# Join Methods

.

## Task 2: Nested Loop Join

Nested Loop (NL) – метод объединения данных вложенными циклами. NL используется в случаях когда:

* два маленьких датасета;
* датасеты большие, но используется когда используется подсказка FIRST\_ROWS;
* условие соединения таблиц является очень эффективным по доступу к внутренней таблице (большей по количеству строк).

Лучше всего работают на небольших таблицах с индексами в условиях соединения. Оптимизатор всегда пытается сначала поместить наименьший источник строк, делая его ведущей таблицей. Получается, что Oracle берет строку из ведущей таблицы (наименьшей по количеству строк) и с ней идет во вторую таблицу, которую в поиске соответствующих данных проходит по циклу.

В нашем примере, Oracle находит в таблице departments строку по rowid из индекса. 1 строка, потому что это PK, соответственно и использовался Unique scan. С этой одной строкой он идет в таблицу employees и проходит циклом по ней в поиске соответствия.

select \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id

and d.department\_id = 10;



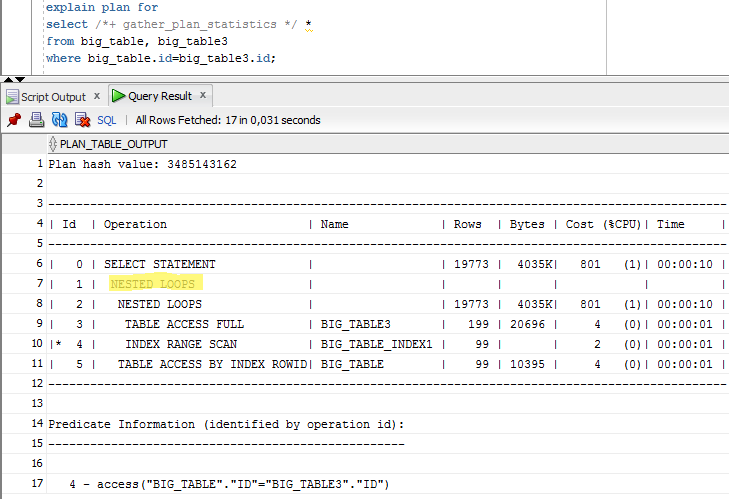
\*Пункт 1 таблицы задания 8

При соединении большой таблицы с индексированным ключом и маленькой, Oracle выбирает NL.

select /\*+ gather\_plan\_statistics \*/ \*

from big\_table, big\_table3

where big\_table.id=big\_table3.id;



\*Пункт 7 таблицы задания 8

## Task 3: Sort-Merge Join

Можно сказать, что Merge Join (MJ) является частным случаем NL. Важно, что у него нет понятия внешней и внутренней таблицы, обе равны. Он считывает две таблицы (уже на этот момент отобраны только те данные, которые соответствуют условиям в where), сортирует эти датасеты по ключу соединения и затем объединяет наборы строк. Происходит чередование двух датасетов. Для каждой строки в первом датасете Oracle проверяет второй набор данных для сопоставления строк и объединяет их. Важным является то, что при переходе из датасета в датасет он основывает свою начальную позицию на совпадении, сделанном в предыдущей итерации. Основные ресурсы операции затрачиваются именно при сортировке, непосредственно сам MJ происходит быстро. Часто очень неэффективный алгоритм именно из-за потенциально больших накладных расходов на сортировку данных перед выполнением соединения.

В первом примере, он берет два множества данных, одно сортирует, другое берет из таблицы, обращаясь к ней по rowid из индекса (сортировка не нужна, потому что индекс уже упорядочен), и потом делает MJ. Размеры датасетов соизмеримы, считаю, что в этом тоже причина выбора MJ.

select \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id;



