

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO BÀI TẬP LỚN**

Môn Học: *Thiết kế và quản trị cơ sở dữ liệu*

Đề Tài: *Join Operators.*

Giảng viên: TS Trần Việt Trung

Nhóm sinh viên:

Nguyễn Thế Minh	20111863
Vũ Công Hào	20111626
Nguyễn Hoàng Khương	20111700
Nguyễn Quốc Việt	20112500

Hà Nội, tháng 5 năm 2015

## Nội dung

1. Tổng quan về phép nối.....	3
2. Nested Loops Join.....	4
2.1 Nested Loops Join .....	4
2.2 Nested Loops Join: Prefetching .....	6
2.3 Nested Loops Join: 11g implementation .....	7
3. Sort Merge Join.....	8
4 . Hash Join .....	9
5. Cartesian Join.....	10
6. Join Type .....	11
6.1 Equijoin and Nonequijoin.....	12
6.2 Outer Join.....	12
6.3 Semijoins .....	13
6.4 Anjoins.....	14

## 1. Tổng quan về phép nối

- Tìm hiểu và đánh giá các thuật toán thực hiện phép nối quan hệ
  - Nested Loops Join
  - Sort Merge Join
  - Hash Join
- Mô tả cả phép toán SQL dành cho phép nối quan hệ.
- Tìm hiểu các loại phép nối khác nhau.

Một phép nối:

- Định nghĩa mối quan hệ giữa hai nguồn dữ liệu.
- Là phương pháp tổ hợp dữ liệu từ hai nguồn dữ liệu.
- Được kiểm soát bởi các điều kiện nối - xác định các đối tượng có quan hệ như thế nào.

### Join Methods

A join:

- Defines the relationship between two row sources
- Is a method of combining data from two data sources
- Is controlled by join predicates, which define how the objects are related
- Join methods:
  - Nested loops
  - Sort-merge join
  - Hash join

```
SELECT e.ename, d.dname
FROM dept d JOIN emp e USING (deptno)      ← Join predicate
WHERE e.job = 'ANALYST' OR e.empno = 9999; ← Nonjoin predicate
```

```
SELECT e.ename, d.dname
FROM   emp e, dept d
WHERE  e.deptno = d.deptno AND              ← Join predicate
      (e.job = 'ANALYST' OR e.empno = 9999); ← Nonjoin predicate
```

ORACLE

## Join Methods - Phương pháp nối

Nguồn dữ liệu là một tập dữ liệu có thể tiếp cận bằng các câu truy vấn. Nó có thể là bảng, một chỉ mục, một khung nhìn, hoặc là tập kết quả của một cây nối (tổng hợp các phép nối) chứa nhiều đối tượng khác nhau.

Một điều kiện nối là một điều kiện trong mệnh đề WHERE, dùng để tổ hợp các cột của hai bảng trong phép nối.

Một điều kiện không nối là một điều kiện trong mệnh đề WHERE chỉ liên kết tới một bảng.

Một phép nối tổ hợp kết quả đầu ra từ hai nguồn dữ liệu ( ví dụ như các bảng hay các khung nhìn) và trả về một nguồn dữ liệu kết quả ( tập dữ liệu). Bộ tối ưu câu lệnh có hỗ trợ các thuật toán khác nhau dành cho phép nối gồm các thuật toán sau:

- Nested loops join: Hữu ích khi các tập con dữ liệu nhỏ của dữ liệu được nối với nhau và nếu điều kiện nối cho ta một cách truy cập hiệu quả tới bảng thứ hai.
- Sort Merge Join: Được sử dụng để nối các dòng từ hai nguồn dữ liệu độc lập. Hash joins thông thường có thể thực hiện tốt hơn sort-merge joins. Mặt khác, sort-merge joins có thể thực hiện tốt hơn hash-joins nếu một hay hai nguồn dữ liệu đã được sắp xếp.
- Hash join: Được sử dụng khi nối các tập dữ liệu lớn. Bộ tối ưu sẽ chọn bảng hay nguồn dữ liệu nhỏ hơn trong hai bảng hay hai nguồn dữ liệu để xây dựng một bảng băm dựng trên khóa kết nối trong bộ nhớ chính. Sau đó bộ tối ưu sẽ quét bảng dữ liệu lớn hơn, duyệt bảng băm để tìm thấy các dòng được nối. Phương pháp này là tốt nhất khi bảng dữ liệu nhỏ hơn có thể nạp vừa vào bộ nhớ chính. Chi phí thực hiện được giới thành một luồng đọc cho toàn bộ dữ liệu của cả hai bảng.

