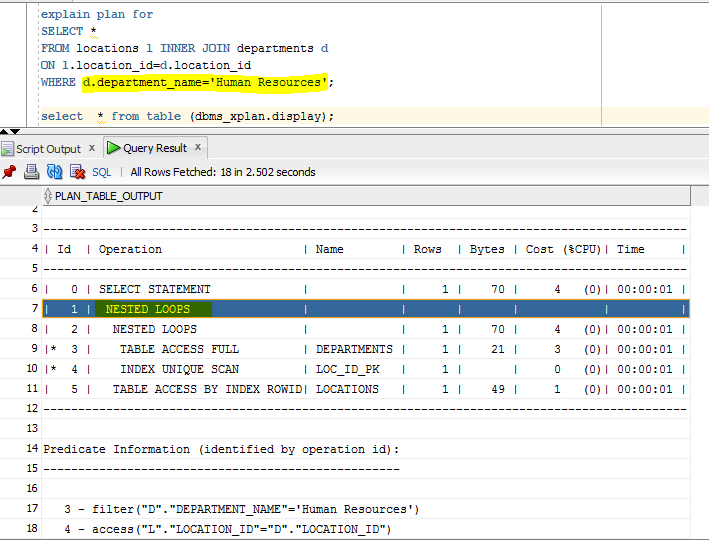
# Join Methods

Для выполнения данного задания, я буду использовать таблицы из схемы HR.

## Task 1: Nested Loop Join

Пример выполнения запроса, в котором происходит nested loop join.



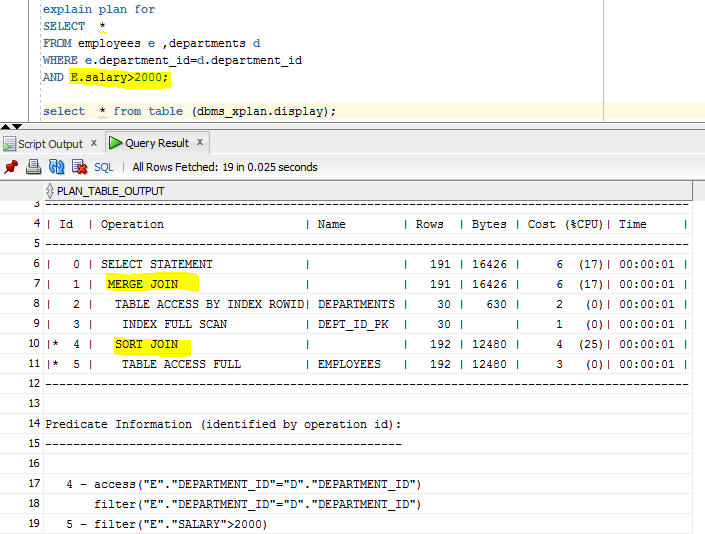
Nested Loop Join работает при выполнении условий:

* Два небольших набора данных
* Наборы данные большие, но присутствует условие, при котором выводится одна строка ( FIRST\_ROWS)

Оптимизатор делает ведущим источник, с наименьшим количеством строк. Затем он берет строку из этой таблицы и с ней проходит вторую таблицу до совпадения.

## Task 3: Sort-Merge Join

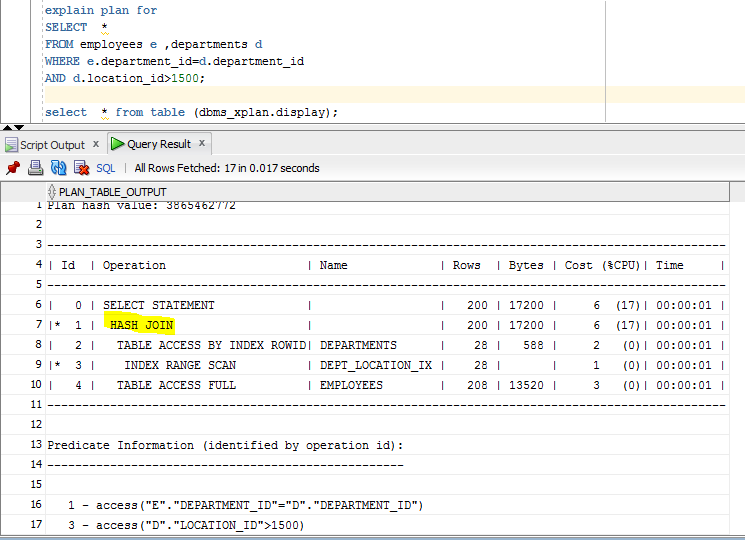
Для следующего задания, будем использовать другой запрос, представленный ниже:



Можно заметить, что оптимизатор предварительно применил фильтр по зарплате и условию соединения, отсортировал их по ключу соединения и потом применил join. Т.е. oracle для каждой строки одной таблицы, проверяет второй набор данных на соответствие ключу соединения и затем объединяет записи.

