



WHERE
VS
HAVING



Cho bảng **KetQuaHocTap** như sau:

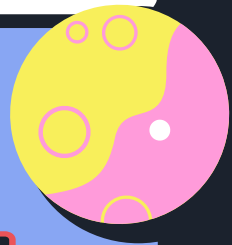
Tên	Môn học	Điểm
A	Toán	8
A	Vật lý	5
B	Toán	4
C	Vật lý	9

WHERE

```
SELECT Tên, sum(Điểm) as  
Tổng điểm  
FROM KetQuaHocTap  
WHERE Điểm > 5  
GROUP BY Tên
```

HAVING

```
SELECT Tên, sum(Điểm) as  
Tổng điểm  
FROM KetQuaHocTap  
WHERE Điểm > 5  
GROUP BY Tên  
HAVING sum(Điểm) > 8
```

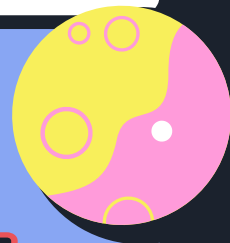


WHERE

Tên	Tổng điểm
A	8
C	9

HAVING

Tên	Tổng điểm
C	9





WHERE

Kiểm tra các điều kiện cho các thuộc tính của bảng, trả kết quả đối chiếu với từng dòng

Không thể sử dụng được với các Aggregate Functions.

Đứng trước GROUP BY

Dùng được với Select, Update, Delete

HAVING

Giới hạn nhóm các hàng trả về trong bảng, trả kết quả đối chiếu cho các nhóm (Sum, Count, Avg,...) được tạo bởi GROUP BY.

Dùng với các điều kiện Aggregate Functions được define trên 1 tập hợp

Đứng sau GROUP BY

Chỉ dùng với Select




The image shows a stylized window frame with a white title bar at the top containing three colored circles (red, yellow, green). The main content area is dark blue and contains the text 'DISTINCT VS GROUP BY'. At the bottom of the window, there is a light blue bar containing a pink pill-shaped button, a white pill-shaped button, and a green progress bar with a red heart icon at the end.


DISTINCT



VS

GROUP BY




Cả hai mệnh đề **DISTINCT** và **GROUP BY** đều làm giảm số lượng bản ghi trả về trong tập kết quả bằng cách loại bỏ các bản ghi trùng lặp.