## 2. Nested Loops Join

- Các dòng dữ liệu được quét.
- Mỗi dòng dữ liệu được trả về sẽ điều khiển a vòng lặp tìm kiếm trong nguồn dữ liệu thứ hai.
- Nối các dòng dữ liệu rồi trả về kết quả.

### 2.1 Nested Loops Join

Trong công thức tổng quát của nested-loops join, một trong hai bảng được định nghĩa là bảng ngoài hay bảng điều khiển. Bảng còn lại được gọi là bảng trong, hay bảng thứ.

Đối với mỗi dòng trong bảng ngoài thỏa mãn điều kiện không nối, tất cả các dòng trong bảng trong thỏa mãn điều kiện nối được phản hồi. Nếu có một chỉ mục, chỉ mục này có thể được sử dụng để truy cập bảng trong bằng khóa của dòng trong bảng (rowid).

Mọi điều kiện không nối trong bảng trong được xác định sau lần phản hồi đầu tiên, trừ phi có một chỉ mục phức tạp tổ hợp cả điều kiện kết nối và điều kiện không kết nối được sử dụng.

## Nested Loops Join

- Driving row source is scanned.
- Each row returned drives a lookup in inner row source.
- Joining rows are then returned.



```
select ename, e.deptno, d.deptno, d.dname
from emp e, dept d
where e.deptno = d.deptno and ename like 'A%';
```

Id	Operation	Name	Rows	Cost
0	SELECT STATEMENT		2	4
1	NESTED LOOPS		2	4
2	TABLE ACCESS FULL	EMP	2	2
3	TABLE ACCESS BY INDEX ROWID	DEPT	1	1
4	INDEX UNIQUE SCAN	PK_DEPT	1	1

```

2 - filter("E"."ENAME" LIKE 'A%')
4 - access("E"."DEPTNO"="D"."DEPTNO")
  
```

ORACLE

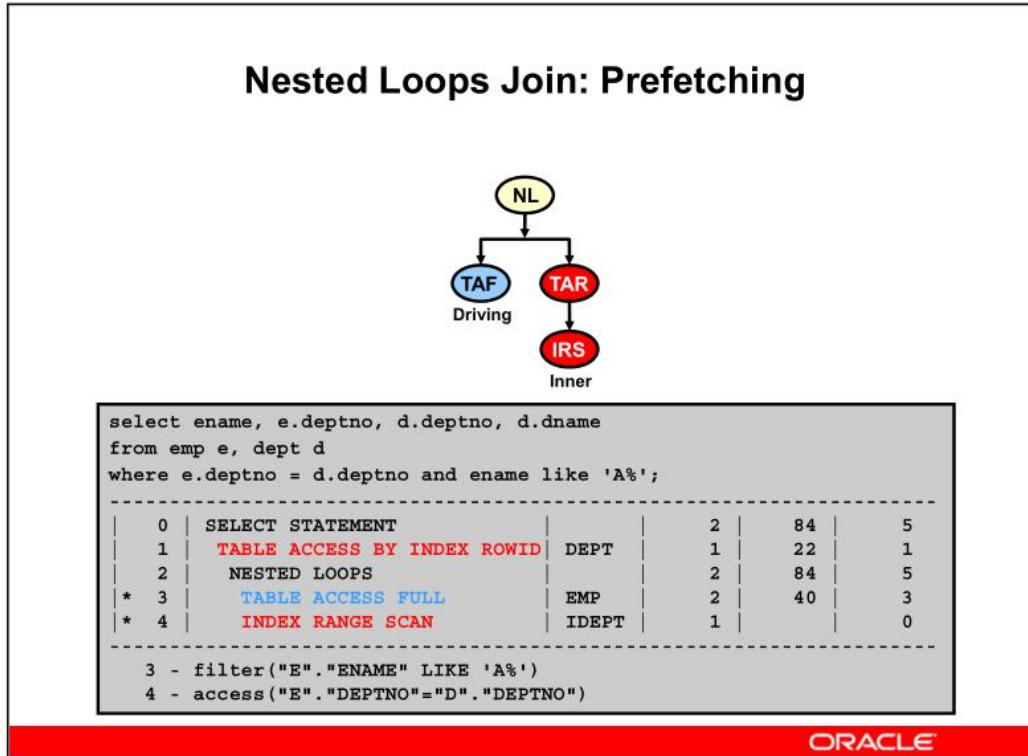
Thuật toán mô tả nested-loops join được thể hiện như sau:

```

for each row R1 in the outer table
  for each row R2 in the inner table
    if R1.join_column = R2.join_column
      return (R1, R2)
  
```

Bộ tối ưu sử dụng nested loop joins khi kết nối số lượng nhỏ các dòng, cùng với một điều kiện nối tốt giữa hai bảng. Bạn có thể chạy từ vòng lặp ngoài tới vòng lặp trong, do đó thứ tự các bảng trong kế hoạch thực hiện là rất quan trọng. Vì vậy, bạn nên sử dụng phương pháp kết nối khác khi hai nguồn dữ liệu là độc lập được kết nối.

