**Links:** [**https://freetuts.net/lenh-inner-join-trong-sql-server-2801.html**](https://freetuts.net/lenh-inner-join-trong-sql-server-2801.html)

## 1. Primary key là gì?

Khóa chính là **tập hợp một hoặc nhiều column giúp phân biệt các record trong một table,** đây là thông tin rất quan trọng bởi nếu thiếu nó thì lược đồ CSDL của bạn chỉ là đồ bỏ đi.

Ví dụ bảng SINHVIEN thì thường ta sẽ có column MASINHVIEN dùng để nhận diện các sinh viên, tên sinh viên có thể bị trùng nhưng mã sinh viên thì không thể trùng.

Đặc điểm của khóa chính như sau:

*Bài viết này được đăng tại [free tuts .net]*

* Có thể thiết lập khóa chính bằng một hoặc nhiều column, trong thực tế thì nên 1 column
* Khóa chính không được NULL, và là duy nhất (unique)
* Khóa chính nếu là kiểu số nguyên thì nên thiết lập [tăng tự động](https://freetuts.net/thiet-lap-id-tang-tu-dong-voi-identity-trong-sql-server-1583.html) sẽ giúp tối ưu database

## 2. Cách tạo primary key

ADVERTISEMENT

Chúng ta có hai cách tạo primary key, thứ nhất là tạo trực tiếp ở lệnh [create table](https://freetuts.net/lenh-create-table-trong-sql-server-1581.html) và thứ hai là sử dụng lệnh alter table.

### Tạo trực tiếp ở lệnh create table

Cách này có hai cú pháp như sau:

**Cú pháp 1**

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE TABLE table\_name (      pk\_column data\_type PRIMARY KEY,      ...  ); |

**Cú pháp 2**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE table\_name (      pk\_column\_1 data\_type,      pk\_column\_2 data type,      ...      PRIMARY KEY (pk\_column\_1, pk\_column\_2)  ); |

Với **cú pháp 1** thì chỉ dùng được ở trường hợp khóa chính chỉ có một column, còn **cú pháp 2** thì dùng cho trường hợp nhiều column.

**Ví dụ 1**: Tạo bảng **activities** thuộc schema **sales**, vì khóa chính chỉ có 1 column nên dùng cú pháp 1.

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE sales.activities (      activity\_id INT PRIMARY KEY IDENTITY,      activity\_name VARCHAR (255) NOT NULL,      activity\_date DATE NOT NULL  ); |

**Ví dụ 2**: Tạo bảng **participants**thuộc schema **sales**, vì khóa chính có 2 column nên dùng cú pháp 2.

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE sales.participants(      activity\_id int,      customer\_id int,      PRIMARY KEY(activity\_id, customer\_id)  ); |

### Tạo bằng lệnh Alter Table

Lệnh ALTER TABLE có thể dùng để thay đổi cấu trúc của table nên ta có thể dùng nó để xác định khóa chinh **sau khi tạo bảng**.

**Ví dụ**: Tạo bảng **events**thuộc schema **sales**. Chúng ta sẽ có hai bước như sau.

Bước 1: Tạo bảng

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE sales.events(      event\_id INT,      event\_name VARCHAR(255),      start\_date DATE NOT NULL,      duration DEC(5,2)  ); |

Bước 2: Sử dụng lệnh alter table để tạo khóa chính

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE sales.events  ADD PRIMARY KEY(event\_id); |

Trên là hai cách tạo khóa chính (primary key) trong SQL Server. Bài này mình dừng ở đây, bài tiếp theo mình sẽ nói đến cách tạo khóa ngoại cho table.

**1. Khóa ngoại Foreign Key là gì?**

Khóa ngoại hay còn gọi là foreign key, đây là mối liên kết giữa hai bảng với nhau tạo thành một lược đồ cơ sở dữ liệu quan hệ.

Giả sử ta có hai bảng vendor\_groups và vendors như sau:

**vendors**

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE procurement.vendors (          vendor\_id INT IDENTITY PRIMARY KEY,          vendor\_name VARCHAR(100) NOT NULL,          group\_id INT NOT NULL,  ); |

**vendor\_group**

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE TABLE procurement.vendor\_groups (      group\_id INT IDENTITY PRIMARY KEY,      group\_name VARCHAR (100) NOT NULL  ); |

Trong đó:

*Bài viết này được đăng tại [free tuts .net]*

* Bảng vendor\_group sẽ lưu trữ danh sách các nhóm nhà cung cấp, ví dụ nhà cung cấp hạng A, hạng B, ...
* Bảng vendors sẽ lưu trữ danh sách nhà cung cấp

ADVERTISEMENT

**Mối liên hệ**: Mỗi nhà cung cấp sẽ thuộc một nhóm nào đó, và một nhóm có thể có nhiều nhà cung cấp, đây là quy luật bình thường trong một hệ thống bán hàng.

Để thẻ hiện mối liên hệ này thì trong bảng **vendors** sẽ có một column group\_id trỏ đến khóa chính của bảng **vendor\_group**, ta gọi đây là foreign key.

**Một số lưu ý của khóa ngoại**:

* Bảng A có khóa ngoại trỏ đến bảng B thì ta gọi A là bảng Cha, còn B là bảng con ( *mình tự đặt cho dễ nhớ ;)*)
* Giá trị của khóa ngoại của bảng con phải tồn tại trong các giá trị khóa chính của bảng cha, đây ta gọi là ràng buộc toàn vẹn.

