Ngôn ngữ SQL

SQL là ngôn ngữ truy vấn có cấu trúc (Structured Query Language), là một ngôn ngữ máy tính để lưu trữ, thao tác và truy xuất dữ liệu được lưu trữ trong cơ sở dữ liệu quan hệ.

SQL là ngôn ngữ tiêu chuẩn cho Hệ thống cơ sở dữ liệu quan hệ (Relational Database System). Tất cả các Hệ thống quản trị cơ sở dữ liệu quan hệ (RDBMS) như MySQL, MS Access, Oracle, Sybase, Informix, Postgres và SQL Server đều sử dụng SQL làm ngôn ngữ cơ sở dữ liệu chuẩn của chúng. Tuy nhiên, có một ít khác biệt giữa các RDMS này như SQL Server dùng T-SQL, Oracle dùng PL/SQL, Access dùng JET SQL,…

SQL được dùng phổ biến những ưu điểm sau:

- Cho phép người dùng truy cập dữ liệu trong hệ thống quản trị cơ sở dữ liệu quan hệ (RDBMS).
- Cho phép người dùng mô tả dữ liệu.
- Cho phép người dùng định nghĩa dữ liệu trong cơ sở dữ liệu và thao tác với dữ liệu đó.
- Cho phép nhúng trong các ngôn ngữ khác bằng cách sử dụng mô-đun, thư viện và trình biên dịch trước SQL.
- Cho phép người dùng tạo (create), thả (drop) cơ sở dữ liệu và bảng.
- Cho phép người dùng tạo chế độ xem (views), thủ tục lưu trữ (stored procedures), các chức năng (functions) trong cơ sở dữ liệu.
- Cho phép người dùng thiết lập quyền trên bảng (tables), thủ tục (procedures) và chế độ xem (views).

Học ngôn ngữ SQL tại https://www.w3schools.com/sql/default.asp. Sau đây là một số lệnh SQL cơ bản với dữ liệu minh họa bên cạnh các bảng SINHVIEN và KHOA bao gồm:

SINHVIEN

MaSV	TenSV	NgaySinh	QueQuan	TBC	MaKH
SV01	Minh	15/12/2000	Cam Ranh	5.6	KH01
SV02	Dũng	14/03/2001	Cam Ranh	7.0	KH01
SV03	Nam	01/06/2000	Ninh Hòa	6.5	KH02
SV04	Khiêm	06/12/2002	Nha Trang	5.5	KH03
SV05	Hòa	17/08/2009	Nha Trang	7.8	KH01
SV06	Khải	15/03/2000	Ninh Hòa	6.5	KH03

Ý nghĩa các cột:

- MaSV: Mã sinh viên
- TenSV: Tên sinh viên
- NgaySinh: Ngày tháng năm sinh của sinh viên
- QueQuan: Quê quán sinh viên
- TBC: Điểm trung bình chung học tập của sinh viên
- MaKH: Mã khoa tương ứng với khoa sinh viên đăng ký học

KHOA

MaKH	TenKH
KH01	Công nghệ thông tin
KH02	Cơ khí
KH03	Cơ bản

Ý nghĩa các cột:

- MaKH: Mã khoa
- TenKH: Tên khoa

Một số hình ảnh minh họa cũng được mượn từ https://www.w3schools.com.

1. Lệnh SELECT…FROM

Dùng để chọn các cột cần hiển thị trong một hay nhiều bảng từ cơ sở dữ liệu. Cú pháp:

```
SELECT cột_1, cột_2, ...
FROM tên_bảng;
```

Có thể hiển thị tất cả các cột trong một bảng với lệnh SELECT *. Ví dụ muốn hiển thị chỉ cột *TenSV* của bảng ***SINHVIEN*** chúng ta viết:

```
SELECT TenSV
FROM SINHVIEN;
```

Hiển thị tất cả dữ liệu từ bảng SINHVIEN có thể viết:

```
SELECT *
FROM SINHVIEN;
```

Kết quả sẽ hiển thị tất cả các cột từ bảng SINHVIEN.

