



Bài 3. Truy vấn dữ liệu trên Hệ quản trị CSDL SQL Server

- Mục đích, yêu cầu: Cung cấp cho sinh viên kiến thức tổng quan về Truy vấn dữ liệu trên Hệ quản trị CSDL SQL Server.
- Hình thức tổ chức dạy học: Lý thuyết, trực tiếp + trực tuyến + tự học
- Thời gian: Lý thuyết(trên lớp: 3; online: 2) Tự học, tự nghiên cứu: 10
- Nội dung chính:

Truy vấn dữ liệu trên Hệ quản trị CSDL SQL Server

I. Ngôn ngữ thao tác dữ liệu

Đối với đa số người sử dụng, SQL được xem như 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 một 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.

II. 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.

Kết quả của câu lệnh sau đây cho biết mã lớp, tên lớp và hệ đào tạo của các lớp:

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

MALOP	TENLOP	HEDAOTAO
C24101	Toán K24	Chính quy
C24102	Tin K24	Chính quy
C24103	Lý K24	Chính quy
C24301	Sinh K24	Chính quy

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.

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	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hoá học	054823951

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. Câu lệnh sau gán bí danh là cho bảng *khoa*

```
SELECT * FROM khoa a
```

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 khoá SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

Chọn tất cả các cột trong bảng

```
SELECT * FROM lop
```

cho kết quả bao như sau:

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05

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. Câu lệnh

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

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



MALOP	TENLOP	NAMNHAPHOC	KHOA
C24101	Toán K24	2000	24
C24102	Tin K24	2000	24
C24103	Lý K24	2000	24
C24301	Sinh K24	2000	24
C25101	Toán K25	2001	25

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`

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

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_trườngtiêu_đề_cột`

Câu lệnh dưới đây:

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

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



Mã lớp	Tên Lớp	Khoá
C24101	Toán K24	24
C24102	Tin K24	24
C24103	Lý K24	24
C24301	Sinh K24	24

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 tùy 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
```

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 'Nữ'
                        END AS gioitinh
FROM sinhvien
```

hoặc

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

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



MASV	HODEM	TEN	GIOITINH
0241010001	Ngô Thị Nhật	Anh	Nữ
0241010002	Nguyễn Thị Ngọc	Anh	Nữ
0241010003	Ngô Việt	Bắc	Nam
0241010004	Nguyễn Đình	Bình	Nam

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.

câu lệnh dưới đây cho biết tên và số tiết của các môn học

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

TENMONHOC	SOTIET
Hoá đại cương	45
Tin học đại cương	60
Ngôn ngữ C	75
Lý thuyết hệ điều hành	60

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. Câu lệnh

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

cho kết quả như sau:

TENMONHOC	(No column name)	SOTIET
Hoá đại cương	Số tiết:	45
Tin học đại cương	Số tiết:	60
Ngôn ngữ C	Số tiết:	75
Lý thuyết hệ điều hành	Số tiết:	60

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 khoá SELECT. Hai câu lệnh dưới đây

```
SELECT khoa FROM lop
```

Và



`SELECT DISTINCT khoa FROM lop`

có kết quả lần lượt như sau:

KHOA	
24	
24	
24	
24	
25	
25	
25	
25	KHOA
25	24
26	25
26	26

Giới hạn số lượng dòng trong kết quả truy vấn

ta chỉ định thêm mệnh đề TOP ngay trước danh sách chọn của câu lệnh SELECT.

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 thể 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 khoá PERCENT như ở ví dụ dưới đây.

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`

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.

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
TI-001	Tin học đại cương	4
TI-002	Ngôn ngữ C	5
TI-003	Lý thuyết hệ điều hành	4

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

Các toán tử so sánh

T toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Câu lệnh:

```
SELECT masv,hodem,ten,,ngaysinh FROM sinhvien WHERE
(tenn='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 nhỏ hơn hoặc bằng 20.

MASV	HODEM	TEN	NGAYSINH
0261010001	Lê Hoàng Phương	Anh	1984-03-04 00:00:00
0261010002	Lê Thị Vân	Anh	1984-10-14 00:00:00
0261020002	Lê Thúc Quốc	Anh	1984-12-04 00:00:00



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:

giá_trị BETWEEN a AND b $\Leftrightarrow a \leq \text{giá_trị} \leq b$

giá_trị NOT BETWEEN a AND b $\Leftrightarrow (\text{giá_trị} < a) \text{ AND } (\text{giá_trị} > b)$

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

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

Danh sách (IN và NOT IN)

Từ khoá 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.

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)
```

Toán tử LIKE và các ký tự đại diện

Từ khoá LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. 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])).

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 Thị Thanh	Châu
Lê Thị	Anh
Lê Văn Khoa	Bảo

Câu lệnh:

```
SELECT hodem,ten
```

```
FROM sinhvien
```

```
WHERE hodem LIKE 'Lê%' AND ten LIKE '[AB]%'
```

Có kết quả là:

HODEM	TEN
Lê Thị	Anh
Lê Văn Khoa	Bảo
Lê Hoàng Phương	Anh

Giá trị NULL

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
```



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

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.

Sắp xếp kết quả truy vấn

Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng. 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
TI-002	Ngôn ngữ C	5
TI-003	Lý thuyết hệ điều hành	4
TI-001	Tin học đại cương	4
TI-004	Cấu trúc dữ liệu và giải thuật	4
TO-001	Đại số tuyến tính	4
TO-002	Giải tích 1	4
HO-001	Hoá đại cương	3

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

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

có kết quả là:



HODEM	TEN	GIOITINH	TUOI
Nguyễn Thị	Bình	0	23
Hoàng Văn	Bình	1	21
Châu Văn Quốc	Bình	1	21
Nguyễn Thanh	Bình	1	22

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 = 'Bình'
ORDER BY 3, 4
```

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 ($i = 2, \dots, n$) có dạng



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

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

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

câu lệnh

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

Cho kết quả như sau:

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

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 khoá ALL trong truy vấn thành phần.