**2. Cách tạo Freign Key trong SQL Server**

Tương tự như khóa chính, chúng ta có hai cách để tạo khóa ngoại. Thứ nhất là tạo ngay lệnh create table và thứ hai là sử dụng lệnh alter table.

**Tạo ngay lệnh Create Table**

Ta sử dụng từ khóa CONSTRAINT ngay ở phía cuối danh sách column.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE procurement.vendors (          vendor\_id INT IDENTITY PRIMARY KEY,          vendor\_name VARCHAR(100) NOT NULL,          group\_id INT NOT NULL,          CONSTRAINT fk\_group FOREIGN KEY (group\_id)          REFERENCES procurement.vendor\_groups(group\_id)  ); |

**Chú ý**:

* fk\_group là tên của khóa ngoại, ta nên đặt tên để sau này dễ dàng quản lý

Trường hợp có nhiều column làm khóa ngoại thì ta sử dụng cú pháp sau:

|  |  |
| --- | --- |
| 1  2  3 | CONSTRAINT fk\_constraint\_name  FOREIGN KEY (column\_1, column2,...)  REFERENCES parent\_table\_name(column1,column2,..) |

# Ràng buộc Check trong SQL

Bài này bạn sẽ được học cách tạo ràng buộc giá trị trong SQL bằng lệnh CHECK, cách viết nó thông qua các hệ quản trị CSDL như SQL Server, MySQL và Oracle.



Ràng buộc giá trị CHECK rất hữu ích trong việc đảm bảo tính toàn vẹn của dữ liệu. Ví dụ bạn muốn điểm số của học sinh phải nằm trong khoảng [0-10], hoặc giới tính là Năm hoặc Nữ, và nhiều trường hợp tương tự thì có thể sử dụng ràng buộc CHECK.

**Mục lục**

* [1. Tạo ràng buộc CHECK trong lệnh CREATE TABLE](https://freetuts.net/rang-buoc-check-trong-sql-1597.html#goto-h2-0)
* [2. Tạo ràng buộc CHECK trong lệnh ALTER TABLE](https://freetuts.net/rang-buoc-check-trong-sql-1597.html#goto-h2-1)
* [3. Xóa ràng buộc CHECK ra khỏi table](https://freetuts.net/rang-buoc-check-trong-sql-1597.html#goto-h2-2)

## 1. Tạo ràng buộc CHECK trong lệnh CREATE TABLE

Trong ví dụ dưới đây sẽ tạo ràng buộc CHECK ở column age, vì là bảng lưu trữ nhân viên nên bắt buộc chỉ tuyển những người trên 18 tuổi.

**MySQL**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE Persons (      ID int NOT NULL,      LastName varchar(255) NOT NULL,      FirstName varchar(255),      Age int,      CHECK (Age>=18)  ); |

**SQL Server / Oracle / MS Access**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE Persons (      ID int NOT NULL,      LastName varchar(255) NOT NULL,      FirstName varchar(255),      Age int CHECK (Age>=18)  ); |

Trường hợp bạn có nhiều column sử dụng CHECK thì nên đặt nó ở cuối các column trong lệnh tạo bảng, dùng toán tử AND để nối các ràng buộc lại. Cách này rất hay vì ta có thể đặt tên cho CHECK.

*Bài viết này được đăng tại [free tuts .net]*

**MySQL / SQL Server / Oracle / MS Access**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | CREATE TABLE Persons (      ID int NOT NULL,      LastName varchar(255) NOT NULL,      FirstName varchar(255),      Age int,      City varchar(255),      CONSTRAINT CHK\_Person CHECK (Age>=18 AND City='Sandnes')  ); |

## 2. Tạo ràng buộc CHECK trong lệnh ALTER TABLE

ADVERTISEMENT

Lệnh Alter Table được sử dụng rất nhiều trong việc thay đổi cấu trúc của table, và nó cũng được dùng để tạo CHECK sau khi bạn đã tạo xong table.

**MySQL / SQL Server / Oracle / MS Access**

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Persons<br>  ADD CHECK (Age>=18); |

Tương tự như  Primary Key và Foreign Key, bạn nên đặt tên cho CHECK để sau này có thể xóa nó dễ dàng hơn.

**MySQL / SQL Server / Oracle / MS Access**

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Persons<br>  ADD CONSTRAINT CHK\_PersonAge CHECK (Age>=18 AND City='Sandnes'); |

## 3. Xóa ràng buộc CHECK ra khỏi table

Nếu bạn đã tạo CHECK rồi nhưng sau đó không muốn có ràng buộc đó nữa thì hãy sử dụng lệnh ALTER TABLE kết hợp với từ khóa DROP nhé.

**SQL Server / Oracle / MS Access**

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Persons  DROP CONSTRAINT CHK\_PersonAge; |

**MySQL**

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Persons  DROP CHECK CHK\_PersonAge; |

Vậy là xong, rất đơn giản và dễ hiểu.

# Gộp dữ liệu với UNION và UNION ALL trong SQL

Có một số trường hợp bắt buộc bạn phải chạy hai câu truy vấn nên kết quả sẽ trả về hai bảng tạm, lúc này nếu bạn muốn gộp hai bảng tạm đó lại thành một bảng thì có thể dùng toán tử UNION.



