữ liệu, ta phải sử dụng thêm mệnh đề FROM và sau đó là danh sách tên các bảng đó. Trong trường hợp này, trong mệnh đề WHERE ta chỉ định thêm điều kiện nối giữa các bảng.

Ví dụ 2.56: Xóa khỏi bảng SINHVIEN những sinh viên lớp *Toán cơ K15*

```
DELETE FROM
sinhvienFROM lop
WHERE lop.malop=sinhvien.malop AND tenlop=N'Toán cơ K15'
```

Sử dụng truy vấn con trong câu lệnh DELETE

Một câu lệnh SELECT có thể được lồng vào trong mệnh đề WHERE trong câulệnh DELETE để làm điều kiện cho câu lệnh tương tự như câu lệnh UPDATE.

Ví dụ 2.57: Xóa những lớp không có sinh viên nào học

```
DELETE FROM lop
WHERE malop NOT IN (SELECT DISTINCT malop
                    FROM sinhvien)
```

Xoá toàn bộ dữ liệu trong bảng

Câu lệnh DELETE không chỉ định điều kiện đối với các dòng dữ liệu cần xoá trong mệnh đề WHERE sẽ xoá toàn bộ dữ liệu trong bảng. Thay vì sử dụng câu lệnh DELETE trong trường hợp này, ta có thể dùng câu lệnh TRUNCATE như sau:

```
TRUNCATE TABLE tên_bảng
```

Ví dụ 2.58: Câu lệnh sau xoá toàn bộ dữ liệu trong bảng *diemthi*:

```
DELETE FROM diemthi
```

có tác dụng tương tự với câu lệnh

```
TRUNCATE TABLE diemthi
```