

Welcome to SQL tutorial

Mục lục

SQL căn bản

[Giới thiệu SQL](#)

Mô tả thế nào là SQL, cách dùng SQL.

[SQL Select](#)

Cách dùng phát biểu SELECT để chọn dữ liệu từ một bảng trong SQL.

[SQL Where](#)

Cách dùng mệnh đề WHERE để chỉ định tiêu chuẩn chọn.

[SQL And & Or](#)

Cách dùng AND và OR để kết nối hai hay nhiều điều kiện trong mệnh đề WHERE.

[SQL Between](#)

Cách dùng BETWEEN....AND để tìm dữ liệu trong một khoảng giới hạn.

[SQL Distinct](#)

Cách dùng từ khóa DISTINCT để chỉ trả về các trị khác nhau trong một cột.

[SQL Order By](#)

Cách dùng từ khóa ORDER BY để trả về các hàng được sắp xếp theo một thứ tự định trước.

[SQL Insert](#)

Cách dùng phát biểu INSERT để chèn hàng mới vào trong một bảng.

[SQL Update](#)

Cách dùng phát biểu UPDATE để cập nhật hay thay đổi các hàng trong một bảng.

[SQL Delete](#)

Cách dùng phát biểu DELETE để xóa các hàng trong một bảng.

[SQL Count](#)

Giải thích các hàm COUNT tạo sẵn trong SQL.

SQL nâng cao

[Các hàm SQL](#)

Giải thích cách dùng các hàm tạo sẵn trong SQL.

[SQL Group By](#)

Giải thích cách dùng hàm GROUP BY tạo sẵn trong SQL.

[Các bí danh SQL](#)

Giải thích cách dùng các bí danh (alias) cho các tên cột và các tên bảng.

[SQL Join](#)

Giải thích cách chọn thông tin từ nhiều bảng.

[SQL Create](#)

Cách tạo các cơ sở dữ liệu và các bảng, và cách xóa chúng.

[SQL Alter](#)

Cách dùng phát biểu ALTER TABLE để thêm hay loại các cột trong một bảng cho trước.

Giới thiệu SQL

SQL là một ngôn ngữ theo chuẩn ANSI để truy xuất các cơ sở dữ liệu.

SQL là gì?

- SQL là **Structured Query Language** – Ngôn ngữ Truy vấn có Cấu trúc
- SQL cho phép bạn truy xuất một cơ sở dữ liệu
- SQL là một ngữ theo chuẩn ANSI
- SQL có thể thực hiện các truy vấn đến một cơ sở dữ liệu
- SQL có thể truy tìm dữ liệu từ một cơ sở dữ liệu
- SQL có thể chèn các mẫu tin mới vào trong một cơ sở dữ liệu
- SQL có thể xóa các mẫu tin trong một cơ sở dữ liệu

- SQL có thể cập nhật các mẫu tin trong một cơ sở dữ liệu
- SQL rất dễ học

SQL là một chuẩn

SQL là một chuẩn ANSI (American National Standards Institute - Viện Tiêu chuẩn Quốc gia Mỹ) cho các hệ thống truy xuất cơ sở dữ liệu. Các phát biểu SQL dùng để truy tìm và cập nhật dữ liệu trong một cơ sở dữ liệu.

SQL làm việc với các trình quản lý cơ sở dữ liệu như Access, DB2, Informix, Microsoft SQL Server, Oracle, Sybase, và nhiều trình khác (đáng tiếc là đa số trong chúng có các phần mở rộng ngôn ngữ SQL riêng).

Các bảng cơ sở dữ liệu

Cơ sở dữ liệu chứa các đối tượng gọi là các **Bảng** (Tables).

Các **Mẫu tin** (Records) lưu trong các bảng này. Các bảng được gọi theo tên bảng (như "Persons", "Orders", "Suppliers").

Các bảng chứa các **Cột** (Columns) và các **Dòng** (Rows) dữ liệu. Dòng chứa các mẫu tin (như mẫu tin về một người). Cột chứa dữ liệu (như First Name, Last Name, Address, và City).

Một ví dụ là bảng "Persons" sau:

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

LastName, FirstName, Address, và City là các cột của bảng. Các dòng chứa ba mẫu tin của 3 người.

Các truy vấn SQL

Với SQL, chúng ta có thể **truy vấn** một cơ sở dữ liệu và nhận được một **kết quả** trả về với dạng bảng.

Một truy vấn giống như sau:

```
SELECT LastName FROM Persons
```

Sẽ trả về một kết quả giống như sau:

LastName
Hansen
Svendson
Pettersen

Chú ý: Vài hệ cơ sở dữ liệu cần một dấu ";" ở cuối phát biểu SQL. Chúng ta không dùng dấu ";" trong bài viết này.