**Mục lục**

* [1. Gộp kết quả với UNION](https://freetuts.net/gop-du-lieu-voi-union-va-union-all-trong-sql-1474.html#goto-h2-0)
* [2. Gộp kết quả với UNION ALL](https://freetuts.net/gop-du-lieu-voi-union-va-union-all-trong-sql-1474.html#goto-h2-1)
* [3. Ví dụ với UNION và UNION ALL](https://freetuts.net/gop-du-lieu-voi-union-va-union-all-trong-sql-1474.html#goto-h2-2)
  + [UNION](https://freetuts.net/gop-du-lieu-voi-union-va-union-all-trong-sql-1474.html#goto-h3-0)
  + [UNION ALL](https://freetuts.net/gop-du-lieu-voi-union-va-union-all-trong-sql-1474.html#goto-h3-1)

## 1. Gộp kết quả với UNION

Trước tiên bạn xem cú pháp của lệnh này đã nhé.

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT statement1    UNION    SELECT statement2 |

UNION có tác dụng két quả hai lệnh SELECT và loại bỏ đi những dòng bị trùng lặp chỉ để lại một dòng duy nhất, tuy nhiên không phải lệnh nào cũng gộp được mà phải thoải điều kiện sau:

* Tên của các column phải giống nhau
* Thứ tự các column phải giống nhau
* Tổng các column phải bằng nhau

**Ví dụ**:

*Bài viết này được đăng tại [free tuts .net]*

* Câu SELECT thứ nhất trả về 3 column đó là: ID, NAME, ADDRESS
* Thì câu SELECT thứ ai cũng phải trả về 3 column như trên

## 2. Gộp kết quả với UNION ALL

ADVERTISEMENT

Có cú pháp và tác dụng tương tự như UNION, chỉ có điều là lệnh UNION ALL sẽ giữ lại tất cả dữ liệu kể cả những dòng bị trùng lặp.

Cú pháp như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT statement1    UNION ALL    SELECT statement2 |

## 3. Ví dụ với UNION và UNION ALL

Giả sử mình có bảng CSDL như sau:

**Bảng EMPLOYEE1**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ID  NAME        AGE     ADDRESS  1   Cường       32      ĐăkLăk  2   Kính        25      Gia Lai  3   Chính       23      Kontum  4   Quyền       25      ĐăkLăk  5   Kim         27      Gia Lai  6   Tình        22      Bình Phước |

**Bảng EMPLOYEE2**

|  |  |
| --- | --- |
| 1  2  3  4  5 | ID  EMPLOYEE\_ID     TASK  1   Cường           Tiếp Tân  2   Kính            Dọn Dẹp  3   Chính           Đặt Phòng  4   Quyền           Nấu Ăn |

Bây giờ mình thực hiện lấy danh sách tất cả nhân viên cua cả hai table trên và gộp vào một kết quả duy nhất, kết quả chỉ trả về NAME.

### UNION

**CODE**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT NAME  FROM EMPLOYEE1    UNION    SELECT NAME  FROM EMPLOYEE2 |

**Kế quả sẽ trả về**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Cường  Kính  Chính  Quyền  Kim  Tình |

Kết quả trả vê đã lọc đi dữ liệu bị trùng.

### UNION ALL

**Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT NAME  FROM EMPLOYEE1    UNION ALL    SELECT NAME  FROM EMPLOYEE2 |

**Kết quả**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Cường  Kính  Chính  Quyền  Kim  Tình  Cường  Kính  Chính  Quyền |

Kết quả trả về vẫn giữ nguyên như cũ.

Như vậy lệnh UNION và UNION đều có công dụng là gộp kết quả từ hai câu lệnh select, nhưng có điểm khác biệt là UNION sẽ loại bỏ dữ liệu trùng lặp còn UNION ALL thì giữ lại tất cả.

Trường hợp tên row không giống nhau thì bạn sử dụng từ khóa AS để đổi tên cho trùng khớp nhé.

**Lệnh Group by**

**Giới thiệu Group By trong MySQL**

Mệnh đề GROUP BY sẽ gom nhóm dữ liệu và chỉ trả về một record cho một nhóm duy nhất, vì vậy nó giúp giảm bớt dữ liệu không cần thiết trong kết quả trả về.

Lệnh này thường kết hợp với các hàm như: SUM, AVG, MAX, MIN và COUNT để đếm hoặc lấy thông tin cần thiết trả về cho từng nhóm.

*Bài viết này được đăng tại [free tuts .net]*

ADVERTISEMENT

**Cú pháp như sau**:

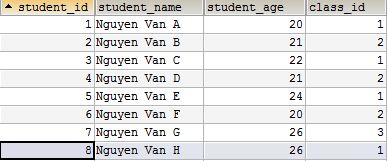
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      c1, c2,..., cn  FROM      table  WHERE      where\_conditions  GROUP BY c1 , c2,...,cn; |

**\* Lưu ý**:

* Lệnh Group By có khoảng trắng, và nó phải nằm sau lệnh WHERE.
* Những field có trong group by bắt buộc phải có trong Select.
* Thứ tự thực thi câu lệnh sẽ là: From -> Where -> Select -> Group By

**2. Cách sư dụng Group By trong MySQL**

Cho bảng Students có dữ liệu như sau:



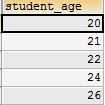
**Bài toán 1**: Hãy lấy danh sách tuổi và số sinh viên có độ tuổi đó.