## 2.2 Nested Loops Join: Prefetching



Oracle 9iR2 đã giới thiệu một cơ chế gọi là nested-loops prefetching. Ý tưởng này nhằm cải tiến vấn đề tối ưu truy cập hệ thống vào ra, cũng như thời gian hồi đáp, quá trình tìm kiếm trên bảng sử dụng chỉ mục được thực hiện song song.

Sự thay đổi trong kế hoạch đầu ra không làm kế hoạch thực hiện lệnh. Nó không ảnh hưởng tới thứ tự kết nối, phương pháp kết nối, phương pháp truy cập hay chiến lược song song hóa.

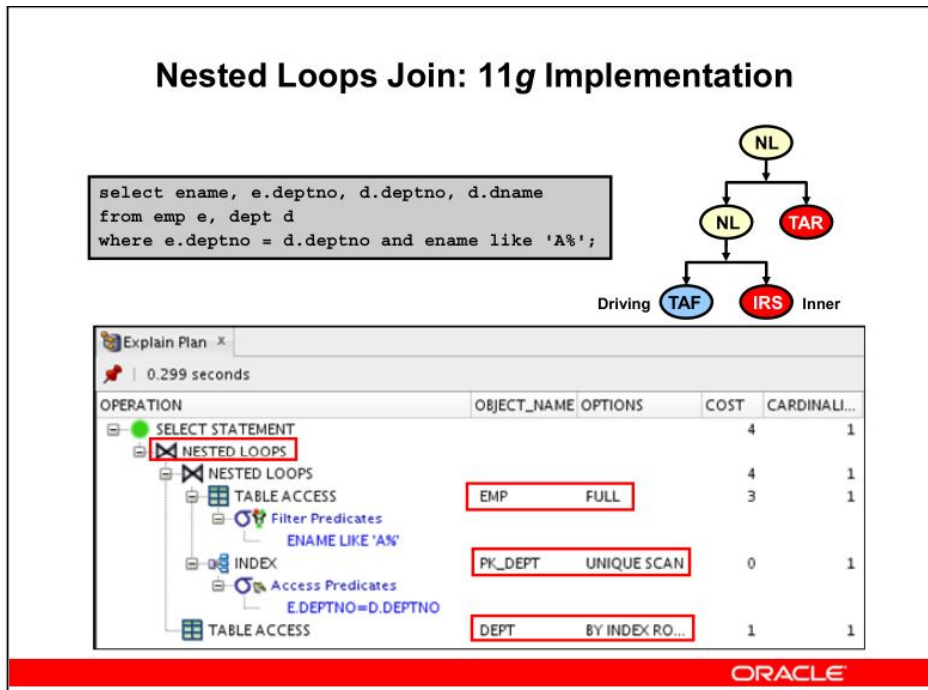
Kỹ thuật tối ưu này chỉ phù hợp khi đường dẫn truy cập bên trong là quét một khoảng chỉ mục và không phù hợp nếu đường dẫn truy cập là quét chỉ mục duy nhất. (phù hợp cho truy vấn khoảng).

Cơ chế prefetching ( nạp trước) được sử dụng khi tìm kiếm trên bảng. Khi một đường dẫn chỉ mục được chọn và câu truy vấn không thể thỏa mãn bởi một mình chỉ mục, dữ liệu xác định bởi ROWID cũng phải được nạp. Truy cập ROWID của dữ liệu được cải tiến bằng việc sử dụng khối dữ liệu đã được nạp trước, bao gồm việc đọc một mảng các block được chỉ bởi các ROWID.

Nếu không nạp trước khối (block) dữ liệu, việc truy cập một lượng lớn các dòng sử dụng thuần cấu trúc cây chỉ mục phân cụm B-tree có thể tốn kém. Mỗi dòng được truy cập bằng chỉ mục có thể ở các khối dữ liệu riêng biệt và do đó cần các thao tác vào ra riêng biệt.

Với việc nạp trước các khối dữ liệu, hệ thống trì hoãn khối dữ liệu đến khi các dòng được xác định bởi chỉ mục đã sẵn sàng để truy cập và sau đó phản hồi các khối dữ liệu đồng thời, hơn là đọc từng khối dữ liệu riêng biệt tại một thời điểm.

