Kỹ thuật phần mềm ứng dụng

Chương 5

Ngôn ngữ truy vấn cấu trúc SQL

Nội dung

- *5.1. Giới thiệu SQL*
- *5.2. Phân loại SQL*
- *5.3. Các lệnh SQL*

5.1 Giới thiệu SQL

- *Ngôn ngữ truy xuất CSDL quan hệ*
- *Là một ngôn ngữ phi thủ tục*
- *Là phương tiện được sử dụng để trao đổi với DBMS*
- *Câu lệnh giống ngôn ngữ tiếng Anh (dễ đọc, dễ hiểu hơn tiếng Anh).*
- *Những câu lệnh của SQL được sử dụng để trích rút và cập nhật dữ liệu trong một hoặc nhiều bảng của cơ sở dữ liệu.*
- *SQL làm việc với hầu hết các hệ quản trị cơ sở dữ liệu như MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase ..*

5.2. Phân loại SQL

- *DDL – Data Definition Language*
  - *Làm việc với cấu trúc CSDL*
- *DML – Data Manipulation Language*
  - *Làm việc với dữ liệu thực sự được lưu trữ*
- *DCL – Data Control Language*

5.3 Các lệnh SQL

- *Tạo và hủy cơ sở dữ liệu*
- *Tạo, hủy và sửa bảng dữ liệu*
- *Thêm, xóa, truy xuất dữ liệu từ bảng*

Tạo và hủy CSDL

1. Tạo CSDL: Cú pháp CREATE DATABASE <Tên CSDL>.

Ví dụ: create database QLSV

2. Hủy CSDL: Cú pháp DROP DATABASE <Tên CSDL>.

Ví dụ: drop database QLSV.

Tạo bảng dữ liệu(Table)

Cú pháp CREATE TABLE <Tên bảng> (<Column 1><Kiểu dữ liệu>[<RBTV>], <Column 2><Kiểu dữ liệu>[<RBTV>],……<Column n><Kiểu dữ liệu>[<RBTV>]).

Ràng buộc toàn vẹn – RBTV: NOT NULL, NULL, UNIQUE, DEFAULT, PRIMARY KEY, FOREIGN KEY /REFERENCES, CHECK

Ví dụ:

CREATE TABLE DM_KHOA (MA_KHOA INT PRIMARY KEY,TEN_KHOA NVARCHAR(50) NULL,GHI_CHU NVARCHAR(255) NULL)

Hủy và sửa bảng dữ liệu(Table)

– ***Hủy Table: Cú pháp DROP TABLE <Tên bảng>.***

Ví dụ:

```
DROP TABLE DM_KHOA
```

– ***Sửa Table:***

<u>Sửa cột của bảng</u>

ALTER TABLE <Tên bảng> ALTER COLUMN <New_Data_Type> [RBTV] .

Ví dụ:

```
ALTER TABLE DM_KHOA ALTER COLUMN TEN_KHOA NVARCHAR(100) NOT NULL
```

<u>Thêm cột của bảng</u>

ALTER TABLE <Tên bảng> ADD <Column_Name><Kiểu dữ liệu> [RBTV] .

Ví dụ:

```
ALTER TABLE DM_KHOA ADD SO_NAM_HOC INT NULL
```

Thêm, xóa dữ liệu khỏi bảng

- ***Thêm dữ liệu vào bảng:***

Cú pháp: INSERT INTO <Tên bảng>(Column1, Column2, .., Column n) VALUES (Giá trị 1, Giá trị 2, …., Giá trị n)

Ví dụ: Cho các lược đồ QH sau

- DM_KHOA (MA_KHOA,TEN_KHOA,GHI_CH)

- DM_SINHVIEN (MSSV, HO_TEN, NGAY_SINH, DIEN_THOAI)

Thêm khóa với vào bảng DM_KHOA:

INSERT INTO DM_KHOA (MA_KHOA,TEN_KHOA,GHI_CHU)

VALUES (1,'Khóa 30','Dài hạn')

- ***Xóa dữ liệu khỏi bảng:***

Cú pháp: DELETE FROM <Tên bảng> [WHERE <Điều kiện>]

Ví dụ:

DELETE FROM DM_KHOA WHERE MA_KHOA=1

Truy xuất dữ liệu

SELECT [DISTINCT] <*Column1*,.., *ColumnN*>

FROM <*Tên bảng 1*>,[<*Tên bảng 2*>],…

[WHERE <*Expression*>]

[GROUP BY g*Column1*, *gColumn2*,..