Thao tác dữ liệu SQL

SQL là một cú pháp để thực hiện các truy vấn. Nhưng ngôn ngữ SQL cũng chứa các cú pháp cập nhật các mẫu tin (record), chèn các mẫu tin mới và xóa các mẫu tin đang tồn tại.

Các lệnh truy vấn và cập nhật này thuộc dạng Ngôn ngữ Thao tác Dữ liệu (Data Manipulation Language - DML) một phần của SQL:

- SELECT – trích dữ liệu từ một cơ sở dữ liệu
- UPDATE – cập nhật dữ liệu trong một cơ sở dữ liệu
- DELETE – xóa dữ liệu từ một cơ sở dữ liệu
- INSERT – chèn dữ liệu mới vào trong một cơ sở dữ liệu

Định nghĩa dữ liệu SQL

Ngôn ngữ Định nghĩa Dữ liệu (Data Definition Language - DDL) một phần của SQL, cho phép tạo hay xóa các bảng cơ sở dữ liệu. Chúng ta cũng có thể định nghĩa các chỉ mục (các khóa - key), chỉ định liên kết giữa các bảng, và ràng buộc giữa các bảng cơ sở dữ liệu.

Các phát biểu DDL quan trọng nhất trong SQL là::

- CREATE TABLE – tạo một bảng cơ sở dữ liệu mới
- ALTER TABLE – thay đổi (alters) một bảng cơ sở dữ liệu

- DROP TABLE – xóa một bảng cơ sở dữ liệu
- CREATE INDEX – tạo một chỉ mục (khóa tìm kiếm)
- DROP INDEX – xóa một chỉ mục

SQL và ASP

SQL là một phần quan trọng của ASP (Active Server Pages), vì ADO (Active Data Object) được dùng trong ASP để truy xuất cơ sở dữ liệu, ADO dựa trên SQL để truy xuất dữ liệu.

Phát biểu SQL Select

Phát biểu **SELECT** chọn các cột dữ liệu từ một cơ sở dữ liệu.

Kết quả dạng bảng được lưu trong một bảng kết quả (gọi là tập kết quả - result set).

Phát biểu SELECT

Phát biểu **SELECT** chọn các cột dữ liệu từ một cơ sở dữ liệu.

Dùng phát biểu này để chọn (**SELECT**) thông tin từ (**FROM**) một bảng như sau:

```
SELECT column_name(s) FROM table_name
```

Ví dụ: Chọn các cột từ một bảng

Để chọn các cột có tên "LastName" và "FirstName", dùng một phát biểu **SELECT** như sau:

```
SELECT LastName,FirstName FROM Persons
```

Bảng "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Kết quả:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

Ví dụ: Chọn tất cả các cột

Để chọn tất cả các cột từ bảng "Person", dùng một ký hiệu * thay thế cho tên các cột như sau:

```
SELECT * FROM Persons
```

Kết quả:

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Bảng kết quả

Kết quả từ một truy vấn SQL được lưu trữ trong một tập kết quả. Tập kết quả có thể xem như một bảng kết quả. Đa số các

trình quản lý cơ sở dữ liệu cho phép duyệt tập kết quả với các hàm lập trình như: Move-To-First-Record, Get-Record-Content, Move-To-Next-Record.....

Mệnh đề SQL Where

Mệnh đề WHERE dùng để chỉ định một tiêu chuẩn (criteria) chọn.

Mệnh đề WHERE

Để chọn có điều kiện dữ liệu từ một bảng, một mệnh đề WHERE có thể thêm vào phát biểu SELECT với cú pháp sau:

```
SELECT column FROM table WHERE column condition value
```

Với mệnh đề WHERE, các điều kiện sau có thể được dùng:

Operator	Condition
=	Bằng
<>	Không bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
LIKE	Sẽ giải thích bên dưới

Chú ý: Vài phiên bản SQL toán tử <> có thể được viết thành !=

Ví dụ: Chọn người từ một công ty

Để chọn những người chỉ sống ở Sandnes, thêm mệnh đề WHERE vào phát biểu SELECT như sau:

```
SELECT * FROM Persons WHERE City='Sandnes'
```

Bảng "Persons":

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951
Svendson	Tove	Borgvn 23	Sandnes	1978
Svendson	Ståle	Kaivn 18	Sandnes	1980
Pettersen	Kari	Storgt 20	Stavanger	1960

Kết quả:

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951
Svendson	Tove	Borgvn 23	Sandnes	1978
Svendson	Ståle	Kaivn 18	Sandnes	1980

Dùng dấu nháy

Chú ý rằng chúng ta dùng dấu nháy đơn bao quanh các trị điều kiện trong các ví dụ. SQL dùng dấu nháy đơn bao quanh các trị văn bản. Phần lớn các hệ quản lý cơ sở dữ liệu cũng chấp nhận dấu nháy kép. Các trị số không được đóng trong dấu nháy.