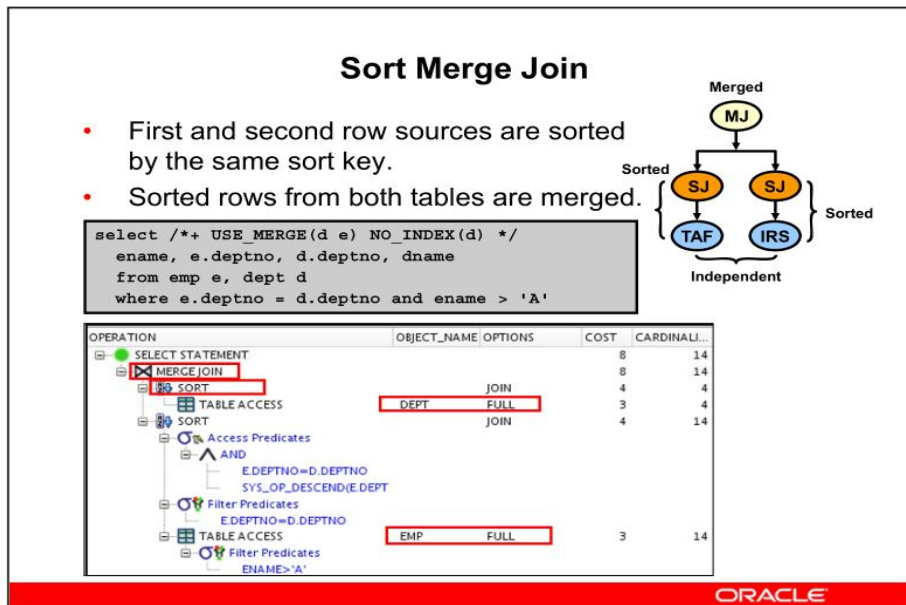
## 2.3 Nested Loops Join: 11g implementation



Oracle database 11g giới thiệu một cách mới để thực hiện kết nối bằng phương pháp Nested loops join. Cách Oracle database 11g thực hiện nested-loops join gọi là NESTED LOOPS implementation, ban đầu hệ thống thực hiện kết nối NESTED LOOPS giữa một bảng với một chỉ mục. Thủ tục này tạo ra một tập các ROWID cho phép bạn có thể sử dụng để tìm kiếm dòng tương ứng từ bảng với chỉ mục. Thay vì đi tới bảng cho mỗi ROWID được sinh ra bởi phép kết nối NESTED LOOPS thứ nhất, hệ thống sử lý một lô các ROWID và thực hiện phép kết nối NESTED LOOPS thứ hai giữa các ROWID và bảng. Kỹ thuật sử lý theo lô các ROWID cải thiện hiệu năng của hệ thống chỉ khi đọc từng khối trong bảng trong một lần.

### 3. Sort Merge Join

- Cả hai nguồn dữ liệu được sắp xếp theo cùng một khóa sắp xếp.
- Các dòng đã được sắp xếp từ hai bảng được trộn lại với nhau.



#### Sort Merge Join

Trong phương pháp nối sort merge join, không có khái niệm bảng điều khiển (bảng ngoài). Thuật toán nối được thực hiện như sau:

1. Lấy tập dữ liệu đầu tiên, sử dụng mọi truy cập và mọi điều kiện lọc, và sắp xếp trên cột nối (cột dùng để nối hai bảng).
2. Lấy tập dữ liệu thứ hai, sử dụng mọi truy cập và mọi điều kiện lọc, và sắp xếp trên cột nối.
3. Với mỗi dòng trong tập dữ liệu thứ nhất, tìm điểm bắt đầu trong tập dữ liệu thứ hai và duyệt đến khi bạn tìm thấy một dòng không để nối.

Thuật toán trộn:

```

get first row R1 from table 1
get first row R2 from table 2
while not at the end of either table
begin
  if R1.join_column = R2.join_column
  begin
    return (R1, R2)
    get next row R2 from table 2
  end
  else if R1.join_column < R2.join_column
  get next row R1 from table 1
  else
  get next row R2 from table 2
end
  
```

Thủ tục trộn tổ hợp hai nguồn dữ liệu đã sắp xếp để phản hồi mọi cặp dòng có chứa giá trị trùng khớp cho cột được sử dụng trong điều kiện nối.



