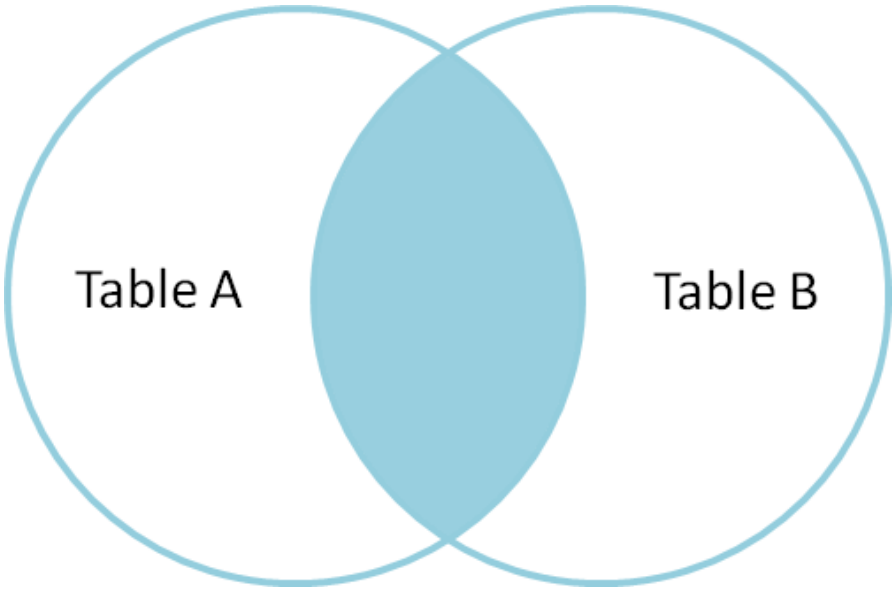
# Các Loại JOIN Trong SQL Server

Vũ Huy Tâm

JOIN là phép kết nối dữ liệu từ nhiều bảng lại với nhau. Khi bạn cần truy vấn các cột dữ liệu từ nhiều bảng khác nhau để trả về trong cùng một tập kết quả, bạn cần dùng JOIN. Đây có lẽ là chức năng được dùng nhiều nhất khi lập trình T-SQL. Nó giúp tái hiện lại thông tin thế giới thực từ dữ liệu lưu trữ trong mô hình quan hệ. Ví dụ, bạn cần JOIN bảng BanHang với bảng SanPham thông qua SanPhamID để lấy về thông tin đầy đủ của một đơn hàng bao gồm cả tên sản phẩm, vì người dùng cần quan tâm đến sản phẩm đó là gì thay vì mã hiệu của nó.
SQL Server cung cấp các kiểu JOIN là INNER JOIN, OUTER JOIN,.

**INNER JOIN** trả về kết quả là các bản ghi mà trường được join ở hai bảng khớp nhau, các bản ghi chỉ xuất hiện ở một trong hai bảng sẽ bị loại.

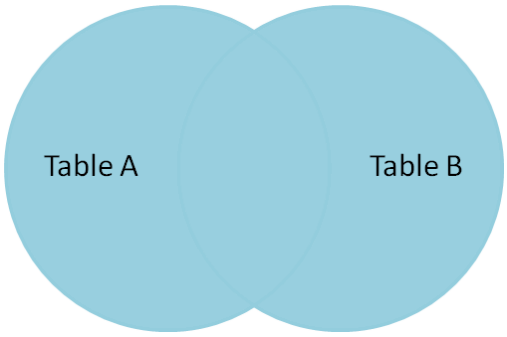



(hình lấy từ codinghorror.com)