Với các trị văn bản:

```
Viết đúng:
SELECT * FROM Persons WHERE FirstName='Tove'
Viết sai:
SELECT * FROM Persons WHERE FirstName=Tove
```

Với các trị số:

```
Viết đúng:
SELECT * FROM Persons WHERE Year>1965
Viết sai:
SELECT * FROM Persons WHERE Year>'1965'
```

Điều kiện LIKE

Điều kiện LIKE dùng chỉ định việc tìm một mẫu trong một cột.
Cú pháp:

```
SELECT column FROM table WHERE column LIKE pattern
```

Một dấu "%" có thể dùng như ký tự đại diện (wildcards) cả trước lẫn sau mẫu.

Ví dụ: Chọn trong bảng Persons với mẫu tên

Phát biểu SQL sẽ trả về những người có firstname bắt đầu với một ký tự 'O'.

```
SELECT * FROM Persons WHERE FirstName LIKE 'O%'
```

Phát biểu SQL sẽ trả về những người có firstname kết thúc với một ký tự 'a'.

```
SELECT * FROM Persons WHERE FirstName LIKE '%a'
```

Phát biểu SQL sẽ trả về những người có firstname chứa mẫu 'la'.

```
SELECT * FROM Persons WHERE FirstName LIKE '%la%'
```

Tất cả các ví dụ trên sẽ trả về kết quả sau:

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951

SQL And & Or

AND & OR

AND và OR kết nối hai hay nhiều điều kiện trong một mệnh đề WHERE.
Toán tử AND hiển thị một cột nếu TẤT CẢ các điều kiện liệt kê đều đúng.
Toán tử OR hiển thị một cột nếu MỘT TRONG các điều kiện liệt kê là đúng.

Bảng gốc (dùng trong các ví dụ)

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Svendson	Stephen	Kaivn 18	Sandnes

Ví dụ

Dùng AND để hiển thị người có firstname là "Tove", và lastname là "Svendson":

```
SELECT * FROM Persons
WHERE FirstName='Tove'
AND LastName='Svendson'
```

Kết quả:

LastName	FirstName	Address	City
Svendson	Tove	Borgvn 23	Sandnes

Ví dụ

Dùng OR để hiển thị người có firstname là "Tove", hoặc có lastname là "Svendson":

```
SELECT * FROM Persons
WHERE firstname='Tove'
OR lastname='Svendson'
```

Kết quả:

LastName	FirstName	Address	City
Svendson	Tove	Borgvn 23	Sandnes
Svendson	Stephen	Kaivn 18	Sandnes

Ví dụ

Bạn cũng có thể dùng phối hợp AND và OR (dùng dấu ngoặc đơn để bao các biểu thức phức tạp):

```
SELECT * FROM Persons WHERE
(FirstName='Tove' OR FirstName='Stephen')
AND LastName='Svendson'
```

Kết quả:

LastName	FirstName	Address	City
Svendson	Tove	Borgvn 23	Sandnes
Svendson	Stephen	Kaivn 18	Sandnes

SQL Between...And

BETWEEN ... AND

Toán tử BETWEEN ... AND chọn tất cả các trị trong khoảng giới hạn giữa hai trị. Các trị này có thể là các số, văn bản, hay ngày tháng.

```
SELECT column_name FROM table_name
WHERE column_name
BETWEEN value1 AND value2
```

Bảng gốc (dùng trong các ví dụ)

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Nordmann	Anna	Neset 18	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Ví dụ 1

Để hiển thị các tên theo thứ tự alphabet giữa hai tên (kể cả hai tên này) "Hansen" và "Pettersen", dùng SQL sau:

```
SELECT * FROM Persons WHERE LastName
BETWEEN 'Hansen' AND 'Pettersen'
```

Kết quả:

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Nordmann	Anna	Neset 18	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Ví dụ 2

Để hiển thị các tên ngoài các tên trong ví dụ trên, dùng toán tử NOT:

```
SELECT * FROM Persons WHERE LastName
NOT BETWEEN 'Hansen' AND 'Pettersen'
```

Kết quả:

LastName	FirstName	Address	City
Svendson	Tove	Borgvn 23	Sandnes

SQL Select Distinct

Từ khóa **DISTINCT** dùng trả về chỉ các trị khác biệt (distinct).

Từ khóa DISTINCT

Phát biểu SQL **SELECT** trả về thông tin từ các cột của bảng. Nhưng làm thế nào nếu chúng ta chỉ muốn chọn các kết quả không trùng nhau?