[HAVING <g*Expression*>]]

[ORDER BY *oColumn1*, *oColumn2* [DESC/ASC], …]

Truy xuất dữ liệu

– ***Truy xuất một cột trong bảng:***

Cú pháp SELECT <Column_Name> FROM <Tên bảng>.

Ví dụ:

SELECT TEN_KHOA FROM DM_KHOA

– ***Truy xuất nhiều cột trong bảng:***

Cú pháp: SELECT <Column_Name1,.., Column i> FROM <Tên bảng>.

Ví dụ:

SELECT TEN_KHOA,GHI_CHU FROM DM_KHOA

– ***Truy xuất tất cả các cột trong bảng:***

Cú pháp SELECT * FROM <Tên bảng>.

Ví dụ:

SELECT * FROM DM_KHOA

Chú ý: *Mệnh đề điều kiện WHERE <Điều kiện> thường được sử dụng trong câu lệnh truy xuất để tìm được một kết quả thỏa mãn điều kiện nào đó, ví dụ :*

*SELECT * FROM DM_KHOA WHERE TEN_KHÓA='Dài hạn'*

Mệnh đề Order By

- *Mục đích: sắp xếp lại dữ liệu theo một trật tự nhất định.*
- *Phải được đặt ở cuối của dòng lệnh.*
- *Có thể phát triển rộng ra khi thực hiện sắp xếp theo nhiều cột,*
  - *Ví dụ:*

  *SELECT * FROM DM_SINHVIEN ORDER BY HO_TEN,NGAY_SINH*
- *Ta có thể dùng chỉ số thứ tự của cột trong bảng để thay cho tên cột,*
  - *Ví du: thay cho câu lệnh trên ta có thể sử dụng câu lệnh sau: SELECT * FROM DM_SINHVIEN ORDER BY 2,3*
- *Chú ý: Mệnh đề ORDER BY sẽ mặc định sắp xếp theo chiều tăng dần, khi đó nếu muốn sắp xếp theo chiều giảm dần phải thêm cú pháp DESC sau mệnh đề ORDER BY,*
  - *ví dụ:*

  *SELECT * FROM DM_SINHVIEN ORDER BY HO_TEN DESC*

Mệnh đề WHERE

- *Mục đích: lọc dữ liệu theo một điều kiện nào đó. Khi đó SQL cung cấp cho người sử dụng một tập các toán tử để lọc dữ liệu theo miền mong muốn,*

TOÁN TỬ	DIỄN GIẢI
=	Ngang bằng
<>	Không ngang bằng
!=	Không ngang bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
!<	Không nhỏ hơn
>	Lớn hơn
>=	Lớn hơn hoặc bằng
!>	Không lớn hơn
BETWEEN	Giữa hai giá trị
IS NULL	Không có giá trị
IS NOT NULL	Giá trị khác Null

Chú ý: *Nếu trong câu lệnh* ***SELECT*** *cùng sử dụng mệnh đề* ***WHERE*** *và mệnh đề* ***ORDER BY*** *thì mệnh đề* ***WHERE*** *phải nằm trước mệnh đề* ***ORDER BY***

Toán tử LIKE

Đây là toán tử được sử dụng trong trình lọc thông minh của SQL, với việc sử dụng kết hợp với các ký tự đại diện toán tử LILE cung cấp rất nhiều các khả năng lọc giữ liệu một cách thuận tiện và phong phú.

- ***Ký tự đại diện % (SQL Server):***

- Tìm các cụm từ có ký tự đầu là một ký tự nào đó

cú pháp ***LIKE '<Ký tự cần tim>%'***.

Ví dụ:

Giả sử ta muốn tìm tất cả các sinh viên có họ bắt đầu bằng ký tự ***'n'****, Khi đó ta se có cú pháp như sau:*

SELECT * FROM DM_SINHVIEN WHERE HO_TEN LIKE 'n%'

- Tìm các cụm từ có ký tự cuối là một ký tự nào đó

cú pháp ***LIKE '%<Ký tự cần tim>'***

Ví dụ, ta muốn tìm tất cả các sinh viên có tên kết thúc là chữ i, khi đó cú pháp như sau:

SELECT * FROM DM_SINHVIEN WHERE HO_TEN LIKE '%i'

Toán tử LIKE

- Còn nếu khi ta không biết rõ cụm ký tự cần tìm ở nằm ở đâu trong các cụm ký tự thì ta dùng cú pháp như sau: ***LIKE '%<Cụm ký tự cần tim>%'.***