\* Пункт 2 таблицы из задания 8

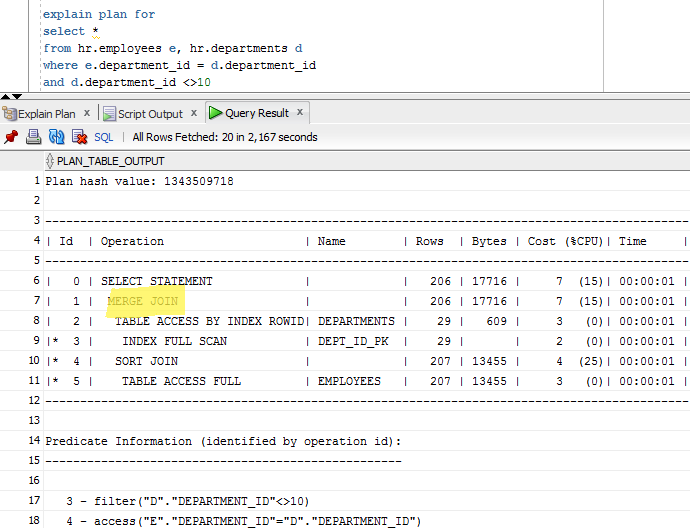
Во втором примере запрос аналогичный, но вводится условие неравенства на индексированное поле department\_id значению 10. Размеры таблиц все равно остаются соизмеримы, поэтому MJ.

select \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id

and d.department\_id <>10;



\* Пункт 3 таблицы из задания 8

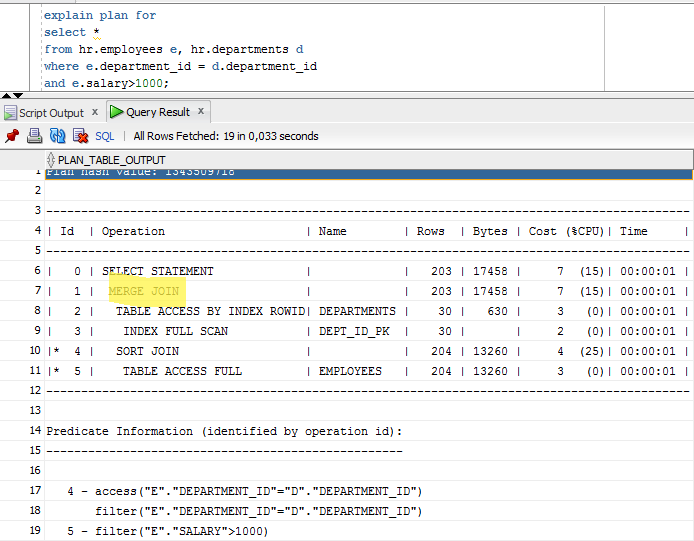
Во втором примере запрос аналогичный, но вводится условие неравенства на индексированное поле department\_id значению 10. Размеры таблиц все равно остаются соизмеримы, поэтому MJ.

select \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id

and e.salary>1000;



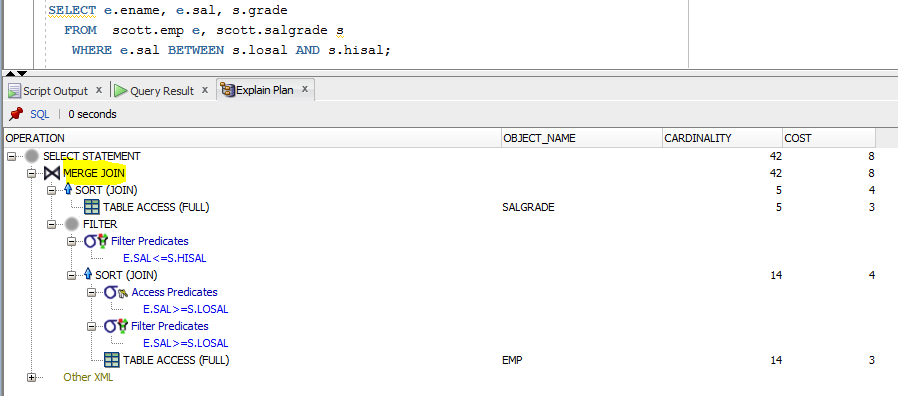
\* Пункт 3 таблицы из задания 8

Во втором случае у таблиц хоть и нет соединения по ключу между собой, но Oracle использует MJ для того, чтобы проверить все данные из таблицы emp на соответствие условию диапазона зарплат из другой таблицы. Делает Full Scan таблиц, потом сортирует их и ищет записи методом “ёлочки”. Т.е. в данном случае никогда не будет Hash Joins.