Với SQL, chúng ta chỉ cần thêm vào một từ khóa **DISTINCT** cho phát biểu **SELECT** với cú pháp sau:

```
SELECT DISTINCT column-name(s) FROM table-name
```

Ví dụ: Chọn tên công ty từ bảng Orders

Ví dụ: Bảng đặt hàng đơn giản:

Company	OrderNumber
Sega	3412
W3Schools	2312
Trio	4678
W3Schools	6798

Phát biểu SQL sau:

```
SELECT Company FROM Orders
```

Sẽ trả về kết quả:

Company
Sega
W3Schools
Trio
W3Schools

Chú ý rằng công ty W3Schools xuất hiện hai lần trong kết quả. Đôi lúc chúng ta không muốn điều này.

Ví dụ: Chọn tên công ty (không trùng tên) từ bảng Orders

Phát biểu SQL sau:

```
SELECT DISTINCT Company FROM Orders
```

Sẽ trả về kết quả:

Company
Sega
W3Schools
Trio

Bây giờ tên công ty W3Schools chỉ xuất hiện một lần trong kết quả.

SQL Order By

Từ khóa **ORDER BY** dùng sắp xếp kết quả thứ tự kết quả.

Sắp xếp các Dòng

Mệnh đề ORDER BY dùng sắp xếp các dòng.

Một số cách sắp xếp:

Company	OrderNumber
Sega	3412
ABC Shop	5678
W3Schools	2312
W3Schools	6798

Ví dụ

Để hiển thị tên công ty (Company) theo thứ tự alphabet:

```
SELECT Company, OrderNumber FROM Orders
ORDER BY Company
```



Kết quả:

Company	OrderNumber
ABC Shop	5678
Sega	3412
W3Schools	6798
W3Schools	2312

Ví dụ

Để hiển thị tên công ty (Company) theo thứ tự alphabet, nếu tên công ty giống nhau thì sắp xếp theo số thứ tự (OrderNumber):

```
SELECT Company, OrderNumber FROM Orders
ORDER BY Company, OrderNumber
```

Kết quả:

Company	OrderNumber
ABC Shop	5678
Sega	3412
W3Schools	2312
W3Schools	6798

Ví dụ

Để hiển thị tên công ty (Company) theo thứ tự alphabet đảo ngược (từ Z đến A):

```
SELECT Company, OrderNumber FROM Orders
ORDER BY Company DESC
```

Kết quả:

Company	OrderNumber
W3Schools	6798
W3Schools	2312
Sega	3412
ABC Shop	5678

SQL INSERT INTO

Chèn các dòng mới

Phát biểu INSERT INTO chèn các dòng mới vào trong một bảng:

```
INSERT INTO table_name
VALUES (value1, value2,...)
```


Bạn có thể chỉ định các cột bạn muốn chèn dữ liệu vào:

```
INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2,...)
```

Chèn một dòng mới

Bảng "Persons":

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger

Phát biểu SQL chèn vào bảng trên:

```
INSERT INTO Persons
VALUES ('Hetland', 'Camilla', 'Hagabakka 24', 'Sandnes')
```

Sẽ cho kết quả như sau:

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

Chèn dữ liệu vào trong các cột chỉ định

Bảng "Persons":

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

Phát biểu SQL chèn dữ liệu vào các cột chỉ định:

```
INSERT INTO Persons (LastName, Address)
VALUES ('Rasmussen', 'Storgt 67')
```

Sẽ cho kết quả như sau::

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes
Rasmussen		Storgt 67	

SQL Update

Update Rows

Phát biểu UPDATE cập nhật hoặc thay đổi các dòng:

```
UPDATE table_name SET column_name = new_value
WHERE column_name = some_value
```

Bảng Person:

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen		Storgt 67	

Cập nhật một cột trong một dòng

Chúng ta sẽ thêm một first name "Nina" đến người có lastname="Rasmussen":

```
UPDATE Person SET FirstName = 'Nina'
WHERE LastName = 'Rasmussen'
```

Cập nhật vài cột trong một dòng

Chúng ta sẽ thay đổi địa chỉ (Address) và thêm tên thành phố.

```
UPDATE Person
SET Address = 'Stien 12', City = 'Stavanger'
WHERE LastName = 'Rasmussen'
```

Kết quả

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

SQL Delete

Xóa các cột

Phát biểu DELETE dùng xóa một hay nhiều dòng trong một bảng.

```
DELETE FROM table_name WHERE column_name = some_value
```

Bảng "Person":

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

Xóa một dòng

"Nina Rasmussen" sẽ bị xóa:

```
DELETE FROM Person WHERE LastName = 'Rasmussen'
```

Kết quả

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger

Các hàm SQL Count

SQL có các hàm tạo sẵn để đếm các mẫu tin cơ sở dữ liệu.