Ví dụ, ta muốn tìm tất cả các sinh viên mà họ tên có cụm từ 'nam', khi đó ta có cú pháp như sau:

SELECT * FROM DM_SINHVIEN WHERE HO_TEN LIKE '%nam%'

Chú ý: *Ký tự đại diện % được sử dụng rất đa dạng và hiệu quả, tùy thuộc vào cách đặt vị trí của nó.*

- *Ký tự đại diện (_)* ***– Chỉ dùng cho SQL Server:*** *Ký tự đại diện _ trên nguyên tắc được sử dụng giống như ký tự đại diện %, song có một điều khác biệt là ký tự đại diện _ chỉ đại diện cho một một ký tự duy nhất*

  Ví dụ: ***SELECT * FROM DM_KHOA WHERE TEN_KHOA LIKE 'a_a'***

- ***Ký tự đại diện khoảnh trống ([ ])- SQL Server :*** *Ký tự đại diện khoảng trống [ ] được sử dụng kết hợp với toán tử* ***LIKE*** *để tìm ra các đối tượng với một đặc điểm nào đó mà đối tượng đó thỏa mãn.*

  Ví dụ: *Ta có bảng danh mục khóa như sau:*

	MA_KHOA	TEN_KHOA	GHI_CHU
1	1	50	Dài hạn
2	2	51	Dài hạn
3	3	10	Tại chức
4	4	11	Tại chức
5	5	3	Hợp tác QT
6	6	4	Hợp tác QT

Khi đó nếu ta muốn tìm tất cả các bản ghi mà phần ghi chú của nó bắt đầu bằng chữ ***D*** *hoặc* ***H****, thì ta sử dụng cú pháp lọc như sau:* ***SELECT * FROM DM_KHOA WHERE GHI_CHU LIKE '[DH]%'****, kết quả nhận được như sau:*

	MA_KHOA	TEN_KHOA	GHI_CHU
1	1	50	Dài hạn
2	2	51	Dài hạn
3	5	3	Hợp tác QT
4	6	4	Hợp tác QT

Còn nếu ta sử dụng cú pháp ***SELECT * FROM DM_KHOA WHERE GHI_CHU LIKE '[^DH]%'****, ta sẽ nhận được kết quả như sau:*

	MA_KHOA	TEN_KHOA	GHI_CHU
1	3	10	Tại chức
2	4	11	Tại chức

Chú ý: *Không nên quá lạm dụng vào các ký tự đại diện, chỉ sử dụng chúng khi thật sự cần thiết.*

Một số hàm tính toán

- ***Hàm trung bình – AVG():*** *Dùng để tính giá trị trung bình cho một cột nào đó*
- *Hàm đếm số bản ghi – Count():*

  Đếm số lần xuất hiện của giá trị hoặc số dòng dữ liệu của bảng.

  Count(*) sẽ cho ra tất cả các các bản ghi kể cả các bản ghi có chứa giá trị Null,

  Count(Column_name) sẽ đưa chỉ đưa ra các cột có giá trị
- *Hàm lấy giá trị lớn nhất – MAX(): Dùng để lấy giá trị lớn nhất trong một cột nào đó.*
- *Hàm lấy giá trị nỏ nhất – MIN(): Dùng để lấy giá trị nhỏ nhất trong một cột nào đó. Cách sử dụng tương tự như đối với hàm MAX().*
- *Hàm lấy giá trị tổng – SUM(): Dùng để lấy giá trị tổng của một cột nào đó.*
- *Chú ý: Thường các hàm lấy giá trị tổng được sử dụng kèm với các mệnh đề GROUP BY và HAVING.*

Mệnh đề GROUP BY và HAVING

- *Mệnh đề **GROUP BY** dùng để nhóm các bản ghi có một số tính chất giống nhau, khi đó các tính chất đó sẽ có mặt trong mệnh đề **SELECT** và mệnh đề **GROUP BY**.*

  ***Ví dụ:** Ở phần trước ta dùng hàm **SUM()** để tình tổng các giá trị trong cột **SO_LUONG**, bây giờ ta lại muốn tính tổng số lượng cho từng loại vật tư, khi đó ta sẽ có cu pháp như sau:*