2. Lệnh DISTINCT

Dữ liệu trong các bảng khi hiển thị bằng lệnh SELECT có thể trùng nhau, ví dụ hiển thị cột MaKH của bảng SINHVIEN bằng lệnh sau:

```
SELECT MaKH
FROM SINHVIEN;
```

Kết quả có thể như sau:

KH01

KH01

KH02

KH03

KH01

KH03

Chúng ta có thể dùng lệnh *DISTINCT* để hiển thị các dữ liệu không trùng nhau theo cú pháp sau:

SELECT DISTINCT *cột_1, cột_2, ...*

FROM *tên_bảng*;

Ví dụ hiển thị lại MaKH từ bảng SINHVIEN:

```
SELECT DISTINCT MaKH
FROM SINHVIEN;
```

Kết quả:

KH01

KH02

KH03

3. Lệnh WHERE

Chúng ta có thể hiển thị dữ liệu từ một hay nhiều bảng theo các điều kiệu nào đó bằng lệnh *WHERE*. Cú pháp:

SELECT *cột_1, cột_2, ...*

FROM *tên_bảng*

WHERE *điều_kiện*;

Ví dụ hiển thị thông tin *các sinh viên* có MaKH là KH01, lệnh SQL như sau:

```
SELECT *
FROM SINHVIEN
WHERE MaKH = 'KH01';
```

4. Lệnh AND, OR, NOT

Khi dùng *WHERE* chúng ta có thể dùng các lệnh *AND*, *OR*, hay *NOT* để kết nối các điều kiện khác nhau (*AND*, *OR*) hay phủ định giá trị một biểu thức điều kiện (*NOT*). Cú pháp *AND*:

SELECT *cột_1, cột_2, ...*

FROM *tên_bảng*

WHERE *btdk_1* ***AND*** *btdk_2* ***AND*** *btdk_3* ***AND*** *...;*

Hiển thị kết quả khi tất cả các điều kiện *btdk_1, btdk_2, btdk_3,…*thỏa mãn đồng thời. Ví dụ hiển thị thông tin các sinh viên có MaKH là KH01 và QueQuan là Cam Ranh:

```
SELECT *
FROM SINHVIEN
WHERE MaKH = 'KH01' AND QueQuan = 'Cam Ranh';
```

Nếu muốn hiển thị thông tin chỉ cần thỏa mãn một hay một vài điều kiện nào đó chúng ta có thể dùng *OR* theo cú pháp:

SELECT *cột_1, cột_2, ...*

FROM *tên_bảng*

WHERE *btdk_1* ***OR*** *btdk_2* ***OR*** *btdk_3* ***OR*** *...;*

Ví dụ:

```
SELECT *
FROM SINHVIEN
WHERE MaKH = 'KH01' OR QueQuan = 'Cam Ranh';
```

Lệnh SQL trên hiển thị các sinh viên thỏa mãn hoặc có MaKH là KH01 hoặc có QueQuan là Cam Ranh hoặc cả hai.

Trong trường hợp muốn hiển thị thông tin theo một ngoại lệ nào đó, ví dụ hiển thị thông tin tất cả sinh viên ngoại trừ sinh viên đến từ Cam Ranh, chúng ta có thể dùng lệnh NOT theo cú pháp:

```
SELECT cột_1, cột_2, ...
FROM tên_bảng
WHERE NOT biểu_thức_điều_kiện;
```

Ví dụ lệnh SQL

```
SELECT *
FROM SINHVIEN
WHERE NOT QueQuan = 'Cam Ranh';
```

Sẽ hiển thị tất cả thông tin sinh viên ngoại trừ sinh viên có QueQuan là Cam Ranh.

5. Lệnh IN, BETWEEN, LIKE

Bên cạnh AND, OR, NOT, các lệnh IN, BETWEEN, và LIKE cũng thường được dùng trong biểu thức điều kiện của WHERE.