Với bài toán này ta phải hiển thị hai thông tin: *Tuổi | Số sinh viên có độ tuổi này*. Ví dụ tuổi 20 thì sẽ có 2 sinh viên, tuổi 21 có 2 sinh viên, tuổi 22 có một sinh viên.

Vì bài toán yêu cầu lấy danh sách theo tuổi nên ta sẽ gom nhóm theo tuổi. Hãy thử câu lệnh sau:

|  |  |
| --- | --- |
| 1  2  3 | SELECT student\_age  FROM students      GROUP BY student\_age |

**Kết quả**:

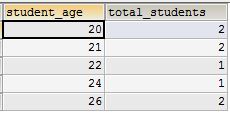


Đây không phải là kết quả mong muốn, bởi chưa có thông tin số sinh viên thuộc độ tuổi đó.

Vì kết quả nó trả về 1 record, chính là đại diện cho nhóm nên ta có thể sử dụng hàm Count để đếm số record trong nhóm. Đây cũng chính là số sinh viên thuộc độ tuổi đó. Bạn có thể đếm một field bất kì nhé.

|  |  |
| --- | --- |
| 1  2  3 | SELECT student\_age, COUNT(student\_name) AS total\_students  FROM students      GROUP BY student\_age |

**Kết quả**:



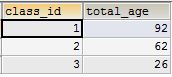
\***Ghi nhớ**: Các hàm Count, Max, Sum sẽ có tác dụng đến các records trong nhóm ở câu lệnh group by.

**Bài toán 2**: Đếm tổng số tuổi của tất cả sinh viên và hiển thị theo lớp (class\_id).

Bài này ta sử dụng hàm SUM và gom nhóm theo class\_id là được.

|  |  |
| --- | --- |
| 1  2  3 | SELECT  class\_id, SUM(student\_age) AS total\_age  FROM students  GROUP BY class\_id |

**Kết quả**:



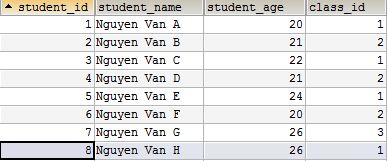
**3. Dùng Having để thêm điều kiện cho Group By**

Bạn có thể thêm điều kiện lọc ở lệnh Group By bằng cách sử dụng mệnh đề Having. Lúc này thứ tự chạy sẽ như sau:

* **From -> Where -> Select -> Group By -> Having -> Order By -> Limit**

Quay lại dữ liệu ở trên mình sẽ ra một bài toán như sau: Đếm tổng số sinh viên theo từng lớp, và chỉ trả về lớp nào có từ 2 sinh viên trở đi.

Mình sẽ hiển thị lại bảng dữ liệu cho bạn dễ hình dung.



Bạn hãy nhìn thứ tự chạy ở trên để suy ngẫm nhé. **Kết quả mong muốn phải được lấy từ lệnh Group By nên điều kiện này không thể đặt ở Where được (where chạy trước group by). Vì vậy ta phải đặt trong Having**.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT  class\_id, COUNT(student\_age) AS total\_student  FROM students  GROUP BY class\_id  HAVING total\_student >= 2 |

**Kết quả**:

having theo lop JPG

Bạn hãy thử đặt điều kiện ở lệnh Where xem thế nào nhé.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT  class\_id, COUNT(student\_age) AS total\_student  FROM students  WHERE total\_student >= 2  GROUP BY class\_id |

Chạy lên sẽ nhận được lỗi không tồn tại field total\_student.

|  |  |
| --- | --- |
| 1  2  3  4  5 | Error Code : 1054  Unknown column 'total\_student' in 'where clause'  Execution Time : 00:00:00:000  Transfer Time  : 00:00:00:000  Total Time     : 00:00:00:000 |

Trên là cách sử dụng lệnh Group By và Group By có Having trong MySQL. Hy vọng đây qua bài này bạn sẽ hiểu hơn về lệnh Group By trong MySQL để có cách sử dụng chính xác hơn.

1. **Lệnh Having trong SQL Server dùng để làm gì?**

Như bạn biết, để thêm điều kiện lọc thì ta thường đặt trong lệnh WHERE. Tuy nhiên theo quy tắc hoạt động của câu SQL sẽ chạy ở Group By trước rồi mới đến Where. Vì vậy trường hợp bạn muốn thêm điều kiện trong lúc Group By thì có giải pháp nào không?

Thật may là ngôn ngữ T-SQL khá thông minh, nó cho phép bạn thêm điều kiện ở Group By bằng lệnh **HAVING**.

**Cú pháp như sau**:

*Bài viết này được đăng tại [free tuts .net]*

SELECT

select\_list

FROM

table\_name

GROUP BY

group\_list

HAVING

conditions;

ADVERTISEMENT

Trong cú pháp này thì conditions chính là điều kiện lọc lúc group by.

Ví **dụ:** bạn muốn tìm tổng số học sinh theo thành phố, nhưng không đếm những học sinh có năm sinh là 2000.