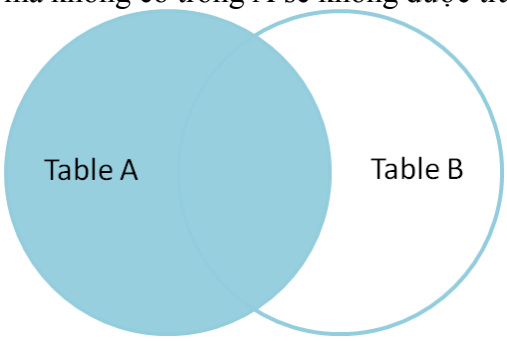
**OUTER JOIN** nới lỏng hơn, lấy về các bản ghi có mặt trong cả hai bảng và cả các bản ghi chỉ xuất hiện ở một trong hai bảng. Kiểu JOIN này được chia làm hai loại:

- FULL OUTER JOIN: kết quả gồm tất cả các bản ghi của cả hai bảng. Với các bản ghi chỉ xuất hiện trong một bảng thì các cột dữ liệu từ bảng kia được điền giá trị

NULL.




- HALF OUTER JOIN (LEFT hoặc RIGHT): nếu bảng A LEFT OUTER JOIN với bảng B thì kết quả gồm các bản ghi có trong bảng A, với các bản ghi không có mặt trong bảng B thì các cột từ B được điền NULL. Các bản ghi chỉ có trong B mà không có trong A sẽ không được trả về.




**CROSS JOIN**: mỗi bản ghi của bảng A được kết hợp với tất cả các bản ghi của bảng B, tạo thành một tích Đề-các giữa hai bảng (số bản ghi trả về bằng tích của số bản ghi trong hai bảng).

Ví dụ:

```
CREATE TABLE T1(ID1 INT, Ten VARCHAR(100) )
INSERT INTO T1
SELECT 1, 'Mozart' UNION ALL
SELECT 2, 'Beethoven' UNION ALL
SELECT 3, 'Chopin'

CREATE TABLE T2(ID2 INT, Email VARCHAR(100) )
INSERT INTO T2
SELECT 2, 'beethoven@gmail.com' UNION ALL
SELECT 3, 'chopin@hotmail.com'  UNION ALL
SELECT 4, 'haydn@yahoo.com'  UNION ALL
SELECT 5, 'bach@yahoo.com'
```

INNER JOIN:

```
SELECT *
FROM T1 JOIN T2 ON T1.ID1 = T2.ID2
```

Results | Messages

| | ID1 | Ten | ID2 | Email |
|---|---|---|---|---|
| 1 | 2 | Beethoven | 2 | beethoven@gmail.com |
| 2 | 3 | Chopin | 3 | chopin@hotmail.com |

FULL OUTER JOIN:

```
SELECT *
FROM T1 FULL OUTER JOIN T2 ON T1.ID1 = T2.ID2
```

Results | Messages

| | ID1 | Ten | ID2 | Email |
|---|---|---|---|---|
| 1 | 1 | Mozart | NULL | NULL |
| 2 | 2 | Beethoven | 2 | beethoven@gmail.com |
| 3 | 3 | Chopin | 3 | chopin@hotmail.com |
| 4 | NULL | NULL | 4 | haydn@yahoo.com |
| 5 | NULL | NULL | 5 | bach@yahoo.com |

LEFT OUTER JOIN:

```
SELECT *
FROM T1 LEFT JOIN T2 ON T1.ID1 = T2.ID2
```

Results | Messages

| | ID1 | Ten | ID2 | Email |
|---|---|---|---|---|
| 1 | 1 | Mozart | NULL | NULL |
| 2 | 2 | Beethoven | 2 | beethoven@gmail.com |
| 3 | 3 | Chopin | 3 | chopin@hotmail.com |