  SELECT TEN_VT,CHUNG_LOAI,DVT,SUM(SO_LUONG) AS SO_LUONG FROM DM_VT

  GROUP BY TEN_VT,CHUNG_LOAI,DVT

  *Mệnh đề **HAVING** dùng để lọc các bản ghi vừa được nhóm bằng mệnh đề **GROUP BY**.*

  ***Ví du:** Ta cần tính tổng của từng loại vật tư, tuy nhiên chỉ đối với các vật tư mà số lượng tổng của nó lơn hơn 50, khi đó ta sẽ sử dụng cú pháp sau:*

  SELECT TEN_VT,CHUNG_LOAI,DVT,SUM(SO_LUONG) AS SO_LUONG FROM DM_VT

  GROUP BY TEN_VT,CHUNG_LOAI,DVT

  HAVING SUM(SO_LUONG)>50

Mệnh đề GROUP BY và HAVING

Chú ý: *Nếu khi trong câu lựa chọn* ***SELECT*** *có dử dụng cả mệnh đề* ***WHERE*** *và* ***HAVING*** *thì mệnh đề* ***WHERE*** *sẽ được thực hiện trước khi nhóm, còn mệnh đề* ***HAVING*** *sẽ được thực hiện sau khi nhóm.*

Ví dụ: *Ta có cú pháp sau:*

```
SELECT TEN_VT,CHUNG_LOAI,DVT,SUM(SO_LUONG) AS SO_LUONG FROM DM_VT
WHERE CHUNG_LOAI='Xe nguyen chiec'
GROUP BY TEN_VT,CHUNG_LOAI,DVT
HAVING SUM(SO_LUONG)>50
```

Kết nối các bảng — SQL JOIN

- *Mục đích: kết nối các bảng khac nhau để lựa chọn được nhưng thông tin cần thiết tùy thuộc vào mục đích sử dụng.*
- ***Kết hợp hai bảng:*** *Giả sử ta có hai bảng* ***DM_KHOA****, và* ***DM_LOP*** *như sau:*

DM_KHOA

MA_KHOA	TEN_KHOA	GHI_CHU
1	Khoa 50	Dai han
2	Khoa 51	Dai han
4	Khoa 15	Tai chuc
5	Khoa 16	Tai chuc
6	Khoa 03	Hop tac QT
7	Khoa 04	Hop tac QT

DM_LOP

MA_LOP	TEN_LOP	LOP_TRUONG	GHI_CHU	MA_KHOA
1	03A	Nguyen Van Chien	NULL	6
2	03B	Le Thi Thu	NULL	6
3	03C	Tran Hoai Nam	NULL	6
4	51CNTT	Nguyen Van Tung	NULL	2
5	16VL	Hoang The Khanh	NULL	8

Kết nối các bảng — SQL JOIN

- *Nếu ta muốn đưa ra danh sách lớp cùng với tên khóa của lớp đó:*

  SELECT DM_LOP.TEN_LOP,DM_KHOA.TEN_KHOA FROM DM_LOP,DM_KHOA

  WHERE DM_LOP.MA_KHOA=DM_KHOA.MA_KHOA

  Và ta nhận được kết quả như sau:

TEN_LOP	TEN_KHOA
O3A	Khoa O3
O3B	Khoa O3
O3C	Khoa O3
51CNTT	Khoa 51

*Tuy nhiên thường trong thực tế có rất nhiều các yêu cầu khác nhau cho việc kết nối các bảng, và khi đó thay cho việc sử dụng việc so sánh các cột ở các bảng khác nhau người ta thường dùng các mệnh đề kết nối **JOIN** để thực hiện, sau đây là một số mệnh đề kết nối thông dụng nhất.*

Kết nối các bảng — SQL JOIN

- ***INNER JOIN**: Mệnh đề kết nối **INNER JOIN** sẽ trả về tất cả các dòng từ hai bảng thỏa mãn điều kiện so khớp. Các dòng mà không thỏa mãn điều kiện so khớp sẽ không có trong tập kết quả, cú pháp của mệnh đề **INNER JOIN** như sau:*

SELECT Table1.Tencot1, Table1. Tencot2...,Table2. Tencot1, Table.Tencot2.. FROM Table1 INNER JOIN Table2 ON Table1.Cotsosanh=Table2.Cotsosanh

***Ví dụ:** Ta có cú pháp sau:*

```
SELECT
DM_LOP.TEN_LOP,DM_LOP.LOP_TRUONG,DM_KHOA.TEN_KHOA
FROM DM_LOP
INNER JOIN DM_KHOA ON DM_LOP.MA_KHOA=DM_KHOA.MA_KHOA
```