Cú pháp hàm Count

Cú pháp của các hàm COUNT tạo sẵn như sau:

```
SELECT COUNT(column) FROM table
```

Hàm COUNT(*)

Hàm COUNT(*) trả về số hàng chọn được trong một phép chọn.
Với bảng "Persons" sau:

Name	Age
Hansen, Ola	34
Svendson, Tove	45
Pettersen, Kari	19

Ví dụ này trả về số hàng trong bảng:

```
SELECT COUNT(*) FROM Persons
```

Kết quả:

```
3
```

Ví dụ này trả về số người lớn hơn 20 tuổi:

```
SELECT COUNT(*) FROM Persons WHERE Age>20
```

Kết quả:

```
2
```

Hàm COUNT(column)

Hàm COUNT(column) trả về số hàng (ngoại trừ hàng có giá trị NULL) trong cột chỉ định.

Với bảng "Persons":

Name	Age
Hansen, Ola	34
Svendson, Tove	45
Pettersen, Kari	

Ví dụ này tìm số người có ghi tuổi tại field "Age" trong bảng "Persons":

```
SELECT COUNT(Age) FROM Persons
```

Kết quả:

```
2
```

Hàm COUNT(column) cũng dùng để tính số hàng không chứa trị. Chú ý kết quả sẽ nhỏ hơn số hàng trong bảng.

COUNT DISTINCT

Từ khóa DISTINCT với COUNT có thể dùng để đếm số kết quả khác nhau (không trùng nhau).

Cú pháp như sau:

```
SELECT DISTINCT COUNT(column(s)) FROM table
```

Với bảng "Orders":

Company	OrderNumber
Sega	3412
W3Schools	2312
Trio	4678
W3Schools	6798

Với phát biểu SQL sau:

```
SELECT COUNT(Company) FROM Orders
```

Sẽ trả về kết quả:

```
4
```

Với phát biểu SQL sau:

```
SELECT DISTINCT COUNT(Company) FROM Orders
```

Sẽ trả về kết quả:

```
3
```

Các hàm SQL

SQL có một số hàm tạo sẵn để đếm và tính toán.

Cú pháp dùng hàm

Cú pháp cho các hàm SQL tạo sẵn như sau::

```
SELECT function(column) FROM table
```

Bảng gốc (dùng trong các ví dụ)

Name	Age
Hansen, Ola	34
Svendson, Tove	45
Pettersen, Kari	19

Hàm AVG(column)

Hàm AVG trị trung bình của dữ liệu trong một cột có được nhờ phép chọn. Các trị NULL sẽ không được tính toán.

Ví dụ

Ví dụ này trả về tuổi trung bình của những người trong bảng "Persons":

```
SELECT AVG(Age) FROM Persons
```

Kết quả

```
32.67
```

Ví dụ

Ví dụ này trả về tuổi trung bình của những người có tuổi lớn hơn 20 tuổi:

```
SELECT AVG(Age) FROM Persons where Age>20
```

Kết quả

```
39.5
```

Hàm MAX(column)

Hàm MAX trả về trị lớn nhất trong một cột. Các trị NULL sẽ không được tính toán.

Ví dụ

```
SELECT MAX(Age) FROM Persons
```

Kết quả:

```
45
```

Hàm MIN(column)

Hàm MIN trả về trị lớn nhất trong một cột. Các trị NULL sẽ không được tính toán.

Ví dụ

```
SELECT MIN(Age) FROM Persons
```

Kết quả:

```
19
```

Chú ý: Các hàm MIN và MAX cũng có thể dùng trên các cột văn bản, để tìm trị lớn nhất và nhỏ nhất theo thứ tự alphabet.

Hàm SUM(column)

Hàm SUM tổng của một cột có được nhờ phép chọn. Các trị NULL sẽ không được tính toán.

Ví dụ

Ví dụ này trả về tổng số tuổi của những người trong bảng "Persons":

```
SELECT SUM(Age) FROM Persons
```

Kết quả:

98

Ví dụ

Ví dụ này trả về tổng số tuổi của những người lớn hơn 20 tuổi.

```
SELECT SUM(Age) FROM Persons where Age>20
```

Kết quả:

79

SQL Group By và SQL Having

Các hàm tổng (như SUM) thường kèm theo chức năng GROUP BY.

Từ khóa GROUP BY

Từ khóa GROUP BY được thêm vào SQL vì các hàm tổng (như SUM) trả về tổng của tất cả các trị trong cột mỗi khi chúng ta gọi đến.

Thiếu chức năng GROUP BY, không thể tìm tổng của mỗi nhóm trị riêng trong cột.