SELECT city, count(student\_id) as total

FROM Students

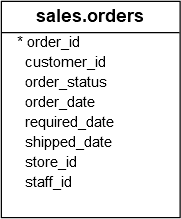
GROUP BY city

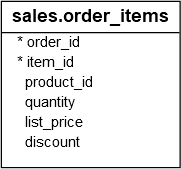
HAVING year\_bon <> 2000

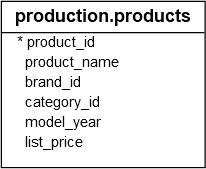
Bây giờ ta sẽ làm ví dụ thực tế hơn nhé.

**2. Ví dụ Having trong SQL Server**

Trước tiên hãy giúp mình tạp một số table như sau:







**Trong đó**:

* Table products dùng lưu trữ sản phẩm
* Table orders dùng lưu trữ đơn hàng
* Table order\_items dùng lưu trữ danh sách sản phẩm của mỗi đơn hàng. Như vậy nó sẽ có mối quan hệ (n : 1) với cả hai bảng còn lại.

**Having với hàm Count**

Câu lệnh dưới đây sử dụng mệnh đề HAVING để tìm các khách hàng đã đặt ít nhất hai đơn hàng mỗi năm:

SELECT

customer\_id,

YEAR (order\_date),

COUNT (order\_id) order\_count

FROM

sales.orders

GROUP BY

customer\_id,

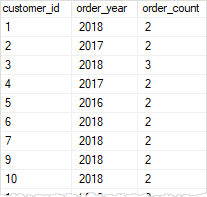
YEAR (order\_date)

HAVING

COUNT (order\_id) >= 2

ORDER BY

customer\_id;



**Having với hàm SUM**

Câu lệnh dứoi đây tìm thấy giá niêm yết tối đa và tối thiểu trong mỗi loại sản phẩm. Sau đó, nó lọc ra sản phẩm có giá niêm yết tối đa lớn hơn 4.000 hoặc giá niêm yết tối thiểu dưới 500:

SELECT

category\_id,

MAX (list\_price) max\_list\_price,

MIN (list\_price) min\_list\_price

FROM

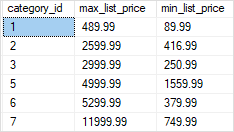
production.products

GROUP BY

category\_id

HAVING

MAX (list\_price) > 4000 OR MIN (list\_price) < 500;



**Having với hàm AVG**

Câu lệnh dưới đây tìm các chuyên mục sản phẩm có giá niêm yết trung bình (*tất cả sản phẩm của chuyên mục đó*) từ 500 đến 1.000:

SELECT

category\_id,

AVG (list\_price) avg\_list\_price

FROM

production.products

GROUP BY

category\_id

HAVING

AVG (list\_price) BETWEEN 500 AND 1000;

SQL Server HAVING AVG example png

Trên là cách sử dụng Having trong SQL Server. Nếu bạn thường xuyên sử dụng Group By thì không thể thiếu câu lệnh này.

Và một lưu ý mình muốn nhắc nhở, đó là Having sẽ hoạt động trước lênh Where nhé. Vì vậy bạn không thể sử dụng Where để lọc trong lúc gom nhóm bằng Group By.

**1. Lệnh JOIN trong SQLite là gì?**

Lệnh JOIN trong SQLite được sử dụng để kết hợp các bản ghi từ hai hoặc nhiều bảng trong cơ sở dữ liệu. Lệnh JOIN sẽ kết hợp các trường từ hai bảng bằng cách sử dụng các giá trị chung cho mỗi bảng.

SQLite định nghĩa ba loại join chính như sau:

* CROSS JOIN
* INNER JOIN
* OUTER JOIN

Giả sử chúng ta có 2 bảng COMPANY và DEPARTMENT. Chúng ta đã sử dụng các câu lệnh INSERT để điền dữ liệu vào bảng COMPANY. Vì vậy, giả sử chúng ta đã có các dòng dữ liệu sau trong bảng COMPANY

*Bài viết này được đăng tại [free tuts .net]*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | ID          NAME        AGE         ADDRESS     SALARY  ----------  ----------  ----------  ----------  ----------  1           Paul        32          California  20000.0  2           Allen       25          Texas       15000.0  3           Teddy       23          Norway      20000.0  4           Mark        25          Rich-Mond   65000.0  5           David       27          Texas       85000.0  6           Kim         22          South-Hall  45000.0  7           James       24          Houston     10000.0 |

ADVERTISEMENT

Bảng DEPARTMENT được định nghĩa như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE DEPARTMENT(     ID INT PRIMARY KEY      NOT NULL,     DEPT           CHAR(50) NOT NULL,     EMP\_ID         INT      NOT NULL  ); |

Dưới đây là danh sách các câu lệnh INSERT để điền dữ liệu vào bảng DEPARTMENT

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | INSERT INTO DEPARTMENT (ID, DEPT, EMP\_ID)  VALUES (1, 'IT Billing', 1 );    INSERT INTO DEPARTMENT (ID, DEPT, EMP\_ID)  VALUES (2, 'Engineering', 2 );    INSERT INTO DEPARTMENT (ID, DEPT, EMP\_ID)  VALUES (3, 'Finance', 7 ); |

Cuối cùng bảng DEPARTMENT có các dòng dữ liệu như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5 | ID          DEPT        EMP\_ID  ----------  ----------  ----------  1           IT Billing  1  2           Engineering 2  3           Finance     7 |

**2. CROSS JOIN trong SQLite**

CROSS JOIN kết nối mọi hàng của bảng đầu tiên với mỗi hàng của bảng thứ hai. Nếu các bảng đầu vào có hàng x và y, tương ứng, bảng kết quả sẽ có hàng x \* y. Vì CROSS JOIN có khả năng tạo các bảng cực lớn, nên phải cẩn thận chỉ sử dụng chúng khi thích hợp.