Và kết quả nhận được là:

TEN_LOP	LOP_TRUONG	TEN_KHOA
03A	Nguyen Van Chien	Khoa 03
03B	Le Thi Thu	Khoa 03
03C	Tran Hoai Nam	Khoa 03
51CNTT	Nguyen Van Tung	Khoa 51

Kết nối các bảng — SQL JOIN

- ***LEFT JOIN:*** *Mệnh đề* ***LEFT JOIN*** *sẽ trả về tất cả các dòng có trong bảng 1 bất kể các dòng đó có so khớp với một dòng nào đó ở bên bảng 2 hay không. Cú pháp của mệnh đề* ***LEFT JOIN*** *như sau:*

SELECT Table1.Tencot1, Table1. Tencot1...,Table2. Tencot1, Table.Tencot2.. FROM Table1 LEFT JOIN Table2 ON Table1.Cotsosanh=Table2.Cotsosanh

Ví dụ: *Ta có cú pháp sau:*

SELECT DM_LOP.TEN_LOP,DM_LOP.LOP_TRUONG,DM_KHOA.TEN_KHOA FROM DM_LOP LEFT JOIN DM_KHOA ON DM_LOP.MA_KHOA=DM_KHOA.MA_KHOA

Khi đó ta sẽ nhận được kết quả như sau:

TEN_LOP	LOP_TRUONG	TEN_KHOA
03A	Nguyen Van Chien	Khoa 03
03B	Le Thi Thu	Khoa 03
03C	Tran Hoai Nam	Khoa 03
51CNTT	Nguyen Van Tung	Khoa 51
16VL	Hoang The Khanh	NULL

Kết nối các bảng — SQL JOIN

- *RIGHT JOIN: Mệnh đề này ngược với mệnh đề LEFT JOIN, nó trả về tất cả các hàng có mặt trong bảng thứ 2. Cú pháp như sau:*
- ***SELECT Table1.Tencot1, Table1. Tencot2...,Table2. Tencot1, Table.Tencot2.. FROM Table1 RIGHT JOIN Table2 ON Table1.Cotsosanh=Table2.Cotsosanh***

  Ví dụ: *Ta có cú pháp sau:*

  SELECT

  DM_LOP.TEN_LOP,DM_LOP.LOP_TRUONG,DM_KHOA.TEN_KHOA

  FROM DM_LOP

  RIGHT JOIN DM_KHOA ON DM_LOP.MA_KHOA=DM_KHOA.MA_KHOA

  Và kết quả nhận được sẽ là:

TEN_LOP	LOP_TRUONG	TEN_KHOA
NULL	NULL	Khoa 50
51CNTT	Nguyen Van Tung	Khoa 51
NULL	NULL	Khoa 15
NULL	NULL	Khoa 16
03A	Nguyen Van Chien	Khoa 03
03B	Le Thi Thu	Khoa 03
03C	Tran Hoai Nam	Khoa 03
NULL	NULL	Khoa 04

Kết nối các bảng — SQL JOIN

- ***Phép hợp – UNION:*** *Đây là phép toán dùng để nối hai hay nhiều câu truy vấn (Query) lại với nhau. Khác với mệnh đề* ***JOIN****, các kết nối của mệnh đề* ***JOIN*** *được thực hiện theo chiều ngang, còn đối với* ***UNION*** *kết nối dữ liệu được thực hiện theo chiều dọc.*

  Ví dụ: *Ta có hai câu truy vấn sau:*

  SELECT THOI_GIAN TEN_VT,CHUNG_LOAI,DVT,SO_LUONG FROM DM_VT

  WHERE CHUNG_LOAI='Xe nguyen chiec'

THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
2007-09-01 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	10
2007-09-02 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	20
2007-09-03 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	30
2007-09-04 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	50

SELECT THOI_GIAN,TEN_VT,CHUNG_LOAI,DVT,SO_LUONG FROM DM_VT

WHERE CHUNG_LOAI='Phu tung'

THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
2007-09-05 00:00:00.000	Nap hop xich WAVE 110	Phu tung	Cai	100
2007-09-06 00:00:00.000	Lop xe Fiture Neo	Phu tung	Cai	100
2007-09-07 00:00:00.000	Yen xe WAVE 110	Phu tung	Cai	200