Cú pháp của GROUP BY như sau:

```
SELECT column,SUM(column) FROM table GROUP BY column
```

Ví dụ GROUP BY

Bảng "Sales":

Company	Amount
W3Schools	5500
IBM	4500
W3Schools	7100

Với SQL:

```
SELECT Company, SUM(Amount) FROM Sales
```

Trả về kết quả như sau:

Company	SUM(Amount)
W3Schools	17100
IBM	17100
W3Schools	17100

SQL trên không trả về tổng riêng biệt của từng công ty. Dùng mệnh đề GROUP BY như sau:

```
SELECT Company,SUM(Amount) FROM Sales
GROUP BY Company
```

Sẽ trả về kết quả đúng:

Company	SUM(Amount)
W3Schools	12600
IBM	4500

Từ khóa The HAVING

Từ khóa HAVING được thêm vào SQL vì từ khóa WHERE không thể dùng với các hàm tổng (như hàm SUM).

Thiếu từ khóa HAVING sẽ không thể kiểm tra các điều kiện dùng hàm tổng.

Cú pháp của HAVING như sau:

```
SELECT column,SUM(column) FROM table
GROUP BY column
HAVING SUM(column) condition value
```

Bảng "Sales":

Company	Amount
W3Schools	5500
IBM	4500
W3Schools	7100

Với SQL:

```
SELECT Company,SUM(Amount) FROM Sales  
GROUP BY Company HAVING SUM(Amount)>10000
```

Trả về kết quả

Company	SUM(Amount)
W3Schools	12600

Các bí danh (Alias) SQL

Với SQL, các bí danh (alias) có thể dùng thay các tên cột và các tên bảng.

Bí danh tên Cột

Cú pháp như sau:

```
SELECT column AS column_alias FROM table
```

Bí danh tên Bảng

Cú pháp như sau:

```
SELECT column FROM table AS table_alias
```

Ví dụ: Dùng bí danh tên Cột

Bảng "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Với SQL sau:

```
SELECT LastName AS Family, FirstName AS Name  
FROM Persons
```

Sẽ trả về kết quả sau:

Family	Name
Hansen	Ola
Svendson	Tove
Pettersen	Kari

Ví dụ: Dùng bí danh tên Bảng

Bảng "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes

Pettersen	Kari	Storgt 20	Stavanger
-----------	------	-----------	-----------

Với SQL sau:

```
SELECT LastName, FirstName
FROM Persons AS Employees
```

Sẽ trả về kết quả sau:

Bảng Employees:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

SQL Join

Joins and các Khóa (Key)

Đôi khi chúng ta chọn dữ liệu từ hai bảng để tạo kết quả. Chúng ta thực hiện một kết nối (join).

Các bảng trong cơ sở dữ liệu có thể liên hệ với các bảng khác thông qua các khóa. Một khóa chính (primary key) là một cột với các trị duy nhất cho mỗi hàng. Mục tiêu là ràng buộc dữ liệu, tham chiếu chéo các bảng, không cần lặp lại tất cả dữ liệu trong từng bảng.

Trong bảng "Employees" phía dưới, cột "ID" là khóa chính, nghĩa là cột này **không** có hai hàng cùng ID. ID dùng phân biệt hai người nếu cả hai có cùng tên.

Khi bạn xem bảng ví dụ phía dưới, chú ý rằng:

- Cột "ID" là khóa chính của bảng "Employees"
- Cột "ID" trong bảng "Orders" dùng để tham chiếu các tên trong bảng "Employees" không cần đưa các tên này vào bảng "Orders"

Employees:

ID	Name
01	Hansen, Ola
02	Svendson, Tove
03	Svendson, Stephen
04	Pettersen, Kari

Orders:

ID	Product
01	Printer
03	Table
03	Chair

Tham chiếu đến hai Bảng

Chúng ta có thể chọn dữ liệu từ hai bảng bằng cách tham chiếu đến hai bảng, như sau:

Ví dụ

Ai đã đăng ký một sản phẩm và đăng ký sản phẩm nào?

```
SELECT Employees.Name, Orders.Product
FROM Employees, Orders
WHERE Employees.ID = Orders.ID
```

Kết quả

Name	Product
Hansen, Ola	Printer
Svendson, Stephen	Table

Svendson, Stephen	Chair
-------------------	-------

Ví dụ

Ai đã đăng ký một máy in?

```
SELECT Employees.Name
FROM Employees, Orders
WHERE Employees.ID = Orders.ID
AND Orders.Product = 'Printer'
```

Kết quả

Name
Hansen, Ola

Dùng các Kết nối (Join)

HOẶC, chúng ta có thể chọn dữ liệu từ hai bảng với từ khóa JOIN, giống như sau:

Ví dụ INNER JOIN**Cú pháp**

```
SELECT field1, field2, field3
FROM first_table
INNER JOIN second_table
ON first_table.keyfield = second_table.foreign_keyfield
```

Ai đã đăng ký một sản phẩm và đăng ký sản phẩm nào?

```
SELECT Employees.Name, Orders.Product
FROM Employees
INNER JOIN Orders
ON Employees.ID = Orders.ID
```

INNER JOIN trả về tất cả các hàng từ hai bảng khi điều kiện được so trùng. Nếu các hàng trong bảng Employees không so trùng trong bảng Orders, hàng đó sẽ **không** được liệt kê ra.