Cú pháp của CROSS JOIN trong SQLite như sau:

|  |  |
| --- | --- |
| 1 | SELECT ... FROM table1 CROSS JOIN table2 ... |

Dựa vào các bảng trên, bạn có thể viết CROSS JOIN như sau:

|  |  |
| --- | --- |
| 1 | sqlite> SELECT EMP\_ID, NAME, DEPT FROM COMPANY CROSS JOIN DEPARTMENT; |

Truy vấn trên sẽ tạo ra kết quả như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | EMP\_ID      NAME        DEPT  ----------  ----------  ----------  1           Paul        IT Billing  2           Paul        Engineering  7           Paul        Finance  1           Allen       IT Billing  2           Allen       Engineering  7           Allen       Finance  1           Teddy       IT Billing  2           Teddy       Engineering  7           Teddy       Finance  1           Mark        IT Billing  2           Mark        Engineering  7           Mark        Finance  1           David       IT Billing  2           David       Engineering  7           David       Finance  1           Kim         IT Billing  2           Kim         Engineering  7           Kim         Finance  1           James       IT Billing  2           James       Engineering  7           James       Finance |

**3. INNER JOIN trong SQLite**

INNER JOIN tạo một bảng kết quả mới bằng cách kết hợp các giá trị cột của hai bảng (bảng1 và bảng2) dựa trên join-predicate. Truy vấn so sánh từng hàng của bảng1 với từng hàng của bảng2 để tìm tất cả các cặp hàng thỏa mãn join-predicate. Khi join-predicate được thỏa mãn, các giá trị cột cho mỗi cặp hàng A và B phù hợp được kết hợp thành một hàng kết quả.

INNER JOIN là loại join mặc định và phổ biến nhất. Bạn có thể sử dụng từ khóa INNER tùy ý.

Cú pháp của INNER JOIN trong SQLite như sau:

|  |  |
| --- | --- |
| 1 | SELECT ... FROM table1 [INNER] JOIN table2 ON conditional\_expression ... |

Để tránh dư thừa và giữ cho cụm từ ngắn hơn, các điều kiện INNER JOIN có thể được khai báo bằng biểu thức USING. Biểu thức này chỉ định danh sách một hoặc nhiều cột.

|  |  |
| --- | --- |
| 1 | SELECT ... FROM table1 JOIN table2 USING ( column1 ,... ) ... |

NATURAL JOIN tương tự như JOIN ... USING, chỉ khi nó tự động kiểm tra sự bằng nhau giữa các giá trị của mỗi cột tồn tại trong cả hai bảng

|  |  |
| --- | --- |
| 1 | SELECT ... FROM table1 NATURAL JOIN table2... |

Dựa vào các bảng trên, bạn có thể viết INNER JOIN như sau:

|  |  |
| --- | --- |
| 1  2 | sqlite> SELECT EMP\_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT     ON COMPANY.ID = DEPARTMENT.EMP\_ID; |

Truy vấn trên sẽ tạo ra kết quả như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5 | EMP\_ID      NAME        DEPT  ----------  ----------  ----------  1           Paul        IT Billing  2           Allen       Engineering  7           James       Finance |

**4. OUTER JOIN trong SQLite**

OUTER JOIN là một phần mở rộng của INNER JOIN. Mặc dù tiêu chuẩn SQL xác định ba loại OUTER JOIN: LEFT, RIGHT, và FULL, SQLite chỉ hỗ trợ LEFT OUTER JOIN.

OUTER JOIN có một điều kiện giống hệt với INNER JOIN, được thể hiện bằng cách sử dụng từ khóa ON, USING, hoặc NATURAL. Bảng kết quả ban đầu được tính theo cùng một cách. Sau khi tính toán primary JOIN, OUTER JOIN  sẽ lấy bất kỳ hàng không liên kết nào từ một hoặc cả hai bảng, đưa chúng ra bằng NULL và nối chúng vào bảng kết quả.

Sau đây là cú pháp của LEFT OUTER JOIN

|  |  |
| --- | --- |
| 1 | SELECT ... FROM table1 LEFT OUTER JOIN table2 ON conditional\_expression ... |

Để tránh dư thừa và giữ cho cụm từ ngắn hơn, các điều kiện OUTER JOIN có thể được khai báo bằng biểu thức USING. Biểu thức này chỉ định danh sách một hoặc nhiều cột.

|  |  |
| --- | --- |
| 1 | SELECT ... FROM table1 LEFT OUTER JOIN table2 USING ( column1 ,... ) ... |

Dựa vào các bảng trên, bạn có thể viết một inner join như sau:

|  |  |
| --- | --- |
| 1  2 | sqlite> SELECT EMP\_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT     ON COMPANY.ID = DEPARTMENT.EMP\_ID; |

Truy vấn trên sẽ tạo ra kết quả như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | EMP\_ID      NAME        DEPT  ----------  ----------  ----------  1           Paul        IT Billing  2           Allen       Engineering              Teddy              Mark              David              Kim  7           James       Finance |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **Lệnh Inner Join trong SQL Server**

Trong bài này ta sẽ học cách sử dụng lệnh INNERR JOIN trong SQL Server, nó sẽ giúp bạn **liên kết các bảng với nhau thông qua khóa ngoại Foreign Key**.