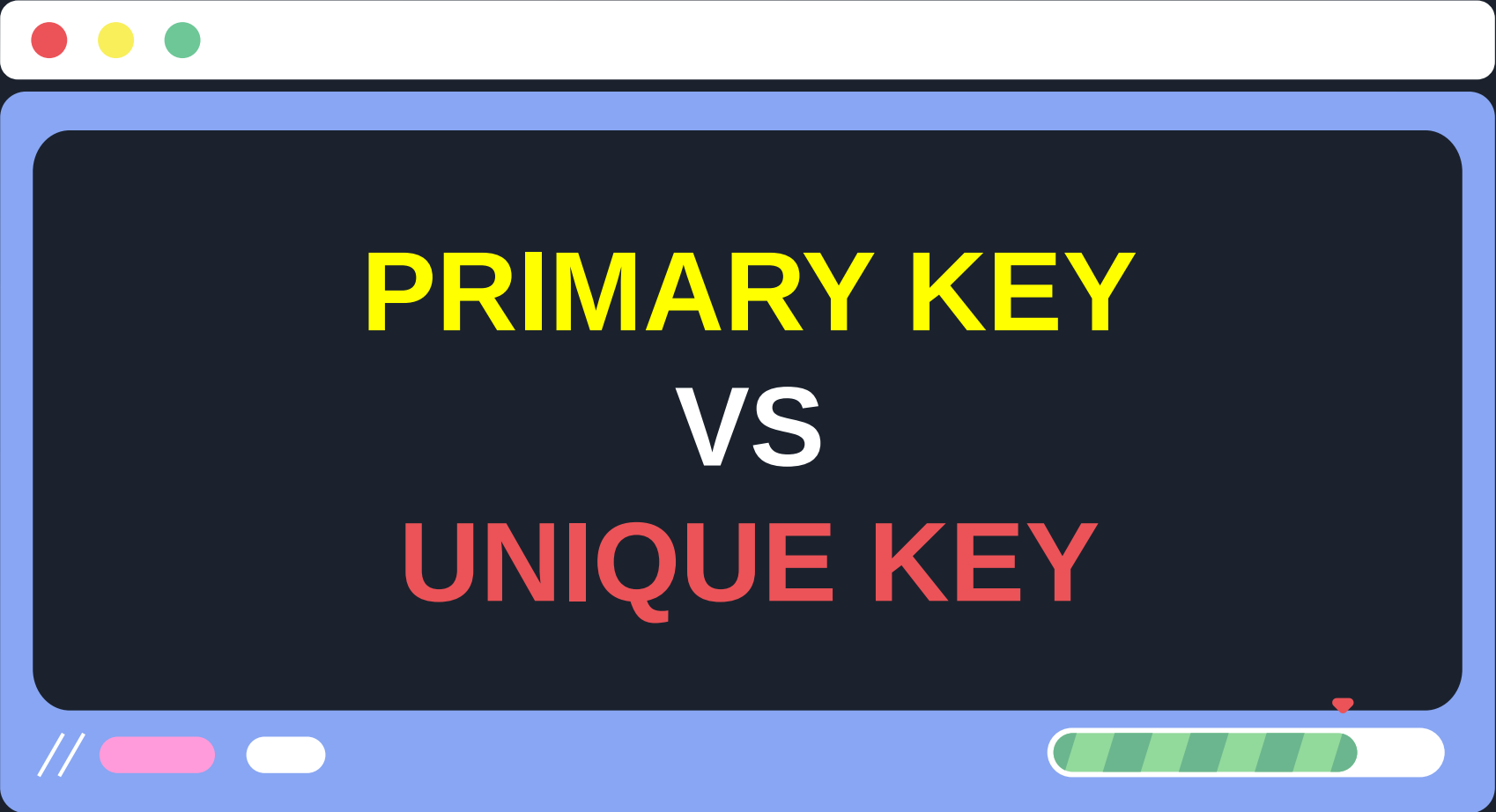




Nếu muốn **sắp xếp dữ liệu giống nhau thành các nhóm**, hãy sử dụng **GROUP BY**, nếu **chỉ muốn một danh sách duy nhất của một cột cụ thể**, hãy sử dụng **DISTINCT**.

GROUP BY được sử dụng để **nhóm các hàng mà bạn muốn tính toán**. **DISTINCT** sẽ **KHÔNG** thực hiện bất kỳ phép tính nào.





PRIMARY KEY VS UNIQUE KEY



	Primary Key	Unique Key
Công dụng	Nó được sử dụng để làm định danh duy nhất cho mỗi hàng trong bảng.	Xác định tính duy nhất của một hàng, nhưng không là khóa chính.
NULL	Không chấp nhận giá trị Null	Chấp nhận giá trị NULL
Số lượng khóa có thể được xác định trong bảng	Chỉ có duy nhất 1 primary key trong 1 bảng	Có thể nhiều hơn 1
Index	Tạo chỉ mục theo nhóm	Tạo chỉ mục không phân cụm



SQL SEVER VS POSTGRES SQL

VỀ LÝ THUYẾT

SQL SEVER	POSTGRESQL
Hệ thống quản trị cơ sở dữ liệu quan hệ	Hệ thống quản trị cơ sở dữ liệu quan hệ đối tượng
Sản phẩm thương mại của Microsoft	Mã nguồn mở (Hoàn toàn miễn phí)
Chỉ chạy trên Microsoft hoặc Linux	Chạy trên hầu hết các hệ điều hành

VỀ CÂU LỆNH

TẠO DATABASE

SQL SEVER

CREATE DATABASE mydb

POSTGRESQL

\$ createdb mydb

XÓA DATABASE

SQL SEVER

DROP DATABASE mydb

POSTGRESQL

\$ dropdb mydb



ĐỔI TÊN TABLE

SQL SEVER

```
EXEC sp_rename 'SinhVien',  
'HocSinh';
```

POSTGRESQL

```
ALTER TABLE SinhVien  
RENAME TO HocSinh
```

ĐỔI TÊN COLUMN TRONG TABLE

SQL SEVER

```
EXEC sp_rename 'HocSinh.MaSV',  
'MaHS', 'COLUMN';
```

POSTGRESQL

```
ALTER TABLE HocSinh  
RENAME COLUMN MaSV  
TO MaHS;
```



THAY ĐỔI KIỂU DỮ LIỆU

SQL SEVER

```
ALTER TABLE SinhVien  
ALTER COLUMN MaHS char(10)
```

POSTGRESQL

```
ALTER TABLE SinhVien  
ALTER COLUMN MaHS TYPE  
char(10)
```

THÊM MỘT COLUMN VÀO TABLE

SQL SEVER

```
ALTER TABLE HocSinh  
ADD DiaChi nvarchar(50)
```

POSTGRESQL

```
ALTER TABLE HocSinh  
ADD COLUMN DiaChi nvarchar(50)
```



XÓA MỘT COLUMN KHỎI TABLE

SQL SEVER

```
ALTER TABLE HocSinh  
DROP DiaChi
```

POSTGRESQL

```
ALTER TABLE HocSinh  
DROP COLUMN DiaChi
```

XÓA DỮ LIỆU TABLE

SQL SEVER

```
DELETE HocSinh  
WHERE MaHS = 'CK101'
```

POSTGRESQL

```
DELETE FROM HocSinh  
WHERE MaHS = 'CK101';
```




PHÉP KẾT NỐI JOIN



ĐỊNH NGHĨA



Join dùng để kết nối nhiều bản ghi từ 2 bảng dữ liệu trong 1 cơ sở dữ liệu quan hệ và kết quả được đưa vào một bảng (tạm). Trong ngôn ngữ truy vấn theo cấu trúc (SQL) có ba loại kết hợp sau: nội, ngoại và chéo. Kết ngoại được chia ra thêm thành kết ngoại bên trái (left outer join), kết ngoại bên phải (right outer join), và kết ngoại đủ (full outer join).