SELECT e.ename, e.sal, s.grade

FROM scott.emp e, scott.salgrade s

WHERE e.sal BETWEEN s.losal AND s.hisal;



\* Пункт 4 таблицы из задания 8

Пример для сравнения:

select \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id

and e.salary>12008; -- по условию 6 записей в емплоях и 30 в департаментах

и

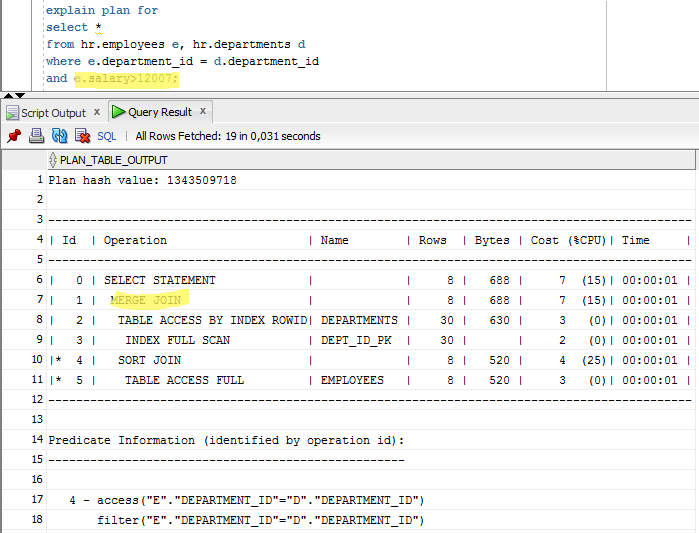
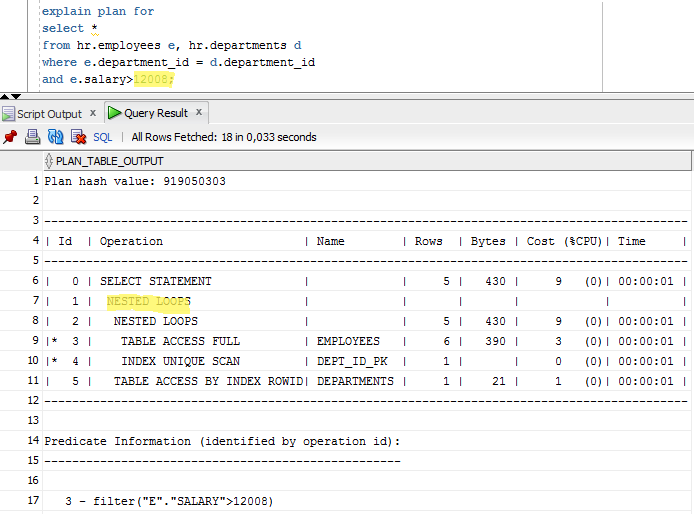
select \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id

and e.salary>12007; -- по условию 8 записей в емплоях и 30 в департаментах

В одном случае идет MJ, в другом NL - могу сделать вывод, что в при определенных условиях количество записей превышает определенный порог, когда Oracle, соизмеряя размеры таблиц, выбирает MJ (примерно одинаковые таблицы) или NL (одна поменьше и с ее записями можно ходить по циклу в большей).