Hiện có ba các JOIN liên kết khóa ngoại chủ yếu như sau:

* Inner Join
* Left Join
* Right Join

Tuy nhiên trong bài đầu tiên này ta chỉ tìm hiểu **Inner Join** thôi nhé, hai loại còn lại sẽ được trình bày ở các bài tiếp theo.

**Mục lục**

* [1. Giới thiệu Inner Join trong SQL Server](https://freetuts.net/lenh-inner-join-trong-sql-server-2801.html#goto-h2-0)
* [2. Ví dụ về Inner Join trong SQL Server](https://freetuts.net/lenh-inner-join-trong-sql-server-2801.html#goto-h2-1)

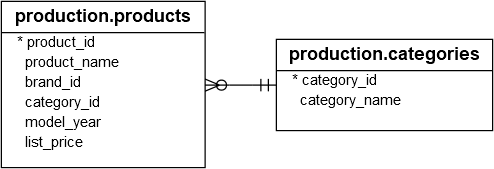
## 1. Giới thiệu Inner Join trong SQL Server

Không chỉ SQL Server mà ở hầu hết các hệ quản trị CSDL quan hệ hiện nay thì Inner join được sử dụng nhiều nhất trong các loại Join, nó giúp ta liên kết nhiều table với nhau thông qua ràng buộc khóa ngoại.

*Bài viết này được đăng tại [free tuts .net]*

ADVERTISEMENT

Giả sử ta có hai table như sau:



Bảng products có một khóa ngoại liên kêt đến bảng categories: products.category\_id = categories.category\_id.

Câu hỏi đặt ra là làm sao lấy được thông tin của mỗi sản phẩm và chuyên mục mà sản phẩm đó thuộc về. Nếu bạn nghĩ đến truy vấn hai bảng thì đó là phép **tích đề cát**, và đương nhiên kết quả sẽ không như mong đợi.

SELECT

\*

FROM

products, categories

Nó sẽ trả ra kết quả là cấp số nhân của tổng record của hai bảng. Ví dụ bảng products có 10 sản phẩm, bảng categories có 2 chuyên mục thì nó trả về tổng cộng 20 record (10 x 2 = 20).

Đây là dữ liệu không đúng, vì vậy ta sẽ phải sử dụng một phép tích khác, **đó là phép Inner Join**.

Cú pháp của Inner Join như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      select\_list  FROM      T1  INNER JOIN T2 ON join\_predicate; |

Trong đó:

* select\_list là danh sách các field bạn muốn chọn ở hai bảng.
* T1 và T2 là hai table cần JOIN với nhau.
* INNER JOIN là toán tử JOIN
* ON join\_predicate là điều kiện JOIN.

Như table trên thì ta sẽ viết câu truy vấn JOIN như sau:

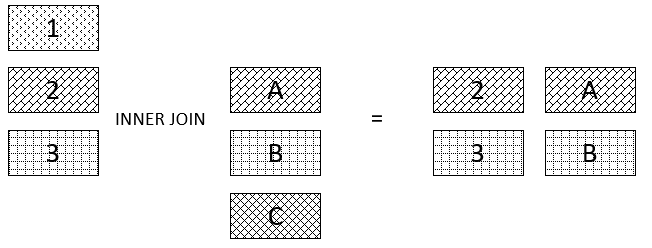
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      product\_name,      category\_name,      list\_price  FROM      production.products p  INNER JOIN production.categories c      ON c.category\_id = p.category\_id  ORDER BY      product\_name DESC; |

Hãy chú ý c.category\_id = p.category\_id chính là điều kiện JOIN giữa hai bảng. Nó sẽ so sánh record giữa hai bảng nếu có giá trị category\_id trùng nhau thì sẽ chọn. Vì vậy kết quả sẽ chính xác hơn.

Kêt quả:



Hãy xem bức ảnh dưới đây để hiểu hơn lệnh Inner Join.



Giữa hai bảng chỉ có hai cặp trùng nhau đó là 2 - A và 3 - B.

Lưu ý rằng Inner Join là từ khóa đầy đủ, bạn có thể rút gọn bằng cách viết JOIN mà thôi.

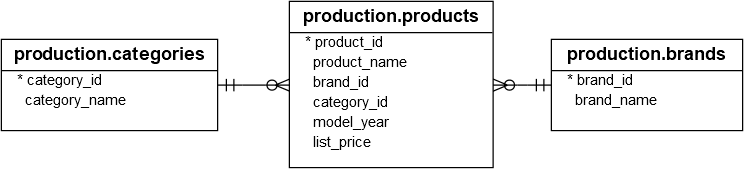
|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      select\_list  FROM      T1  JOIN T2 ON join\_predicate; |

SQL Server sẽ tự động hiểu đây là loại INNER JOIN.