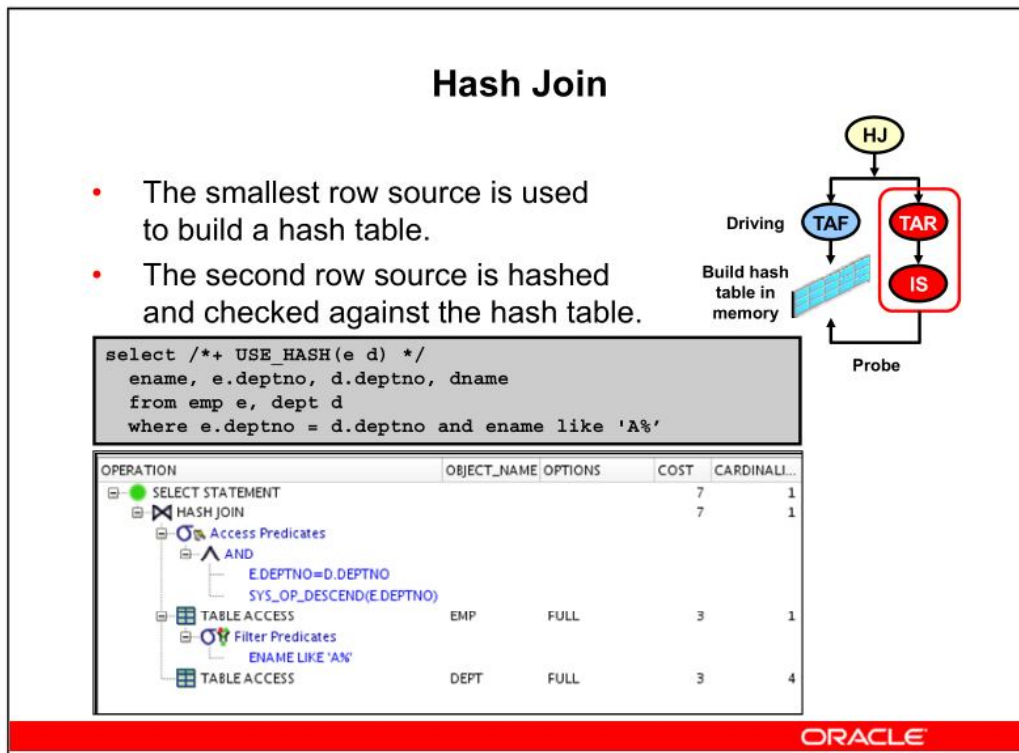
Nếu một nguồn dữ liệu đã được sắp xếp trong thao tác trước ( có một chỉ mục trên cột nối), thủ tục sắp xếp trộn sẽ dừng việc sắp xếp trên nguồn dữ liệu đó. Khi bạn thực hiện phép nối trộn, bạn phải nạp tất cả các dòng của hai nguồn dữ liệu trước khi trả về dòng đầu tiên và tới thao tác tiếp theo. Việc sắp xếp có thể làm phép nối tốn kém, đặc biệt là khi sắp xếp không thể thực hiện trên bộ nhớ chính.

Bộ tối ưu có thể lựa chọn phép nối kết hợp phép nối hàm băm khi cần nối lượng lớn dữ liệu nếu bất kỳ điều kiện nào dưới đây là đúng:

- Điều kiện kết nối giữa hai bảng không phải là kết nối bằng (equijoin).
- Sắp xếp được thực hiện bởi thao tác trước đó.

Chú ý: Phép nối trộn rất hữu ích khi điều kiện nối giữa hai bảng là một điều kiện không bằng ví dụ: <, <=, >, >=.

## 4 . Hash Join



### Hash Join

Để thực hiện một phép nối hàm băm giữa hai nguồn dữ liệu, hệ thống cần đọc tập dữ liệu đầu tiên và xây dựng một mảng các giỏ băm trong bộ nhớ. Một dòng thuộc vào một giỏ băm nếu số băm khớp với kết quả mà hệ thống thu được bằng cách áp dụng một hàm băm nội tại với cột nối hay các cột của của dòng.

Hệ thống bắt đầu đọc tập dữ liệu thứ hai, sử dụng cơ chế nào đó để lấy các dòng, và sử dụng cùng hàm băm với cột nối hay các cột để tính giá trị của giỏ băm liên quan. Sau đó hệ thống kiểm tra xem có dòng nào trong giỏ băm hay không. Thao tác này được tìm kiếm trên bảng băm.

Nếu có một số dòng nào đó trong giỏ băm liên quan, hệ thống thực hiện kiểm tra chính xác trên cột nối hay các cột xem nếu có sự trùng khớp không. Bất kỳ dòng trùng khớp lập tức được thông báo hay được chuyển tới bước tiếp theo trong kế hoạch thực hiện (execution plan).

Do đó, khi thực hiện một phép nối băm, bạn cần nạp tất cả các cột từ nguồn dữ liệu nhỏ nhất đến khi trả về dòng đầu tiên cho thao tác tiếp theo.