CROSS JOIN:

```
SELECT *
FROM T1 CROSS JOIN T2
```

Results | Messages

| | ID1 | Ten | ID2 | Email |
|---|---|---|---|---|
| 1 | 1 | Mozart | 2 | beethoven@gmail.com |
| 2 | 1 | Mozart | 3 | chopin@hotmail.com |
| 3 | 1 | Mozart | 4 | haydn@yahoo.com |
| 4 | 1 | Mozart | 5 | bach@yahoo.com |
| 5 | 2 | Beethoven | 2 | beethoven@gmail.com |
| 6 | 2 | Beethoven | 3 | chopin@hotmail.com |
| 7 | 2 | Beethoven | 4 | haydn@yahoo.com |
| 8 | 2 | Beethoven | 5 | bach@yahoo.com |
| 9 | 3 | Chopin | 2 | beethoven@gmail.com |
| 10 | 3 | Chopin | 3 | chopin@hotmail.com |
| 11 | 3 | Chopin | 4 | haydn@yahoo.com |
| 12 | 3 | Chopin | 5 | bach@yahoo.com |

Lưu ý là "A INNER JOIN B" có thể viết tắt thành "A JOIN B", còn "A LEFT OUTER JOIN B" có thể viết "A LEFT JOIN B".

Trong các loại JOIN trên, chỉ trừ HALF OUTER JOIN còn tất cả đều có tính đối xứng, nghĩa là "A JOIN B" tương tự như "B JOIN A". Riêng HALF OUTER JOIN thì phân biệt thứ tự, ví dụ "A LEFT JOIN B" khác với "B LEFT JOIN A". Tuy nhiên, "A LEFT JOIN B" tương đương với "B RIGHT JOIN A", vì thế để tránh nhầm lẫn tôi luôn hay dùng LEFT JOIN thay cho RIGHT JOIN, và ngầm qui định trong đầu là A là bảng chính còn B là bảng join (bảng để kéo thêm dữ liệu vào).

Bạn cần đặc biệt chú ý khi dùng CROSS JOIN vì nó có thể tạo ra bùng nổ về số bản ghi. Ví dụ nếu hai bảng có số bản ghi tương ứng là 1 nghìn và 1 triệu thì kết quả sẽ là 1 tỷ bản ghi. Nói chung CROSS JOIN thường được dùng rất ít nhưng vẫn có lúc cần, ví dụ bảng kết quả điểm thi của sinh viên là CROSS JOIN của bảng sinh viên và bảng môn học (giả sử mỗi sinh viên cần lấy đủ tất cả các môn học).

**Các kiểu viết JOIN**

Cách viết JOIN như trên gọi là theo kiểu ANSI (ANSI style), cách viết kiểu cũ là đưa điều kiện join vào mệnh đề WHERE:

```
--INNER JOIN kiểu ANSI
SELECT *
FROM T1 JOIN T2 ON T1.ID = T2.ID
```

```sql
--INNER JOIN kiểu cũ
SELECT *
FROM T1, T2
WHERE T1.ID = T2.ID

-- LEFT OUTER JOIN kiểu ANSI
SELECT *
FROM T1 LEFT JOIN T2 ON T1.ID = T2.ID
--LEFT OUTER JOIN kiểu cũ
SELECT *
FROM T1, T2
WHERE T1.ID *= T2.ID
```

Cả hai kiểu viết đều cho cùng kết quả và cũng không khác nhau về hiệu năng thực hiện. Tuy vây bạn có thể thấy cách viết kiểu ANSI trong sáng hơn. Kiểu viết này gần với diễn đạt của ngôn ngữ tự nhiên hơn, nó tách bạch rõ ràng điều kiện join ra khỏi điều kiện lọc dữ liệu (dùng ở mệnh đề WHERE). Đây là kiểu viết bạn nên dùng. Microsoft trong 10 năm qua liên tục khuyến cáo dùng kiểu viết ANSI và nhắc nhở, các phiên bản sau có thể sẽ không hỗ trợ kiểu cũ. Và đến nay, bản SQL Server 2008 đã không còn hỗ trợ LEFT JOIN viết theo kiểu cũ (trừ khi bạn phải đặt lại COMPATIBILITY_LEVEL xuống mức thấp hơn).