## 2. Ví dụ về Inner Join trong SQL Server

Bây giờ ta sẽ làm một ví dụ phức tạp hơn đó là mối liên hệ **giữa ba bảng.**

Hãy xem cấu trúc của ba bảng như sau:



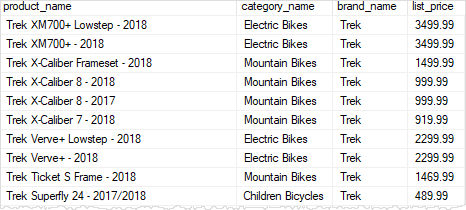
Bảng products sẽ có hai mối liên hệ. Mỗi sản phẩm sẽ thuộc một category và thuộc một brand.

**Bây giờ có bài toán yêu cầu như sau**: Hãy lấy danh sách sản phẩm kèm thông tin về chuyên mục và thương hiệu mà nó thuộc về.

Để làm bài này thì ta thực hiện phép JOIN giữa ba bảng với nhau, và bảng products là bảng chính bởi nó có các khóa ngoại liên kết để hai bảng còn lại.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      product\_name,      category\_name,      brand\_name,      list\_price  FROM      production.products p  INNER JOIN production.categories c ON c.category\_id = p.category\_id  INNER JOIN production.brands b ON b.brand\_id = p.brand\_id  ORDER BY      product\_name DESC; |

Đây là kết quả trả về từ câu truy vấn này.



Trên là những thông tin cơ bản về phép Inner Join trong SQL Server. Hy vọng bạn hiểu bài, nếu có vướng mắc gì hãy bình luận hoặc tham gia nhóm hỏi đáp freetuts để đăng câu hỏi nhé.

# Left Join trong SQL

Ở bài trước chúng ta đã được tìm hiểu [Inner Join](https://freetuts.net/inner-join-trong-sql-1533.html) và biết được tại sao nên sử dụng Inner Join thay thế cho [tích đề các](https://freetuts.net/tich-de-cac-trong-sql-1475.html). Và tiếp tục loạt bài viết học SQL thì trong bài này mình sẽ nói đến Left Join.



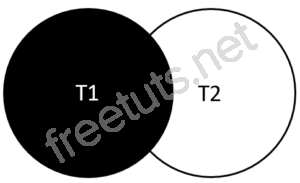
**Mục lục**

* [1. Left Join trong SQL](https://freetuts.net/left-join-trong-sql-1535.html#goto-h2-0)
* [2. Ví dụ Left Join trong SQL](https://freetuts.net/left-join-trong-sql-1535.html#goto-h2-1)

## 1. Left Join trong SQL

Với Inner Join thì kết quả trả về sẽ là hợp của hai bảng, nghĩa là chỉ có những records nào thỏa điều kiện ở ON thì mới được chọn. Còn với Left Join thì khác, nó sẽ ưu tiên chọn tất cả những records thỏa điều kiện ở ON lẫn những record không thỏa điều kiện ở ON trong bảng phía bên trái.

Nói rõ hơn thì nó sẽ chọn tất cả records của table nằm ở phía bên trái, còn table phía bên phải sẽ chọn những records nào thỏa điều kiện ở ON. Xem hình để hiểu rõ hơn.



*Bài viết này được đăng tại [free tuts .net]*

## 2. Ví dụ Left Join trong SQL

ADVERTISEMENT

Giả sử ta có bảng SINHVIEN như sau:

**SINHVIEN**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | MASINHVIEN  |  TENSINHVIEN   |   TUOISINHVIEN  |  MAKHOA  ------------+----------------+-----------------+--------  1           |  Cường         |   28            |  1  2           |  Kính          |   26            |  2  3           |  Quyền         |   20            |  1  4           |  Tình          |   25            |  2  5           |  Hùng          |   25            |  0 |

Và bảng KHOA.

**KHOA**

|  |  |
| --- | --- |
| 1  2  3  4 | MAKHOA  |    TENKHOA  --------+-------------------------------  1       |    Công Nghệ Thông Tin  2       |    Toán |

Bây giờ mình muốn lấy danh sách sinh viên và thông tin về Khoa mà sinh viên đang học. nếu nhìn vào bảng dữ liệu thì sẽ thấy sinh viên có tên là Hùng sẽ không thuộc một khoa nào, nên nếu ta thực hiện phép **Inner Join** thì kết quả trả về sẽ thiếu sinh viên này. Nhưng nếu ta thực hiện **Left Join** thì sẽ có đầ đủ, chỉ có điều các column về Khoa sẽ bị NULL.

|  |  |
| --- | --- |
| 1  2 | SELECT MASINHVIEN, TENSINHVIEN, TUOISINHVIEN, SV.MAKHOA, TENKHOA  FROM SINHVIEN SV LEFT JOIN KHOA K ON SV.MAKHOA = K.MAKHOA |

Kết quả sẽ trả về như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | MASINHVIEN | TENSINHVIEN | TUOISINHVIEN | MAKHOA | TENKHOA  -----------+-------------+--------------+--------+---------  1          |  Cường      |   28         |  1     | Công Nghệ Thông Tin  2          |  Kính       |   26         |  2     | Toán  3          |  Quyền      |   20         |  1     | Công Nghệ Thông Tin  4          |  Tình       |   25         |  2     | Toán  5          |  Hùng       |   25         |  0     | |

Như vậy những columns nào nằm ở bảng bên phải (RIGHT) mà không thỏa điều kiện ở ON thì sẽ có giá trị rỗng.