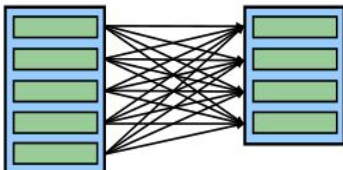
Thuật toán:

```
for each row R1 in the build table
begin
    calculate hash value on R1.join_column
    insert R1 into the appropriate hash bucket
end
for each row R2 in the probe table
begin
    calculate hash value on R2.join_column
    for each row R1 in the corresponding hash bucket
        if R1.join_column = R2.join_column
            return (R1, R2)
    end
```

Chú ý: Phép nối băm được thực hiện chỉ cho nối bằng(equijoin) và rất hữu ích khi nối dữ liệu có kích thước lớn.

## 5. Cartesian Join

**Cartesian Join**



```
select ename, e.deptno, d.deptno, dname
from emp e, dept d where ename like 'A%';
```

Explain Plan x

0.295 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			6	4
MERGE JOIN		CARTESIAN	6	4
TABLE ACCESS	EMP	FULL	3	1
Filter Predicates				
ENAME LIKE 'A%'				
BUFFER			3	4
TABLE ACCESS	DEPT	FULL	3	4

**ORACLE**

## Cartesian Join

Phép tích đề-các được sử dụng khi nối hai hay nhiều bảng mà không có điều kiện nối.

Bộ tối ưu nối mọi dòng từ một nguồn dữ liệu với mọi dòng của nguồn dữ liệu khác, tạo nên tích đề - các giữa hai tập.

Một phép kết nối theo tích Descartes có thể xem như là một vòng lặp lồng nhau không có loại bỏ. Tập dữ liệu đầu tiên được đọc lần lượt (coi như lặp lại việc đọc) và với mỗi dòng của nó lại được kết nối với toàn bộ các dòng của bảng kia. Lặp đi lặp lại việc kết nối đó một cách đầy đủ cho đến khi các dòng ở bảng đầu tiên được đọc hết.

Lưu ý: Nói chung, việc kết nối theo tích Descartes là không được mong muốn.

## 6. Join Type

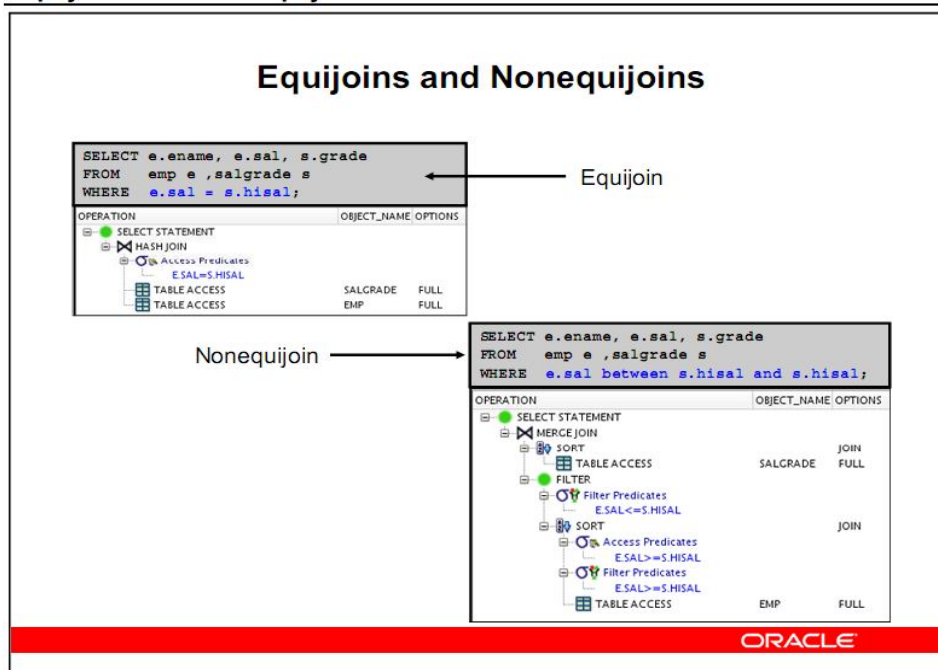
- Kết nối từ hai cột thuộc 2 nguồn dữ liệu và kết quả trả về là một cột
- Các kiểu kết nối bao gồm:
  - + Join (Equijoin/Natural – Nonequijoin)
  - + Outer join (Full, Left, and Right)
  - + Semi join: EXISTS subquery
  - + Anti join: NOT IN subquery
  - + Star join (Optimization)

Các kiểu kết nối bao gồm:

- Join (equijoin and nonequijoin): Trả về các hàng phù hợp với vị từ tham gia
- Outer join: Trả về các hàng các hàng phù hợp với vị từ tham gia và hàng không phù hợp được tìm thấy
- Semi join: Trả lại các hàng phù hợp với các truy vấn con EXISTS. Tìm một hàng phù hợp trong bảng sau đó dừng lại
- Anti join: Trả lại hàng không phù hợp trong các truy vấn con NOT IN. Dừng lại ngay lập tức khi tìm thấy kết quả
- Star join: Đây không phải là một kiểu kết nối, nó là một cái nhãn cho việc thực hiện xử lý tối ưu hiệu năng, làm việc tốt hơn trong điều khiển thực tế và các mô hình