Kết nối các bảng — SQL JOIN

- *Ta có thể sử dụng mệnh đề* ***UNION*** *để nối hai câu truy vấn trên như sau:*

```
SELECT THOI_GIAN,TEN_VT,CHUNG_LOAI,DVT,SO_LUONG  FROM
DM_VT
        WHERE CHUNG_LOAI='Xe nguyen chiec'
UNION
SELECT THOI_GIAN,TEN_VT,CHUNG_LOAI,DVT,SO_LUONG
FROM DM_VT
        WHERE CHUNG_LOAI='Phu tung'
```

Và kết quả nhận được sẽ như sau:

THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
2007-09-01 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	10
2007-09-02 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	20
2007-09-03 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	30
2007-09-04 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	50
2007-09-05 00:00:00.000	Nap hop xich WAVE 110	Phu tung	Cai	100
2007-09-06 00:00:00.000	Lop xe Fiture Neo	Phu tung	Cai	100
2007-09-07 00:00:00.000	Yen xe WAVE 110	Phu tung	Cai	200

Mệnh đề ORDER BY

Bây giờ giả sử ta muốn sắp xếp kết quả nhận được theo tên vật tư bằng cách sử dụng mệnh đề ORDER BY.

Khi đó ta sẽ có cu pháp sau:

```
SELECT THOI_GIAN,TEN_VT,CHUNG_LOAI,DVT,SO_LUONG  FROM DM_VT
        WHERE CHUNG_LOAI='Xe nguyen chiec‘
UNION
SELECT THOI_GIAN,TEN_VT,CHUNG_LOAI,DVT,SO_LUONG   FROM DM_VT
        WHERE CHUNG_LOAI='Phu tung‘
ORDER BY TEN_VT
```

Và kết quả nhận được là:

THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
2007-09-06 00:00:00.000	Lop xe Fiture Neo	Phu tung	Cai	100
2007-09-05 00:00:00.000	Nap hop xich WAVE 110	Phu tung	Cai	100
2007-09-02 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	20
2007-09-04 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	50
2007-09-01 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	10
2007-09-03 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	30
2007-09-07 00:00:00.000	Yen xe WAVE 110	Phu tung	Cai	200

*Như vậy để để sắp xếp kết quả nhận được khi sử dụng mệnh đề **UNION** bằng cách áp dụng mệnh đề **ORDER BY** thì mệnh đề **ORDER BY** phải được đặt ở câu truy vấn sau cùng.*

Truy vấn lồng nhau — Truy vấn con

Giả sử ta có hai bảng danh mục khóa học và danh mục lớp học như sau:

DM_KHOA

MA_KHOA	TEN_KHOA	GHI_CHU
1	Khoa 50	Dai han
2	Khoa 51	Dai han
4	Khoa 15	Tai chuc
5	Khoa 16	Tai chuc
6	Khoa 03	Hop tac QT
7	Khoa 04	Hop tac QT

DM_LOP

MA_LOP	TEN_LOP	LOP_TRUONG	GHI_CHU	MA_KHOA
1	03A	Nguyen Van Chien	NULL	6
2	03B	Le Thi Thu	NULL	6
3	03C	Tran Hoai Nam	NULL	6
4	51CNTT	Nguyen Van Tung	NULL	2
5	16VL	Hoang The Khanh	NULL	8

*Khi đó chẳng hạn ta muốn tìm tất cả các lớp thuộc khóa có tên là **"Khoa 03"**, khi đó ta có thể sử dụng cú pháp sau:*

```
SELECT * FROM DM_LOP
 WHERE MA_KHOA=(SELECT MA_KHOA FROM DM_KHOA WHERE
TEN_KHOA='Khoa 03')
```

Kết quả nhận được như sau:

MA_LOP	TEN_LOP	LOP_TRUONG	GHI_CHU	MA_KHOA
1	03A	Nguyen Van Chien	NULL	6
2	03B	Le Thi Thu	NULL	6
3	03C	Tran Hoai Nam	NULL	6

*Khi đó cú pháp **SELECT MA_KHOA FROM DM_KHOA WHERE TEN_KHOA='Khoa 03'** được gọi là truy vấn con (SubQuery).*

Phép toán IN

- *Cú pháp truy vấn ở trên có thể được viết lại như nhau:*

  SELECT * FROM DM_LOP

  WHERE MA_KHOA IN (SELECT MA_KHOA FROM DM_KHOA WHERE TEN_KHOA='Khoa 03')

  Kết quả nhận được tương tự như khi sử dụng dấu =

- *Tóan tử **IN** có nghĩa là có trong tập hợp trả về từ kết quả truy vấn, mệnh đề **NOT IN** thi kết quả nhận được sẽ ngược lại.*