Lệnh BETWEEN dùng để chọn giá trị trong một khoảng nào đó. Cú pháp:

```
SELECT Tên_cột_hiển_thị
FROM Tên_bảng
WHERE Tên_cột BETWEEN giá_trị_1 AND giá_trị_2;
```

Ví dụ lệnh SQL hiển thị danh sách sinh viên có điểm trung bình chung TBC từ 6.5 đến 7.8:

```
SELECT *
FROM SINHVIEN
WHERE TBC BETWEEN 6.5 AND 7.8;
```

Lệnh IN là một hình thức khác của lệnh OR. Cú pháp:

```
SELECT Tên_cột_hiển_thị
FROM Tên_bảng
WHERE Tên_cột IN (giá_trị_1, giá_trị_2,...);
```

Lệnh này tương đương với lệnh OR như sau:

SELECT *Tên_cột_hiển_thị*

FROM *Tên_bảng*

WHERE *Tên_cột* = *giá_trị_1* ***OR*** tên_cột = *giá_trị_2* ***OR****...*;

Ví dụ:

```
SELECT *
FROM SINHVIEN
WHERE MaKH IN ('KH01', 'KH02');
```

Tương đương:

```
SELECT *
FROM SINHVIEN
WHERE MaKH = 'KH01' OR MaKH = 'KH02';
```

Ngoài việc dùng IN, BETWEEN, chúng ta có thể dùng LIKE để xác định dữ liệu trong những điều kiện phức tạp hơn như tìm kiếm các khách hàng có tên bắt đầu bằng chữ a hay tên có chữ cái thứ hai là r, v.v. Cú pháp LIKE:

SELECT *Tên_cột_hiển_thị*

FROM *Tên_bảng*

WHERE *Tên_cột* ***LIKE*** *mẫu_điều_kiện*;

mẫu_điều_kiện là biểu thức kết hợp các ký hiệu đặc biệt (wildcard) và hai kí hiệu đặc biệt thường dùng là:

- %: đại diện cho không, một, hay nhiều ký tự
- _: đại diện cho một ký tự

Ví dụ lệnh SQL:

```
SELECT *
FROM SINHVIEN
WHERE TenSV LIKE 'k%';
```

Lệnh SQL trên hiển thị thông tin các sinh viên có tên bắt đầu bằng chữ **k** (hay **K**).

Một lệnh SQL khác:

```
SELECT *
```

```
FROM SINHVIEN
WHERE TenSV LIKE 'k_%_%_%_%';
```

Lệnh SQL trên hiển thị thông tin các sinh viên có tên bắt đầu bằng chữ **k** (hay **K**) và tên chứa ít nhất 5 ký tự.

Tham khảo thêm về cách dùng các wildcards tại
https://www.w3schools.com/sql/sql_wildcards.asp

6. Lệnh ORDER BY

Lệnh ORDER BY dùng để sắp xếp kết quả theo thứ tự tăng (ASC) hay giảm (DESC) dần theo các cột. Cú pháp:

SELECT *cột_1, cột_2, ...*

FROM *tên_bảng*

[***WHERE*** *biểu_thức_điều_kiện* **]**

ORDER BY *cột_1, cột_2, ...**ASC*** | ***DESC;***

Lệnh *WHERE* đặt trong dấu [] vì có thể có hoặc không khi dùng *ORDER BY*. Mặc định khi dùng ORDER BY, các giá trị được xếp tăng dần; nếu muốn xếp giảm dần có thể thêm từ khóa *DESC*. Ví dụ sắp xếp thông tin sinh viên tăng dần theo điểm TBC:

```
SELECT MaSV, TenSV, TBC
FROM SINHVIEN
ORDER BY TBC;
```

Có thể sắp xếp giảm dần theo TBC:

```
SELECT MaSV, TenSV, TBC
FROM SINHVIEN
ORDER BY TBC DESC;
```

7. Lệnh INSERT INTO…VALUES

Chúng ta có thể thêm thông tin đến bảng với lệnh *INSERT INTO* theo cú pháp sau:

INSERT INTO *tên_bảng (cột_1, cột_2, cột_3, ...)*

VALUES *(giá_trị_1, giá_trị_2, giá_trị_3, ...);*

Thêm các giá trị giá_trị_1, giá_trị_2, giá_trị_3, ...tương ứng đến các cột cột_1, cột_2, cột_3, ... của bảng. Nếu chúng ta thêm các giá trị đến tất cả các cột của bảng thì dùng cú pháp:

```
INSERT INTO tên_bảng
VALUES (giá_trị_1, giá_trị_2, giá_trị_3, ...);
```