Anti Join and Semi Join được xem là các loại kết nối, mặc dù các truy vấn con được xây dựng dựa trên cấu trúc của SQL. Anti Join và Semi Join là các thuật toán tối ưu được sử dụng để làm phẳng các truy vấn con theo cách mà họ có thể giải quyết được trong một cách kết nối phù hợp

## 6.1 Equijoin and Nonequijoin



Điều kiện kết nối sẽ xác định liệu một kết nối là một equijoin hay một nonequijoin. Một equijoin là một kiểu kết nối với điều kiện kết nối chứa một toán tử bình đẳng. Khi một điều kiện kết nối liên quan tới hai bảng bởi một toán tử khác hơn sự bình đẳng, nó là một nonequijoin.

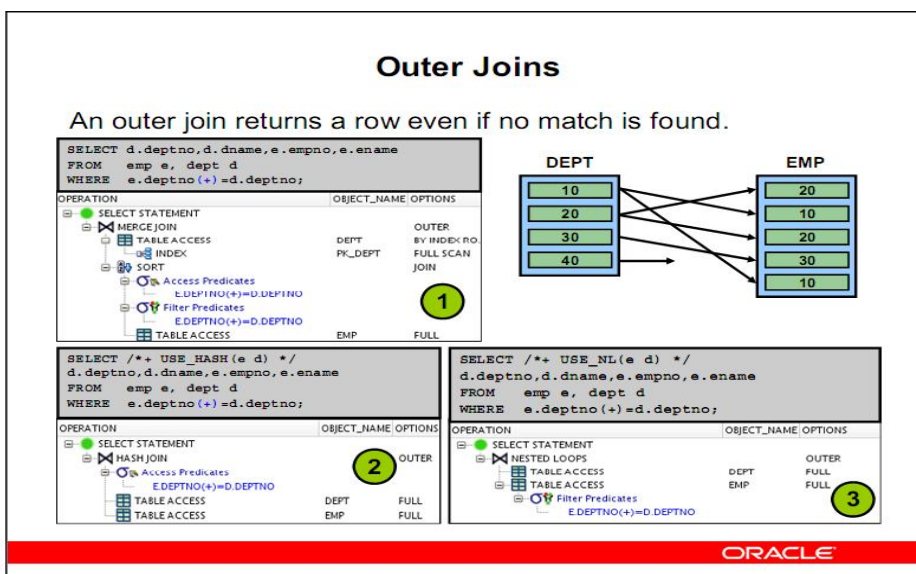
Equijoin là được sử dụng phổ biến nhất. Một ví dụ với mỗi một equijoin và một nonequijoin được thể hiện trong các slide. Nonequijoins ít được sử dụng

Đề nâng cao hiệu quả SQL, sử dụng equijoins bất cứ khi nào có thể. Câu lệnh thực hiện các equijoins trên các giá trị của cột là dễ dàng điều chỉnh nhất

Lưu ý:

Nếu bạn có một nonequijoin, một hash join là không thể.

## 6.2 Outer Join



Các kiểu kết nối đơn giản được sử dụng phổ biến trong hệ thống. Các kiểu kết nối khác mở rộng các chức năng bổ sung, nhưng được sử dụng nhiều hơn trong chuyên ngành

The outer join operator được đặt ở bên thiếu của các truy vấn. Nói cách khác, nó được đặt vào bảng mà có các thông tin kết nối bị thiếu.