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

III. 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:

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hoá học	054823951
DHT05	Khoa Sinh học	054822934
DHT06	Khoa Địa lý - Địa chất	054823837

Bảng KHOA

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05
C25101	Toán K25	25	Chính quy	2001	DHT01
C25102	Tin K25	25	Chính quy	2001	DHT02
C25103	Lý K25	25	Chính quy	2001	DHT03
C25301	Sinh K25	25	Chính quy	2001	DHT05
C26101	Toán K26	26	Chính quy	2002	DHT01
C26102	Tin K26	26	Chính quy	2002	DHT02

Bảng LOP

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 những dòng này vào kết quả truy vấn

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hoá học	054823951
DHT05	Khoa Sinh học	054822934
DHT06	Khoa Địa lý - Địa chất	054823837

MALOP	TENLOP	KHOA	MEADAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05
C25101	Toán K25	25	Chính quy	2001	DHT01
C25102	Tin K25	25	Chính quy	2001	DHT02
C25103	Lý K25	25	Chính quy	2001	DHT03
C25301	Sinh K25	25	Chính quy	2001	DHT05
C26101	Toán K26	26	Chính quy	2002	DHT01
C26102	Tin K26	26	Chính quy	2002	DHT02

MALOP	TENLOP
C24102	Tin K24
C25102	Tin K25
C26102	Tin K26

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'
```

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



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 khoá 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

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn

<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	khác
!>	Không lớn hơn
!<	Không nhỏ hơn



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 khoá ngoài và khoá 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.

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.

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* (naturaljoin). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khoá ngoài và khoá 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. Để thực hiện phép nối tự nhiên, câu lệnh trong 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) . Thông thường, các điều kiện này được kết hợp với điều kiện nối thông qua toán tử AND. 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='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.

Để biết được họ tên và ngày sinh của các sinh viên có cùng ngày sinh với sinh viên *Trần Thị Kim 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 *Trần Thị Kim Anh* và bảng *sinhvien* với bí danh là *b* sử dụng để xác định các sinh viên trùng ngày sinh với sinh viên *Trần Thị Kim Anh*.

Câu lệnh được viết như sau:

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

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
```

Để hiển thị họ tên và ngày sinh của các sinh viên lớp *TinK24*, thay vì sử dụng câu lệnh:

```
SELECT hodem,ten,ngaysinh
```

```
FROM sinhvien,lop
```

```
WHERE tenlop='Tin K24' 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='Tin K24'
```

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.

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*.

IV. 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: *cho 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] <i>biểu_thức</i>)	Tính tổng các giá trị.
AVG([ALL DISTINCT] <i>biểu_thức</i>)	Tính trung bình của các giá trị
COUNT([ALL DISTINCT] <i>biểu_thức</i>)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn
MAX(<i>biểu_thức</i>)	Tính giá trị lớn nhất
MIN(<i>biểu_thức</i>)	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.

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 Huế:

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

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.

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
C24101	Toán K24	5
C24102	Tin K24	8
C24103	Lý K24	7
C24301	Sinh K24	5
C25101	Toán K25	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.

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.

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
```



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 COMPUTE và 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 (chẳng hạn xuất hiện trong một phép so sánh bằng)

Phép so sánh đối với kết quả truy vấn con

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

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 *TI-001*

```
SELECT * FROM monhoc
WHERE sodvht>=(SELECT sodvht FROM monhoc WHERE
mamonhoc='TI-001')
```

Câu lệnh dưới đây cho biết họ tên của những sinh viên lớp *TinK25*

```
SELECT hodem,ten
FROM sinhvien JOIN lop ON sinhvien.malop=lop.malop WHERE
tenlop='Tin K25' AND ngaysinh<ALL(SELECT ngaysinh FROM
sinhvien JOIN lop ON sinhvien.malop=lop.malop WHERE
lop.tenlop='Toán K25')
```

và câu lệnh:

```
SELECT hodem,ten FROM sinhvien JOIN lop on
sinhvien.malop=lop.malop WHERE tenlop='Tin K25' AND
year(ngaysinh)= ANY(SELECT year(ngaysinh) FROM sinhvien JOIN
lop ON sinhvien.malop=lop.malop WHERE lop.tenlop='Toán K25')
```

cho biết họ tên của những sinh viên lớp *TinK25*.

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)
```

Để hiển thị họ tên của những sinh viên lớp Tin K25 có năm sinh bằng với năm sinh của một sinh viên nào đó của lớp Toán K25, 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='Tin K25' AND
```




```
year(ngaysinh) IN (SELECT year(ngaysinh) FROM sinhvien JOIN
lop ON sinhvien.malop=lop.malop WHERE lop.tenlop='Toán K25')
```

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)
```

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.

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)
```

Tài liệu tham khảo:

[1]. Giáo trình SQL Server – Đỗ Ngọc Sơn, Phan Văn Viên - Tài liệu lưu hành nội bộ của Trường Đại học Công nghiệp Hà Nội, 2015.

[2]. Giáo trình hệ quản trị cơ sở dữ liệu - Đỗ Ngọc Sơn; Phan Văn Viên; Nguyễn Phương Nga - NXB Khoa học Kỹ thuật

[3]. Bài tập Hệ quản trị Cơ sở dữ liệu – Phạm Văn Hà, Trần Thanh Hùng, Đỗ Ngọc Sơn, Nguyễn Thị Thanh Huyền – Trường Đại học Công nghiệp Hà Nội, 2020.