Ví dụ:

```
INSERT INTO KHOA (MaKH, TenKH)
VALUES ('KH04', 'Môi Trường');
```

8. Lệnh UPDATE

Dùng lệnh *UPDATE* để chỉnh sửa, thay đổi các thông tin có sẵn từ các bảng của cơ sở dữ liệu. Cú pháp:

```
UPDATE tên_bảng
SET cột_1 = giá_trị_1, cột_2 = giá_trị_2, ...
WHERE điều_kiện;
```

Cập nhật giá trị các cột *cột_1*, *cột_2*,…tương ứng với *giá_trị_1*, *giá_trị_2*,…Cần chú ý sử dụng WHERE để cập nhật tại một vài hàng dữ liệu nào đó, nếu không dùng WHERE thì tất cả các hàng trong bảng sẽ được cập nhật.

Ví dụ lệnh SQL:

```
UPDATE KHOA
SET TenKH = 'Môi trường'
WHERE MaKH = 'KH01';
```

Cập nhận thông tin bảng KHOA tại MaKH là KH01.

9. Lệnh DELETE

Dùng lệnh **DELETE** để xóa các dữ liệu tồn tại sẵn trong cơ sở dữ liệu. Cú pháp:

```
DELETE FROM tên_bảng
WHERE điều_kiện;
```

Sử dụng WHERE để xác định các hàng cần xóa trong bảng. Nếu bỏ qua lệnh WHERE, tất cả các hàng của bảng sẽ bị xóa.

Ví dụ lệnh SQL sau:

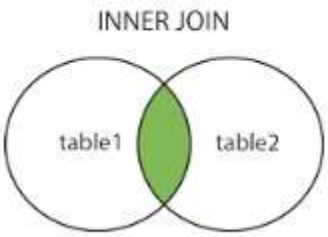
```
DELETE FROM SINHVIEN
WHERE TenSV = 'Khải';
```

Xóa tất cả thông tin các sinh viên có tên là Khải từ bảng SINHVIEN.

10. Lệnh INNER JOIN, LEFT JOIN, RIGHT JOIN, OUTER JOIN

Trong trường hợp muốn hiển thị dữ liệu được lấy từ nhiều bảng có quan hệ với nhau, chúng ta có thể dùng các lệnh JOIN.

Lệnh ***INNER JOIN*** hiển thị dữ liệu thuộc về cả hai bảng. Hình ảnh trực quan ***INNER JOIN*** với hai bảng ***table1*** và ***table2*** như sau:



Nguồn https://www.w3schools.com/

Cú pháp:

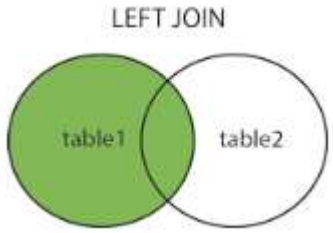
```
SELECT Các_cột_hiển_thị
FROM Tên_bảng_thứ_nhất (table1)
INNER JOIN Tên_bảng_thứ_hai (table2)
ON Tên_bảng_thứ_nhất.Tên_cột = Tên_bảng_thứ_hai.Tên_cột;
```

Các cột trên lệnh ON là các cột thể hiện mối quan hệ giữa hai bảng và các cột này có thể khác tên nhưng có cùng giá trị. Ví dụ bảng SINHVIEN và bảng KHOA có quan hệ với nhau dựa trên cột MaKH (khóa chính (PK) bảng KHOA) và cột MaKH (trong bảng SINHVIEN – *khóa ngoại (FK)*).