  Ví dụ:

  SELECT * FROM DM_LOP

  WHERE MA_KHOA NOT IN (SELECT MA_KHOA FROM DM_KHOA WHERE TEN_KHOA='Khoa 03')

MA_LOP	TEN_LOP	LOP_TRUONG	GHI_CHU	MA_KHOA
1	03A	Nguyen Van Chien	NULL	6
2	03B	Le Thi Thu	NULL	6
3	03C	Tran Hoai Nam	NULL	6

Khi đó kết quả nhận được sẽ là:

MA_LOP	TEN_LOP	LOP_TRUONG	GHI_CHU	MA_KHOA
4	51CNTT	Nguyen Van Tung	NULL	2
5	16VL	Hoang The Khanh	NULL	8

Như vậy thực chất của phép toán IN là kiểm tra sự tồn tại của một giá trị trong một tập hợp

Mệnh đề SELECT ALL, DISTINCT

- ***Mệnh đề SELECT ALL:*** *Chọn tất cả các dòng trong bảng.*
- ***Mệnh đề DISTINCT****: Mệnh đề DISTINCT cho phép chọn nhưng có loại bỏ các cột trùng lặp thông tin. Nghĩa là DISTINCT chỉ có hiệu lực trên các trường có mặt trong mệnh đề* ***SELECT.***

  Ví dụ: *Ta có bảng danh mục vật tư như sau:*

MA_VT	THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
1	2007-09-01 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	10
2	2007-09-02 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	20
3	2007-09-03 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	30
4	2007-09-04 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	50
5	2007-09-05 00:00:00.000	Nap hop xich WAVE 110	Phu tung	Cai	100
6	2007-09-06 00:00:00.000	Lop xe Fiture Neo	Phu tung	Cai	100
7	2007-09-07 00:00:00.000	Yen xe WAVE 110	Phu tung	Cai	200

Khi đó nếu ta sử dụng cú pháp: ***SELECT DISTINCT TEN_VT,CHUNG_LOAI,DVT FROM DM_VT****, ta sẽ nhận được kết quả như sau:*

TEN_VT	CHUNG_LOAI	DVT
Lop xe Fiture Neo	Phu tung	Cai
Nap hop xich WAVE 110	Phu tung	Cai
Xe may Future II	Xe nguyen chiec	Chiec
Xe may WAVE 110	Xe nguyen chiec	Chiec
Yen xe WAVE 110	Phu tung	Cai

LỆNH UPDATE

- *Dùng để chỉnh sửa lại dữ liệu trong bảng dữ liệu, cú pháp như sau:*

  ***UPDATE** <Tenbang> **SET** <Column1=Value> [, <Column2=Value>,…]*

  *[**WHERE** <Dieukien>]*

  ***Ví dụ:** Ta có bảng danh mục vật tư như sau:*

MA_VT	THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
1	2007-09-01 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	10
2	2007-09-02 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	20
3	2007-09-03 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	30
4	2007-09-04 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	50
5	2007-09-05 00:00:00.000	Nap hop xich WAVE 110	Phu tung	Cai	100
6	2007-09-06 00:00:00.000	Lop xe Fiture Neo	Phu tung	Cai	100
7	2007-09-07 00:00:00.000	Yen xe WAVE 110	Phu tung	Cai	200

*Khi đó nếu ta muốn sửa số lượng nhập về kho của xe máy WAVE 110 ngày 01/09/2007 từ 10 chiếc thành 20 chiếc ta sẽ dùng cú pháp sau:**UPDATE DM_VT SET SO_LUONG=20 WHERE MA_VT=1**, kết quả sau khi sửa như sau:*

MA_VT	THOI_GIAN	TEN_VT	CHUNG_LOAI	DVT	SO_LUONG
1	2007-09-01 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	20
2	2007-09-02 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	20
3	2007-09-03 00:00:00.000	Xe may WAVE 110	Xe nguyen chiec	Chiec	30
4	2007-09-04 00:00:00.000	Xe may Future II	Xe nguyen chiec	Chiec	50
5	2007-09-05 00:00:00.000	Nap hop xich WAVE 110	Phu tung	Cai	100
6	2007-09-06 00:00:00.000	Lop xe Fiture Neo	Phu tung	Cai	100
7	2007-09-07 00:00:00.000	Yen xe WAVE 110	Phu tung	Cai	200