Kết quả

Name	Product
Hansen, Ola	Printer
Svendson, Stephen	Table
Svendson, Stephen	Chair

Ví dụ LEFT JOIN**Cú pháp**

```
SELECT field1, field2, field3
FROM first_table
LEFT JOIN second_table
ON first_table.keyfield = second_table.foreign_keyfield
```

Liệt kê tất cả nhân viên, và các đăng ký mua của họ nếu có.

```
SELECT Employees.Name, Orders.Product
FROM Employees
LEFT JOIN Orders
ON Employees.ID = Orders.ID
```

LEFT JOIN trả về tất cả các hàng từ bảng thứ nhất (Employees), cho dù nó không được so trùng trong bảng thứ hai (Orders). Nếu các hàng trong bảng Employees không so trùng trong bảng Orders, những hàng này **cũng được** liệt kê.

Kết quả

Name	Product
Hansen, Ola	Printer
Svendson, Tove	
Svendson, Stephen	Table
Svendson, Stephen	Chair
Pettersen, Kari	

Ví dụ RIGHT JOIN

Cú pháp

```
SELECT field1, field2, field3
FROM first_table
RIGHT JOIN second_table
ON first_table.keyfield = second_table.foreign_keyfield
```

Liệt kê tất cả nhân viên, và các đăng ký mua của họ nếu có.

```
SELECT Employees.Name, Orders.Product
FROM Employees
RIGHT JOIN Orders
ON Employees.ID = Orders.ID
```

RIGHT JOIN trả về tất cả các hàng từ bảng thứ hai (Orders), cho dù nó không được so trùng trong bảng thứ nhất (Employees). Nếu có bất kỳ hàng nào trong bảng Orders không được so trùng trong bảng Employees, các hàng này **cũng được** liệt kê.

Kết quả

Name	Product
Hansen, Ola	Printer
Svendson, Stephen	Table
Svendson, Stephen	Chair

Ví dụ

Ai đăng ký một máy in?

```
SELECT Employees.Name
FROM Employees
INNER JOIN Orders
ON Employees.ID = Orders.ID
WHERE Orders.Product = 'Printer'
```

Kết quả

Name
Hansen, Ola

SQL Tạo Cơ sở dữ liệu và Bảng

Tạo một Cơ sở dữ liệu

Để tạo một cơ sở dữ liệu:

```
CREATE DATABASE database_name
```

Tạo một bảng

Để tạo một bảng trong một cơ sở dữ liệu:

```
CREATE TABLE table_name
(
  column_name1 data_type,
  column_name2 data_type,
  .....
)
```

Ví dụ

Ví dụ này minh họa các bạn tạo một bảng tên "Person", với bốn cột tên: "LastName", "FirstName", "Address", và "Age":

```
CREATE TABLE Person
(
  LastName varchar,
  FirstName varchar,
  Address varchar,
  Age int
)
```

Ví dụ này minh họa cách bạn chỉ định kích thước tối đa của vài cột:

```
CREATE TABLE Person
(
  LastName varchar(30),
  FirstName varchar,
  Address varchar,
  Age int(3)
)
```

Kiểu dữ liệu được chỉ định là kiểu dữ liệu chứa trong cột. Bảng dưới chứa các kiểu dữ liệu thường gặp nhất trong SQL:

Kiểu dữ liệu	Mô tả
integer(size) int(size) smallint(size) tinyint(size)	Chỉ chứa số nguyên. Số ký tự số tối đa được chỉ định trong dấu ngoặc đơn
decimal(size, d) numeric(size,d)	Chứa số với phần số. Số ký tự số tối đa được chỉ định trong "size". Số ký tự số tối đa bên phải (phần phân số) được chỉ định trong "d"
char(size)	Chứa chuỗi có kích thước cố định (có thể chứa ký tự chữ, số, và các ký tự đặc biệt). Kích thước cố định được chỉ định trong dấu ngoặc đơn
varchar(size)	Chứa một chuỗi có chiều dài thay đổi (có thể chứa ký tự chữ, số, và các ký tự đặc biệt). Kích thước tối đa được chỉ định trong dấu ngoặc đơn
date(yyymmdd)	Chứa một ngày

Tạo Chỉ mục (Index)

Chỉ mục được tạo ra trên một bảng có sẵn để định vị thêm nhanh và hiệu quả các hàng. Có thể tạo một chỉ mục trên một hoặc nhiều cột của một bảng, với một chỉ mục cho một tên. Người dùng không nhìn thấy các chỉ mục, chúng chỉ dùng để tăng tốc độ truy vấn.

