INNER JOIN - JOIN & THETA JOIN

Cover 

▼ **INNER JOIN & JOIN**

1. **JOIN INTER**

- `INNER JOIN` là một loại kết nối được sử dụng để kết hợp các hàng từ hai hoặc nhiều bảng dựa trên một điều kiện cụ thể.
- Nó chỉ trả về các hàng có các giá trị khớp nhau trong cả hai bảng.
- Cú pháp thường dùng:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

2. **JOIN**

- Trong nhiều hệ quản trị cơ sở dữ liệu (DBMS) như MySQL, SQL Server, PostgreSQL, và một số khác, khi bạn chỉ sử dụng từ khóa `JOIN` mà không chỉ rõ loại kết nối, nó sẽ mặc định hiểu là `INNER JOIN`.
- Vì vậy, `JOIN` và `INNER JOIN` trong các hệ quản trị cơ sở dữ liệu này là tương đương nhau.
- Cú pháp thường dùng:

```
SELECT columns
FROM table1
JOIN table2
ON table1.common_column = table2.common_column;
```

Ví dụ minh họa 1:

Giả sử chúng ta có hai bảng sau:

- **Bảng `Employees`:**

EmployeeID	Name	DepartmentID
1	Alice	1
2	Bob	2
3	Carol	NULL

- **Bảng `Departments`:**

DepartmentID	DepartmentName
1	HR
2	IT

Câu lệnh sử dụng `INNER JOIN`:

```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Câu lệnh sử dụng `JOIN`:

```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Cả hai câu lệnh trên sẽ trả về cùng một kết quả:

Name	DepartmentName
Alice	HR
Bob	IT

Vậy nên, trong nhiều hệ quản trị cơ sở dữ liệu, `JOIN` và `INNER JOIN` thực sự không có sự khác biệt về kết quả truy vấn. Tuy nhiên, để mã SQL của bạn rõ ràng và dễ đọc, bạn nên sử dụng `INNER JOIN` khi ý định của bạn là thực hiện một phép kết nối bên trong (**inner join**)

Ví dụ minh họa 2:

Bảng Employees:

EmployeeID	EmployeeName
1	John Doe
2	Jane Smith
3	Emily Johnson

Bảng Departments:

EmployeeID	DepartmentName
1	Human Resources
2	Finance
3	Haha

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    EmployeeName NVARCHAR(100),
);

INSERT INTO Employees (EmployeeID, EmployeeName) VALUES
(1, 'John Doe'),
(2, 'Jane Smith'),
(3, 'Emily Johnson');

CREATE TABLE Departments (
    EmployeeID INT PRIMARY KEY,
    DepartmentName NVARCHAR(100)
);

INSERT INTO Departments (EmployeeID, DepartmentName) VA
(1, 'Human Resources'),
(2, 'Finance'),
(3, 'Haha');

SELECT Employees.EmployeeID, Employees.EmployeeName, De
FROM Employees
INNER JOIN Departments
```

```
ON Employees.EmployeeID = Departments.EmployeeID;

SELECT Employees.*,  Departments.*
FROM Employees
INNER JOIN Departments
ON Employees.EmployeeID = Departments.EmployeeID;
```

EmployeeID	EmployeeName	DepartmentName
1	John Doe	Human Resources
2	Jane Smith	Finance
3	Emily Johnson	Haha

EmployeeID	EmployeeName	EmployeeID	DepartmentName
1	John Doe	1	Human Resources
2	Jane Smith	2	Finance
3	Emily Johnson	3	Haha

Giả sử thay đổi dữ liệu một chút

Bảng Employees:

EmployeeID	EmployeeName
1	John Doe
2	Jane Smith

Bảng Departments:

EmployeeID	DepartmentName
1	Human Resources
2	Finance
3	Haha

Kết quả:

EmployeeID	EmployeeName	DepartmentName
1	John Doe	Human Resources
2	Jane Smith	Finance

Giải thích:

- **INNER JOIN = JOIN** kết hợp các hàng từ bảng `Employees` và bảng `Departments` dựa trên giá trị chung của cột `EmployeeID`.
- Kết quả chỉ bao gồm các hàng mà `EmployeeID` tồn tại trong cả hai bảng.
- Các cột được trả về là `EmployeeID`, `EmployeeName` từ bảng `Employees` và `DepartmentName` từ bảng `Departments`.

Ví dụ minh họa 3:

Bảng `Employees`:

EmployeeID	EmployeeName	DepartmentID
1	John Doe	1
2	Jane Smith	2
3	Emily Johnson	1
4	Michael Brown	3
5	Robert Green	NULL

Bảng `Departments`:

DepartmentID	DepartmentName
1	Human Resources
2	Finance
3	IT
4	Marketing

```
SELECT Employees.EmployeeID, Employees.EmployeeName, De
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Kết quả:

EmployeeID	EmployeeName	DepartmentName
1	John Doe	Human Resources

2	Jane Smith	Finance
3	Emily Johnson	Human Resources
4	Michael Brown	IT

▼ SỰ KHÁC NHAU GIỮA INNER JOIN VÀ JOIN

1. **Khái niệm**

Trong SQL Server, `INNER JOIN ... ON ...` và `JOIN ... ON ...` không có sự khác nhau về mặt chức năng. `INNER JOIN` là một cách cụ thể hơn để chỉ định loại phép nối mà bạn muốn sử dụng, trong khi `JOIN` mặc định cũng là `INNER JOIN` nếu không có từ khóa nào khác được chỉ định.