Các loại ràng buộc - Constraints

- *Đó là tập các quy tắc nhằm kiểm soát các thao tác đối với dữ liệu như ngăn chặn dữ liệu sai nhập vào CSDL, cảnh báo người dùng hoặc ứng dụng không xóa dữ liệu đúng ..vvv*
- *Gồm có:*

❖ ***Các ràng buộc ở mức cao***

❖ *Các ràng buộc ở mức đặc thù*

Các loại ràng buộc ở mức cao

- *Ràng buộc thực thể – Entity Constraints: Ràng buộc này yêu cầu không tồn tại hai dòng trùng nhau, để thực hiện điều này một hay nhiều cột được đánh dấu là duy nhất. Cột hoặc các cột đó được thiết kế là Primary, Candidate hoặc Alternate Key.*
- *Ràng buộc miền – Domain Constraints: Ràng buộc này yêu cầu chỉ có các giá trị hợp lệ tồn tại trong mỗi cột bao gồm có hay không có cột chấp nhận giá trị* ***Null****. Ví dụ, nếu ta chỉ định dữ liệu của cột* ***THOI_GIAN*** *trong bảng có kiểu ngày giờ, thì khi đó nếu ta cố tình đưa dữ liệu kiểu ký tự vào cột đó thì hệ quản trị sẽ không chấp nhận và báo lỗi.*
- *Ràng buộc toàn vẹn tham chiếu – Reference integrity Constraints: Ràng buộc này được thự hiện khi có hai hay nhiều bảng có quan hệ với nhau. Nó kiểm tra giá trị của cột có phù hợp với cột trong bảng khác có quan hệ với bảng hiện chứa cột ràng buộc hay không.*

Các loại ràng buộc ở mức đặc thù

+ *Ràng buộc khóa chính – Primary Key Constraints: Ràng buộc khóa chính bảo đảm giá trị của một cột hay các cột nào đó là duy nhất trong bảng và không chứa giá trị Null, và trong một bảng chỉ có một khóa chính. Trong MS Access và SQL Server khi một khóa chính được thiết lập thì đồng thời một chỉ mục duy nhất cũng được thiết lập trên cột hoặc các cột của khóa chính. Ta có thể dùng từ khóa* ***PRIMARY KEY*** *để thiết lập khóa chính trong khi thiết kế bảng, hoặc sử dụng cú pháp* ***ALTER TABLE*** *để thêm dàng buộc khóa chính với cú pháp* ***ADD CONSTRAINT****, hoặc xóa bỏ dàng buộc với cú pháp* ***DROP CONSTRAINT****.*

+ *Ràng buộc duy nhất – Unique Constraints: Ràng buộc này yêu cầu tính duy nhất trên một cột nào đó của bảng dữ liệu. Một bảng có thể có nhiều định danh duy nhất. Khi đó nếu ta cố tình đưa một giá trị trùng lắp vào cột được thiết lập ràng buộc duy nhất, thì hệ thống sẽ không chấp nhận và báo lỗi.*

+ *Ràng buộc kiểm tra – Check Constraints:Dàng buộc Check tuân theo dàng buộc miền, nó sẽ kiểm tra xem dữ liệu trong cột được thiết lập ràng buộc có nằm trong miền giá trị được định nghĩa khi tạo ràng buộc không*

Các loại ràng buộc ở mức đặc thù (tiếp)

+ *Ràng buộc mặc định – Default Constraint: Ràng buộc Default chỉ ra một giá trị được tự động đưa vào cột được thiết lập ràng buộc Default nếu trong quá trình thêm mới dữ liệu không cung cấp một giá trị rõ ràng cho cột đó. Giá trị chỉ định trong ràng buộc Default phải tương thích với kiểu dữ liệu của cột được thiết lập ràng buộc.*

Ràng buộc khóa ngoại – Foreign Key Constraints: Ràng buộc khóa ngoại phục vụ cho hai mục đích: nó tuân theo toàn vẹn tham chiếu bằng việc kiểm tra quan hệ giữa các bảng, và nó tuân theo toàn vẹn miền trên cột hay các cột khóa ngoại bằng việc chỉ cho phép các hợp lệ. Một ràng buộc khóa ngoại thường tham chiếu đến giá trị khóa chính của bảng cha, tuy nhiên nó cũng có thể tham chiếu đến bất kỳ giá trị của các khóa duy nhất khác của bảng cha (cột hay các cột có sàng buộc duy nhất UNIQUE).