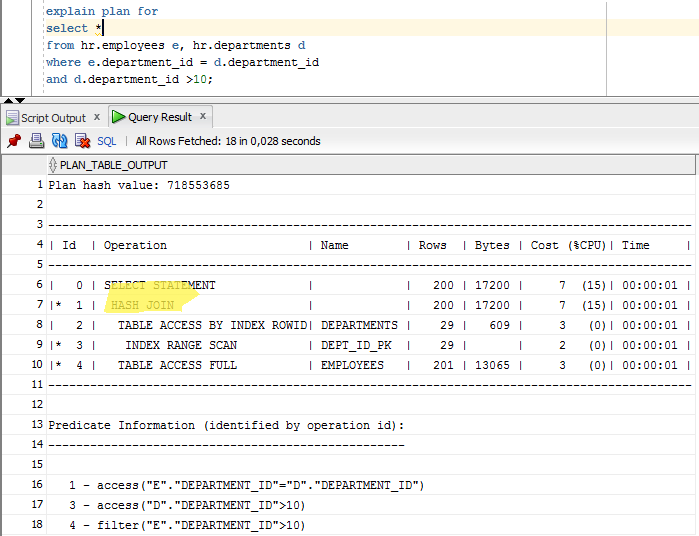
 

## 

## Task 4: Hash Join

Для Hash Join (HJ) важно наличие объединения большого объема данных (или большой части небольшого датасетов) и эквисоединения. При HJ меньшая из двух таблиц хэшируется в памяти, дальше берется вторая таблица и обращается к хэшированным значениям первой таблицы. Важно понимать, что вторая таблица не сортируется, но читается только 1 раз. Hash Join является самым быстрым алгоритмом соединения данных из нескольких источников за счет хэширования в буфер маленькой таблицы, однако он не может быть использован для любых условий соединений.

В нашем примере, есть эквисоединение, и условие отбора (>10) данных по индексируемому полю.



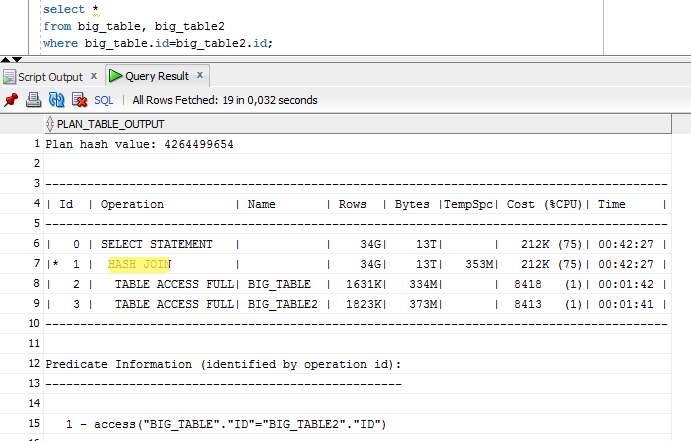
\* Пункт 5 таблицы из задания 8

При соединении больших/средних таблиц по неиндексируемым столбцам, Oracle выбрал Hash Join.

select \*

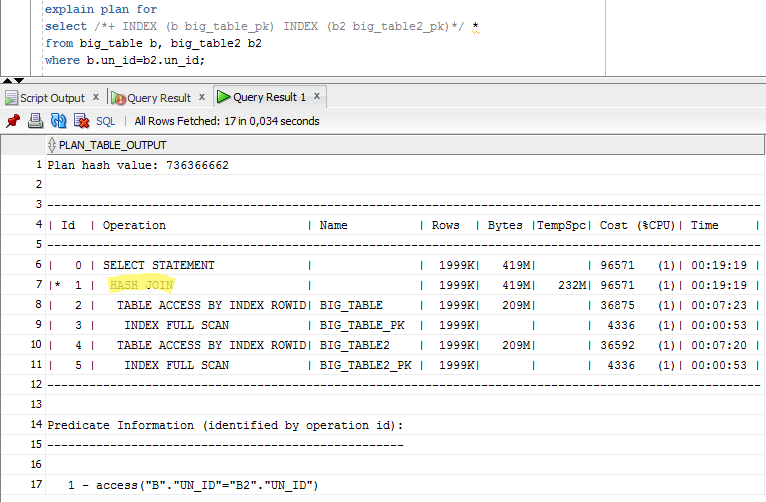
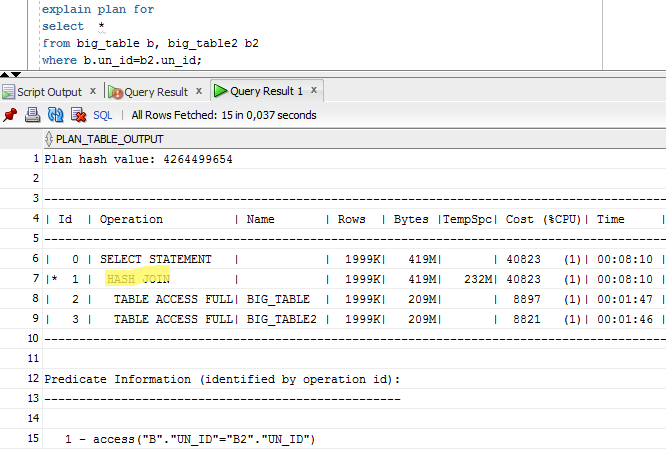
from big\_table, big\_table2

where big\_table.id=big\_table2.id;