2. **Ví dụ chi tiết**

Giả sử chúng ta có hai bảng `Employees` và `Departments` như sau:

Bảng `Employees`:

EmployeeID	EmployeeName	DepartmentID
1	John Doe	1
2	Jane Smith	2
3	Emily Johnson	1
4	Michael Brown	3
5	Robert Green	NULL

Bảng `Departments`:

DepartmentID	DepartmentName
1	Human Resources
2	Finance
3	IT
4	Marketing

Sử dụng `INNER JOIN ... ON ...`

```
SELECT Employees.EmployeeID, Employees.EmployeeName, Dep
artments.DepartmentName
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Sử dụng `JOIN ... ON ...`

```
SELECT Employees.EmployeeID, Employees.EmployeeName, Dep
artments.DepartmentName
FROM Employees
JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Kết quả cho cả hai truy vấn:

EmployeeID	EmployeeName	DepartmentName
1	John Doe	Human Resources
2	Jane Smith	Finance
3	Emily Johnson	Human Resources
4	Michael Brown	IT

Giải thích

1. `INNER JOIN ... ON ...`:
    - `INNER JOIN` chỉ rõ ràng rằng bạn muốn thực hiện phép nối trong, tức là chỉ lấy các hàng mà có giá trị khớp nhau trong cả hai bảng.
2. `JOIN ... ON ...`:
    - `JOIN` là từ khóa chung chung và mặc định là `INNER JOIN`. Nếu bạn chỉ viết `JOIN`, SQL Server sẽ hiểu rằng bạn đang yêu cầu một phép nối trong.

Vì vậy, cả hai truy vấn trên sẽ trả về kết quả giống hệt nhau. Việc sử dụng `INNER JOIN` hoặc chỉ `JOIN` tùy thuộc vào phong cách viết mã của bạn và mức độ rõ ràng bạn muốn thể hiện trong mã SQL của mình.

Trong một số trường hợp, để mã SQL rõ ràng hơn và dễ hiểu hơn, nhiều nhà phát triển sẽ sử dụng `INNER JOIN` để làm rõ ý định của họ, đặc biệt khi có thể có các loại phép nối khác như `LEFT JOIN`, `RIGHT JOIN`, hoặc `FULL JOIN` trong cùng một truy vấn.

▼ **THETA JOIN**

1. **Công dụng**

Phép nối cụ thể (Theta Join) là một phép nối mà các bảng được liên kết dựa trên một điều kiện cụ thể nào đó, không nhất thiết phải là điều kiện bằng nhau. Điều kiện này có thể bao gồm các toán tử như =, <>, <, >, <=, >=.

2. **Cú pháp**

```
SELECT *
FROM table1
JOIN table2
ON condition;
```

3. **Ví dụ minh họa**

```
--Bảng Students
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    StudentName NVARCHAR(100),
    Age INT
);
--Chèn dữ liệu vào bảng Students
INSERT INTO Students (StudentID, StudentName, Age) VALUES
(1, 'Alice', 18),
(2, 'Bob', 20),
(3, 'Charlie', 22),
(4, 'David', 24);

--Bảng Courses
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName NVARCHAR(100),
    MinAge INT
);
```

```
--Chèn dữ liệu vào bảng Courses
INSERT INTO Courses (CourseID, CourseName, MinAge) VALUES
(1, 'Math 101', 18),
(2, 'History 201', 20),
(3, 'Science 301', 21);
```

Dữ liệu bảng `Students`:

StudentID	StudentName	Age
1	Alice	18
2	Bob	20
3	Charlie	22
4	David	24

Dữ liệu bảng `Courses`:

CourseID	CourseName	MinAge
1	Math 101	18
2	History 201	20
3	Science 301	21

Truy vấn:

```
SELECT Students.StudentID, Students.StudentName, Course
s.CourseName
FROM Students
INNER JOIN Courses
ON Students.Age >= Courses.MinAge;
```

Giải thích:

- **INNER JOIN** kết hợp các hàng từ bảng `Students` và bảng `Courses` dựa trên điều kiện cụ thể `Students.Age >= Courses.MinAge`.
- Điều này có nghĩa là chỉ các học sinh có tuổi (Age) lớn hơn hoặc bằng tuổi tối thiểu (MinAge) của khóa học mới được kết hợp.

Kết quả:

StudentID	StudentName	CourseName
1	Alice	Math 101
2	Bob	Math 101
2	Bob	History 201
3	Charlie	Math 101
3	Charlie	History 201
3	Charlie	Science 301
4	David	Math 101
4	David	History 201
4	David	Science 301

Các hàng kết quả cho thấy:

- Alice chỉ đủ điều kiện tham gia khóa `Math 101` vì cô ấy 18 tuổi.
- Bob đủ điều kiện tham gia cả `Math 101` và `History 201` vì anh ấy 20 tuổi.
- Charlie đủ điều kiện tham gia tất cả các khóa học vì anh ấy 22 tuổi.
- David cũng đủ điều kiện tham gia tất cả các khóa học vì anh ấy 24 tuổi.

Phép nối cụ thể cho phép chúng ta sử dụng các điều kiện nối phức tạp hơn, không chỉ giới hạn ở việc kiểm tra sự bằng nhau. Điều này rất hữu ích khi cần so sánh các giá trị bằng cách sử dụng các toán tử khác như lớn hơn, nhỏ hơn, hoặc các điều kiện phức tạp hơn.