Ví dụ lệnh SQL sau hiển thị các thông tin MaSV, TenSV, TenKH, TBC từ các bảng SINHVIEN và KHOA:

```
SELECT SINHVIEN.MaSV, SINHVIEN.TenSV, KHOA.TenKH, SINHVIEN.TBC
FROM SINHVIEN
INNER JOIN KHOA
ON KHOA.MaKH = SINHVIEN.MaKH;
```

Lệnh ***LEFT JOIN*** hiển thị dữ liệu thuộc về bảng bên trái và dữ liệu thuộc về cả hai bảng. Hình ảnh trực quan ***LEFT JOIN*** với hai bảng ***table1*** và ***table2*** như sau:

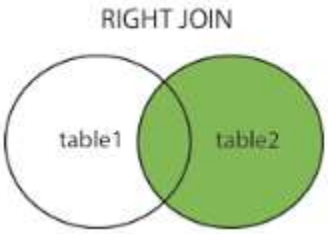


Nguồn https://www.w3schools.com/

Cú pháp:

```
SELECT Các_cột_hiển_thị
FROM Tên_bảng_thứ_nhất (table1)
LEFT JOIN Tên_bảng_thứ_hai (table2)
ON Tên_bảng_thứ_nhất.Tên_cột = Tên_bảng_thứ_hai.Tên_cột;
```

Lệnh ***RIGHT JOIN*** hiển thị dữ liệu thuộc về bảng bên phải và dữ liệu thuộc về cả hai bảng. Hình ảnh trực quan ***RIGHT JOIN*** với hai bảng ***table1*** và ***table2*** như sau:

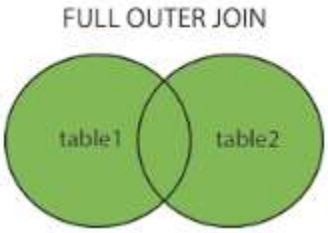


Nguồn https://www.w3schools.com/

Cú pháp:

```
SELECT Các_cột_hiển_thị
FROM Tên_bảng_thứ_nhất (table1)
RIGHT JOIN Tên_bảng_thứ_hai (table2)
ON Tên_bảng_thứ_nhất.Tên_cột = Tên_bảng_thứ_hai.Tên_cột;
```

Lệnh ***FULL OUTER JOIN*** hiển thị tất cả dữ liệu thuộc về cả hai bảng. Hình ảnh trực quan ***FULL OUTER JOIN*** với hai bảng ***table1*** và ***table2*** như sau:



Nguồn https://www.w3schools.com/

Cú pháp:

```
SELECT Các_cột_hiển_thị
FROM Tên_bảng_thứ_nhất (table1)
FULL OUTER JOIN Tên_bảng_thứ_hai (table2)
ON Tên_bảng_thứ_nhất.Tên_cột = Tên_bảng_thứ_hai.Tên_cột;
```

11. Các hàm thống kê

SQL hỗ trợ một số hàm phục vụ cho việc thống kê số liệu trong các bảng. Chúng ta sẽ cùng tìm hiểu một số hàm thống kê cơ bản.

Hàm MAX và MIN

Hàm MAX và MIN lần lượt trả về giá trị lớn nhất và nhỏ nhất của cột được chọn. Cú pháp hàm MIN:

```
SELECT MIN(Tên_cột_được_chọn)
FROM Tên_bảng
[WHERE điều_kiện];
```

Điều kiện WHERE có thể có hoặc không. Ví dụ lệnh SQL sau sẽ trả về giá trị nhỏ nhất trong cột TBC của bảng SINHVIEN:

```
SELECT MIN(TBC)
FROM SINHVIEN;
```

Chú ý: Cột có tiêu đề là MIN (TBC) là cột phát sinh trong quá trình truy vấn. Nếu muốn tên cột phù hợp hơn, ví dụ thay vì là MIN (TBC) theo mặc định chúng ta sẽ thay đổi thành MINTBC, chúng ta sẽ dùng từ khóa ***AS*** như sau:

```
SELECT MIN(TBC) AS MINTBC
FROM SINHVIEN;
```

Đối lập với hàm MIN là hàm MAX với cú pháp:

SELECT MAX(*Tên_cột_được_chọn*)

FROM *Tên_bảng*

[***WHERE*** *điều_kiện*];

Điều kiện WHERE có thể có hoặc không. Ví dụ lệnh SQL sau sẽ trả về giá trị lớn nhất trong cột TBC của bảng SINHVIEN:

```
SELECT MAX(TBC)
FROM SINHVIEN;
```

Có thể đặt tên cột thông qua từ khóa AS tương tự MIN.