\* Пункт 6 таблицы из задания 8

При соединении больших/средних таблиц по индексируемым столбцам одинаковых размеров, Oracle тоже выбрал Hash Join. При этом он не использовал индексы, и даже если заставить его искать по ним данные, в соединении ничего не меняется.

\* Пункт 10 таблицы из задания 8

## Task 5: Cartesian Join

Oracle данных использует Cartesian Join (CJ) или декартово объединение, когда одна или несколько таблиц не имеют каких-либо условий соединения для любых других таблиц в запросе. Оптимизатор соединяет каждую строку одного датасета с каждой строкой другого датасета, создавая декартово произведение двух наборов.

Хинт Ordered задает порядок в котором оракл будет брать таблицы. Поскольку вначале оракл берет меньшую таблицу и к ней присоединяет большую, то в нашем примере мы его заставляет взять вначале большую, а затем маленькую. Он считает, что в данном случае оптимально взять для соединения MERGE JOIN CARTESIAN.

select /\*+ORDERED \*/ \*

from hr.employees e, hr.departments d

where e.department\_id = d.department\_id

and d.department\_id = 10;

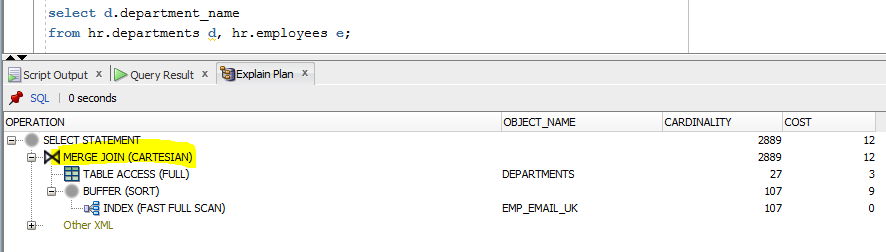


\* Пункт 8 таблицы из задания 8

В запросах, где нет условия where для соединения, Oracle соединяет каждую строку первой таблицы с каждой строкой второй таблицы, выдавая тем самым MERGE JOIN CARTESIAN.

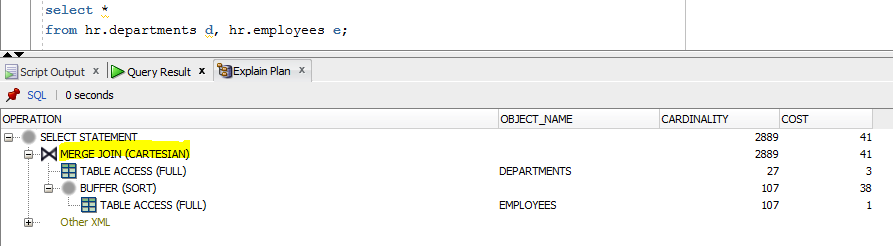
select d.department\_name

from hr.departments d, hr.employees e;



select \*

from hr.departments d, hr.employees e;



\* Пункт 9 таблицы из задания 8

## Task 6: Left/Right Outer Joins

Левое/правое внешнее объединение возвращает все строки из одной таблицы, и только те строки из объединенной таблицы, где выполняется условие объединения, там где нет соответствия заполняются Null. В нашем примере left join - в employees 107 записей, в departments – 22 записи, из них только 106 являются соединяемыми между собой. Выводится 107 записей. Во втором примере right join - в employees 107 записей, в departments – 22 записи, из них только 106 являются общими. Выводится 122 записи, потому что к 106 общим добавляются 16 из departments, которым нет соответствий в employees.