Chú ý: Cập nhật một bảng chứa chỉ mục cần nhiều thời gian hơn cập nhật một bảng không chứa chỉ mục, vì chỉ mục cũng cần cập nhật. Tuy nhiên, ý tưởng tốt là tạo chỉ mục chỉ trên các cột thường tìm kiếm nhất.

Một Chỉ mục duy nhất

Tạo một chỉ mục duy nhất trên một bảng. một chỉ mục duy nhất nghĩa là không thể có hai hàng có cùng một trị chỉ mục.

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

"column_name" chỉ định cột bạn muốn chỉ mục.

Một Chỉ mục đơn giản

Tạo một chỉ mục đơn giản trên một bảng. Khi từ khóa UNIQUE không có, các trị trùng sẽ được cho phép.

```
CREATE INDEX index_name
ON table_name (column_name)
```

"column_name" chỉ định cột bạn muốn chỉ mục.

Ví dụ

Ví dụ này tạo một chỉ mục đơn giản, có tên "PersonIndex", trên field LastName của bảng Person:

```
CREATE INDEX PersonIndex
ON Person (LastName)
```

Nếu bạn muốn chỉ mục các trị trong một cột theo thứ tự **giảm** (descending), bạn có thể thêm từ **DESC** sau tên cột:

```
CREATE INDEX PersonIndex
ON Person (LastName DESC)
```

Nếu bạn muốn chỉ mục nhiều hơn một cột bạn có thể liệt kê các tên cột trong dấu ngoặc đơn, tách chúng bằng dấu phẩy:

```
CREATE INDEX PersonIndex
ON Person (LastName, FirstName)
```

Xóa chỉ mục

Bạn có thể xóa một chỉ mục có trong một bảng với phát biểu DROP.

```
DROP INDEX table_name.index_name
```

Xóa một cơ sở dữ liệu hoặc bảng

Để xóa một cơ sở dữ liệu:

```
DROP DATABASE database_name
```

Để xóa một bảng:

```
DROP TABLE table_name
```

Để xóa toàn bộ dữ liệu trong bảng mà không xóa bảng:

```
DELETE TABLE table_name
```

SQL Alter Table

Alter Table

Phát biểu ALTER TABLE dùng để thêm hay loại bỏ các cột trong một bảng cho trước.

```
ALTER TABLE table_name
ADD column_name datatype
ALTER TABLE table_name
DROP column_name
```

Person:

LastName	FirstName	Address
Pettersen	Kari	Storgt 20

Ví dụ

Để thêm một cột tên "City" vào bảng "Person":

```
ALTER TABLE Person ADD City varchar(30)
```

Kết quả:

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	

Ví dụ

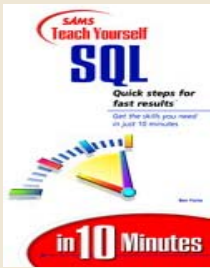
Để loại cột "Address" khỏi bảng "Person":

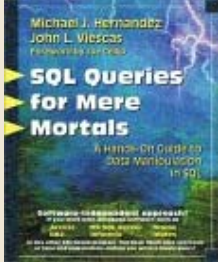
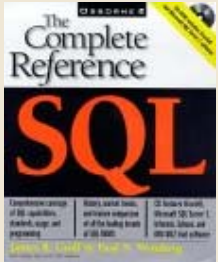

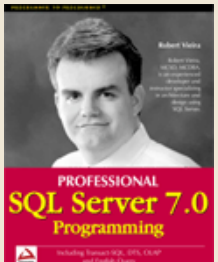
```
ALTER TABLE Person DROP Address
```

Kết quả:

LastName	FirstName	City
Pettersen	Kari	

Sách SQL

Sách	Mô tả
	<p>Teach Yourself SQL in 10 Minutes</p> <p>September 1999</p> <p>Loại sách tutorial, tổ chức thành chuỗi các bài học-10 phút đơn giản.</p>

	<p><u>SQL Queries for Mere Mortals</u></p> <p>August 2000</p> <p>Giúp người dùng mới học cơ bản về các truy vấn SQL, và cung cấp một hướng dẫn tham chiếu cần thiết với người dùng có trình độ cao hơn.</p>
	<p><u>SQL: The Complete Reference</u></p> <p>October 1999</p> <p>Cung cấp tất cả những gì bạn cần biết về SQL.</p>
	<p><u>Professional SQL Server 2000 Programming</u></p> <p>December 2000</p> <p>Cung cấp một hướng dẫn toàn diện để lập trình với SQL Server 2000.</p>
	<p><u>Professional SQL Server 7.0 Programming</u></p> <p>September 1999</p> <p>Cung cấp tổng quan về tất cả các bộ phận của SQL Server.</p>