Hàm SUM

Hàm SUM dùng để tính tổng các giá trị của cột được chọn. Cú pháp:

SELECT SUM(*Tên_cột_được_chọn*)

FROM *Tên_bảng*

[***WHERE*** *điều_kiện*];

Điều kiện WHERE có thể có hoặc không. Ví dụ lệnh SQL sau tính tổng các giá trị từ cột TBC của bảng SINHVIEN:

```
SELECT SUM(TBC)
FROM SINHVIEN;
```

Có thể đặt tên cột thông qua từ khóa AS tương tự MIN, MAX.

Hàm AVG

Hàm AVG dùng để tính giá trị trung bình các giá trị của cột được chọn. Cú pháp:

SELECT AVG(*Tên_cột_được_chọn*)

FROM *Tên_bảng*

[***WHERE*** *điều_kiện*];

Điều kiện WHERE có thể có hoặc không. Ví dụ lệnh SQL sau tính giá trị trung bình các giá trị từ cột TBC của bảng SINHVIEN:

```
SELECT AVG(TBC)
FROM SINHVIEN;
```

Có thể đặt tên cột thông qua từ khóa AS tương tự MIN, MAX, SUM.

Hàm COUNT

Hàm COUNT dùng để đếm các giá trị của cột được chọn. Cú pháp:

SELECT COUNT(*Tên_cột_được_chọn*)

FROM *Tên_bảng*

[***WHERE*** *điều_kiện*];

Điều kiện WHERE có thể có hoặc không. Ví dụ lệnh SQL sau đếm sinh viên dựa vào cột MaSV của bảng SINHVIEN:

```
SELECT COUNT(MaKH)
FROM SINHVIEN;
```

Có thể đặt tên cột thông qua từ khóa AS tương tự MIN, MAX, SUM, AVG.

12. Lệnh GROUP BY

Lệnh GROUP BY thường kết hợp với các hàm thống kê dùng để nhóm dữ liệu theo một hay nhiều cột. Cú pháp tổng quát:

SELECT *Tên_các_cột_hiển_thị*

FROM *Tên_bảng*

[***WHERE*** *điều_kiện*]

GROUP BY *Tên_các_cột_dùng_để_nhóm_dữ_liệu*

[***ORDER BY*** *Tên_các_cột_dùng_để_sắp_xếp*];

Các lệnh WHERE và ORDER BY có thể có hoặc không. Lệnh SQL sau dùng để đếm sinh viên trong bảng SINHVIEN theo QueQuan như sau:

```
SELECT COUNT(MaSV), QueQuan
FROM SINHVIEN
GROUP BY QueQuan;
```

Có thể kết hợp với lệnh GROUP BY để sắp xếp giá trị giảm dần theo số lượng thống kê:

```
SELECT COUNT(MaSV), QueQuan
FROM SINHVIEN
GROUP BY QueQuan
```

```
ORDER BY COUNT(MaSV) DESC;
```

12. Lệnh HAVING

Giống như WHERE, lệnh HAVING dùng để xác định điều kiện để lọc dữ liệu nhưng khác với WHERE, HAVING dùng kết hợp với các hàm thống kê và GROUP BY. Cú pháp tổng quát:

SELECT *Tên_các_cột_hiển_thị*

FROM *Tên_bảng*

[***WHERE*** *điều_kiện*]

[GROUP BY *Tên_các_cột_dùng_để_nhóm_dữ_liệu]*

HAVING *điều_kiện*

[***ORDER BY*** *Tên_các_cột_dùng_để_sắp_xếp*];

Các lệnh WHERE, ORDER BY có thể có hoặc không. Ví dụ lệnh SQL sau dùng để đếm sinh viên trong bảng SINHVIEN dựa trên theo QueQuan và chỉ hiển thị những quê quán (QueQuan) nào có ít nhất 2 sinh viên:

```
SELECT COUNT(MaSV), QueQuan

FROM SINHVIEN

GROUP BY QueQuan

HAVING COUNT(MaSV) >= 2;
```

Tài liệu này chỉ nêu một số lệnh SQL cơ bản và thường sử dụng nhất. Chi tiết về ngôn ngữ SQL có thể tìm hiểu tại https://www.w3schools.com/sql/default.asp.