**Left join:**

ANSI syntax:

select \* --107 записей

from hr.employees e left join hr.departments d on e.department\_id=d.department\_id;

Oracle syntax (+):

select \* -- 107 записей

from hr.employees e, hr.departments d

where e.department\_id=d.department\_id(+);

**Right join:**

ANSI syntax:

select \* -- 122 записей

from hr.employees e right join hr.departments d on e.department\_id=d.department\_id;

Oracle syntax (+):

select \* -- 122 записей

from hr.employees e, hr.departments d

where e.department\_id(+)=d.department\_id;

## Task 7: Full Outer Join

Полное внешнее соединение объединяет две таблицы слева направо и справа налево. Записи, которые есть в двух таблицах, выводятся один раз, чтобы избежать дублирования. Также выводятся записи из двух таблиц, которым нет эквивалента по соединяемым полям в другой таблице.

В нашем примере в employees 107 записей, в departments – 22 записи, из них 106 соединяются по ключу (по сути inner join), 1 запись из employees, у которой нет департамента, и 16 записей из departments, у которых нет сотрудников. Итого, 106+1+16 = 123 записи.

ANSI syntax:

select \* -- 123 записи

from hr.employees e full join hr.departments d on e.department\_id=d.department\_id;

Oracle syntax (+):

select \* -- 123 записи

from hr.employees e, hr.departments d

where e.department\_id(+)=d.department\_id

union

select \*

from hr.employees e, hr.departments d

where e.department\_id=d.department\_id(+);

## Task 8: Results

|  |  |  |  |
| --- | --- | --- | --- |
|  | Table “A” | Table “B” | Join type description |
| 1 | Небольшая таблица с индексом по ключу | Небольшая таблица с индексом по ключу | Соединение по индексированным ключам + условие равенства столбца (неважно ключ это, индексированное поле или нет) конкретному значению получился случай, когда таблица В (1 строка) значительно больше А (107 строк) = Nested Loop |
| 2 | Таблица с индексом по ключу | Таблица с индексом по ключу | Соединение по ключам, таблицы примерно одинаковые по размеру = Merge Join |
| 3 | Таблица с индексом по ключу | Таблица с индексом по ключу | Соединение по ключам + условие неравенства столбца (неважно ключ это, индексированное поле или нет) конкретному значению, таблицы остаются примерно одинаковыми по размеру = Merge Join |
| 4 | Таблица (нет соединения по ключам или индексированным полям) | Таблица (нет соединения по ключам или индексированным полям) | Соединение по ключу отсутствует, и неиндексированный столбец проверяется на <= и <= с столбцами другой таблицы (интерпретация between) = тут вообще без шансов – только Merge Join |
| 5 | Таблица с индексом по ключу | Таблица с индексом по ключу | Соединение по ключу и индексированный столбец проверяется на > числа, небольшая таблица выбирается для хеширования – Hash Join |
| 6 | Средняя таблица без индексированного ключа | Средняя таблица без индексированного ключа | Соединение в данном случае лучше производить хешированием одной из таблиц, т.к. сортировка слишком дорогая операция – Hash Join |
| 7 | Большая таблица с индексированным ключом | Маленькая таблица без индексированного ключа | Как и во многих случаях, когда имеет место большая и маленькая таблица, используется = Nested Loop |
| 8 | Средняя таблица с индексированным ключом | Маленькая таблица с индексированным ключом | С хинтом /\* + ORDERED / Oracle берет большую таблицу, а затем маленькую. Он понимает, что не выполняется вариант NL и выбирает вместо него MJ CARTESIAN. |
| 9 | Таблица любого размера с любым ключом | Таблица любого размера с любым ключом | Без условия соединения всегда будет выдаваться MJ CARTESIAN. |
| 10 | Большая таблица с индексированным ключом | Большая таблица с индексированным ключом | Соединение в данном случае лучше производить хешированием одной из таблиц, т.к. сортировка слишком дорогая операция – Hash Join |

Делая вывод из проделанной работы, могу сказать, что использование типов джойнов зависит от размера объединяемых данных, от размера относительно друг друга и от параметров соединения столбцов (эквисоединение, или неравенство, индексируемый ли столбец).