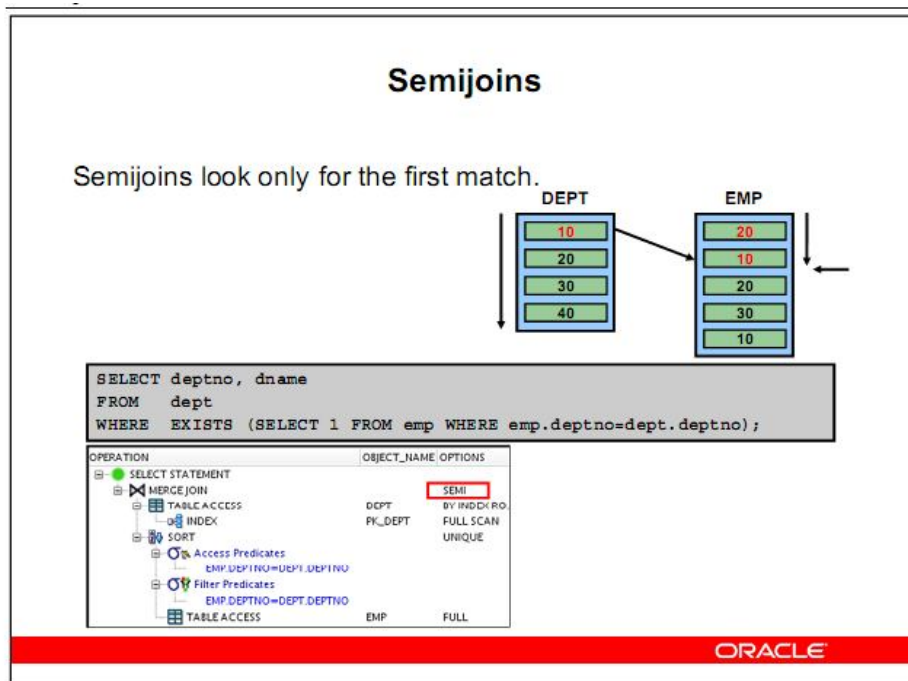
Xem xét EMP và DEPT. Có thể có một bộ phận không có nhân viên. Nếu EMP và DEPT được kết nối với nhau, thành phần đặc biệt này sẽ không xuất hiện ở đầu ra vì không có hàng phù hợp với điều kiện tham gia cho bộ phận nhân viên đó. Bằng việc sử dụng Outer Join, các bộ phận thiếu sẽ được hiển thị

1. Merge Outer joins: Theo mặc định, tối ưu việc sử dụng MERGE OUTER JOIN

2. Outer join with nested loops: Các left/driving bảng luôn luôn là các bảng mà hàng đang được lưu trữ (DEPT trong ví dụ). Đối với mỗi hàng từ DEPT, tìm kiếm tất cả các hàng phù hợp trong EMP. Nếu không có hàng được tìm thấy, giá trị đầu ra DEPT là null cho cột EMP. Nếu hàng được tìm thấy, giá trị đầu ra của DEPT là giá trị của EMP

3. Hash Outer joins: The left/outer bảng mà hàng đang được lưu trữ được sử dụng để xây dựng bảng băm và the right/inner được sử dụng để thăm dò các bảng băm. Khi tìm được kết quả phù hợp, dòng là đầu ra và các entry trong bảng băm được đánh dấu phù hợp với một hàng. Sau khi, các bảng băm được đọc lại một lần nữa và bất kỳ hàng nào không được đánh dấu như là đầu ra giá trị cho cột EMP. Hệ thống bảng băm mà hàng không được lưu trữ và sau đó đọc các bảng mà hàng đang được lưu trữ, thăm dò các bảng băm để xem có hay không một hàng để tham gia kết nối.

### 6.3 Semijoins



Phép nửa kết nối đưa ra kết quả khi đạt được bản ghi đầu tiên thỏa mãn. Một phép nửa kết nối cung cấp một phương pháp biến một câu truy vấn thứ cấp hiện hữu (Exists subquery) trở thành một phép kết nối một cách nội bộ.

Phép nửa kết nối trả lại các dòng tương ứng với một câu truy vấn thứ cấp mà không lặp lại các dòng ở bảng phía bên trái câu truy vấn khi mà có đồng thời nhiều dòng ở bảng phải truy vấn đáp ứng các tiêu chí của câu truy vấn thứ cấp.

Ví dụ, trong sơ đồ trên, với mỗi bản ghi DEPT, chỉ bản ghi tương ứng đầu tiên của bảng EMP là được trả về

trong kết quả kết nối hai bảng. Điều này ngăn cản việc quét với số lượng lớn các bản trùng lặp trong một bảng khi mà tất cả những gì bạn quan tâm chỉ là một bản ghi nào đó phù hợp là đủ.

Khi mà một truy vấn thứ cấp là lồng nhau(phức tạp), có thể đạt được kết quả tương tự bằng cách sử dụng một phép lọc và quét các dòng cho tới khi tìm thấy được bản ghi tương ứng và trả về bản ghi đó.

Chú ý: Một phép nối kết nối có thể luôn luôn sử dụng một kết nối hợp. Việc tối ưu hóa có thể lựa chọn một phương thức nested-loop, hash-join để phép nối kết nối thực hiện tốt

## 6.4 Anjoins

Antijoins trả về các dòng mà không so khớp (NOT IN) với câu truy vấn con ở bên phải. Ví dụ một antijoin có thể chọn một danh sách các phòng mà phòng đó không có bất kỳ nhân viên nào.

Bộ tối ưu sử dụng một giải thuật merge anjoin

### Antijoins