CÁC HÌNH THỨC KẾT HỢP

KẾT NỘI

Phép kết nội (inner join) thực chất là **tìm giao của hai bảng dữ liệu**. Đây là loại kết hợp thường được dùng nhất và được xem như là phép kết hợp mặc định.

Cần đặc biệt chú ý khi kết hợp những bảng trên những cột mà có thể là NULL vì giá trị NULL sẽ không bao giờ có tương ứng.



Bảng "employee"

LastName	DepartmentID
Smith	34
Jones	33
Robinson	34
Jasper	36
Steinberg	33
Rafferty	31

Bảng "department"

DepartmentName	DepartmentID
Sales	31
Engineering	33
Clerical	34
Marketing	35

Ví dụ về phép kết nối

```
SELECT *  
FROM employee  
INNER JOIN department  
ON employee.DepartmentID = department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31



KẾT NGOẠI BÊN TRÁI



Phép kết ngoại bên trái (left outer join) khác rất nhiều với phép kết nội. Thay vì giới hạn những kết quả thu được trong cả hai bảng, nó chỉ giới hạn đối với những kết quả ở bảng bên trái (A). Nghĩa là nếu mệnh đề ON không có bản ghi tương ứng bên bảng B, 1 dòng trong kết quả vẫn được trả về nhưng với giá trị NULL cho mỗi cột trong bảng B.

Nó trả về tất cả những giá trị từ bản bên trái + những giá trị tương ứng với bảng bên phải hoặc là null (khi những giá trị ở bảng bên phải không tương ứng)

Ví dụ về phép kết ngoại bên trái

```
SELECT *  
FROM employee  
LEFT OUTER JOIN department  
ON employee.DepartmentID = department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Jasper	36	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31



KẾT NGOẠI BÊN PHẢI



Phép kết ngoại bên phải (right outer join) hầu như tương tự với phép kết ngoại bên trái, trừ 1 điều là thứ tự các bảng đổi lại. Mỗi bản ghi từ bảng bên phải (B) sẽ được trả về và giá trị NULL sẽ được trả về cho những dòng mà không có bản ghi tương ứng bên bảng A.

Nó trả về tất cả những giá trị từ bảng bên phải + giá trị tương ứng từ bảng bên trái (hoặc null)

Ví dụ về phép kết ngoại bên phải

```
SELECT *  
FROM employee  
RIGHT OUTER JOIN department  
ON employee.DepartmentID = department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35



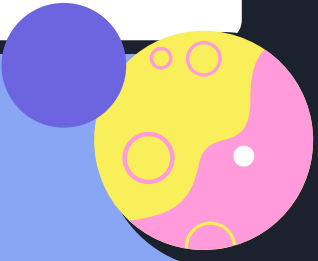

KẾT NGOẠI ĐỦ

Phép kết ngoại đủ (full outer join) kết hợp cả kết quả của cả phép kết ngoại bên trái và phép kết ngoại bên phải. Những phép kết này đưa ra bản ghi của cả hai bảng dữ liệu, và lấp đầy những dòng tương ứng bị thiếu của cả hai phía bằng NULLs.

Ví dụ về phép kết ngoại đủ

```
SELECT *  
FROM employee  
FULL OUTER JOIN department  
ON employee.DepartmentID = department.DepartmentID
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Jasper	36	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35



```
SELECT employee.LastName, employee.DepartmentID,  
department.DepartmentName, department.DepartmentID  
FROM employee  
LEFT JOIN department  
ON employee.DepartmentID = department.DepartmentID  
UNION  
SELECT employee.LastName, employee.DepartmentID,  
department.DepartmentName, department.DepartmentID  
FROM employee  
RIGHT JOIN department  
ON employee.DepartmentID = department.DepartmentID  
WHERE employee.DepartmentID IS NULL
```



KẾT CHÉO



Mặc dù không được sử dụng thường xuyên, phép kết chéo (cross join) là cơ sở mà dựa trên đó phép kết nội được tạo nên. Phép kết chéo trả về tích Descartes của những tập hợp các dòng từ những bảng được kết.

```
SELECT *  
FROM employee CROSS JOIN department;
```

LastName	DepartmentID	DepartmentName	DepartmentID
Smith	34	Sales	31
Smith	34	Engineering	33
Smith	34	Clerical	34
Smith	34	Marketing	35
Jones	33	Sales	31
Jones	33	Engineering	33
Jones	33	Clerical	34
Jones	33	Marketing	35
Robinson	34	Sales	31
Robinson	34	Engineering	33
Robinson	34	Clerical	34
Robinson	34	Marketing	35
Jasper	36	Sales	31
Jasper	36	Engineering	33
Jasper	36	Clerical	34
Jasper	36	Marketing	35
Steinberg	33	Sales	31
Steinberg	33	Engineering	33
Steinberg	33	Clerical	34
Steinberg	33	Marketing	35
Rafferty	31	Sales	31
Rafferty	31	Engineering	33
Rafferty	31	Clerical	34
Rafferty	31	Marketing	35

Thanks~