## Task 4: Hash Join

Для просмотра результата hash join применим следующий запрос:



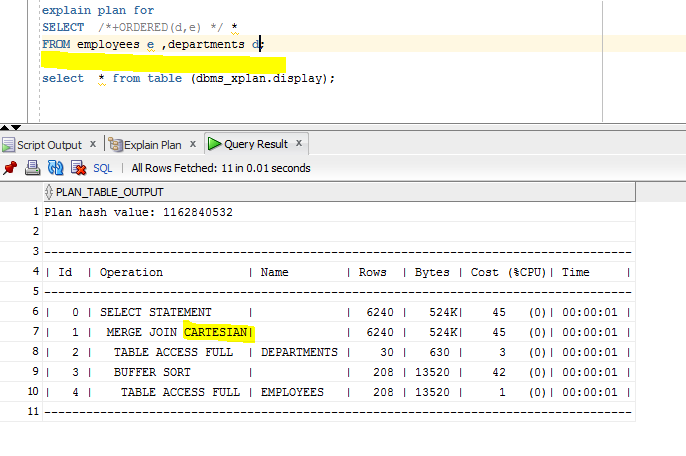
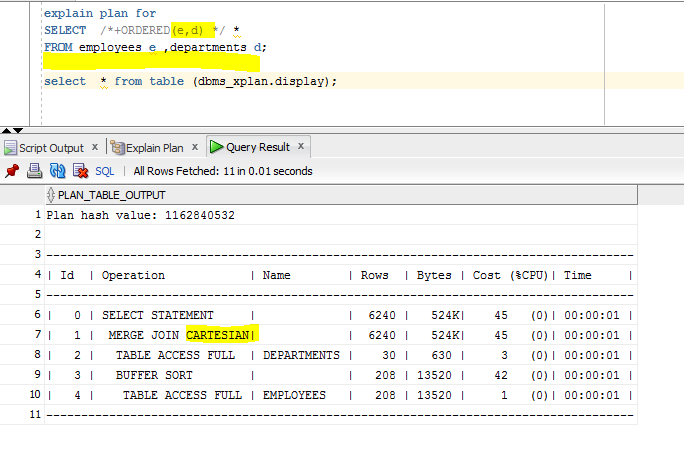
Идея применения hash join заключается в том, что значение ключа МАЛЕНЬКОЙ таблицы заносится в хэш и помещается в буфер (в нашем случае это таблица departmnets). Затем oracle брал вторую таблицу и обращался к хэшированным значениям первой-маленькой таблицы и сравнивал по условию хэш-ключей таблицы, если значение совпадают,то join таблиц выполняется, если нет, то нет. Главное преимущество это то, что значения большей таблицы не сортируется, т.е. мы выигрываем в performance (сортировка таблицы дорогая операция для oracle), и большую таблицу мы просматриваем один раз.

## Task 5: Cartesian Join

Cartesian Join – декартовое произведение таблиц без условий соединения. Оптимизатор соединяет каждую строку одной таблицы со всеми строками из другой таблицы. Таким образом получается результат выполнения Cartesian join.

* ORDERED hint and wrong predicate order.

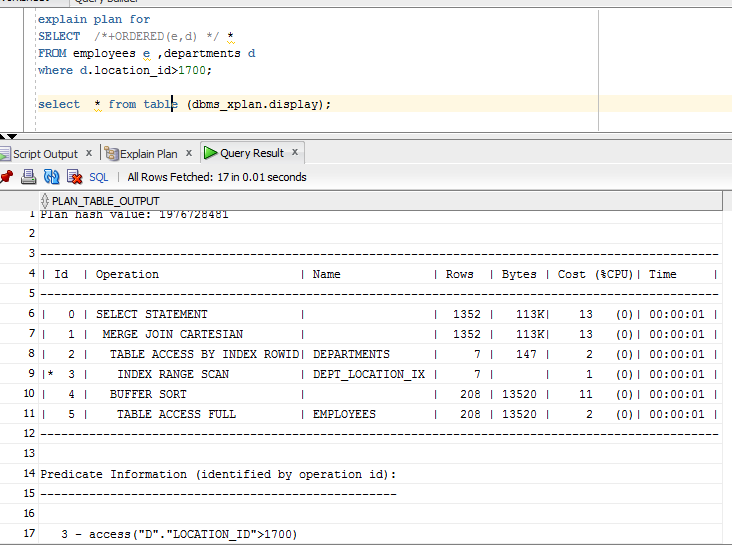
Применение hint ORDERED с правильным и неправильным предикатом



Можно заметить, что Oracle игнорирует правильность предикатов, и соединяет их так, как ему удобно, беря маленькую таблицу за ведущую.

* W/o join conditions.

C использование условия:



Сравним 2 применения Cartesian Join один с использованием условия (это не условие присоединения), второй без условия.

Можно заметить при выполнении условия, количество строк заметно уменьшается, но оно все равно велико из-за декартового произведения. Видно, что при применении условия, одна из таблиц (departments) сканируется по индексу, выбираются нужные строки, и только затем происходит join.

## Task 6: Left/Right Outer Joins

Данные типы join возвращают нам dataset, который состоит из объединенной таблицы (где условие выполняется), а так же строки из той таблицы, тип join который мы используем, где условие не выполняется, заполняются Null.

Implement Left/Right outer joins with:

Left Join. Таблица employees содержит 107 записей, таблица departments 22 записей. При применении inner join получаем 106 записей, т.е. один сотрудник не имеет департамента. При выполнении left join к employees получаем 107 записей, т.к. общее количество сотрудников – 107, т.е. то, что нам надо.

* ANSI syntax (left join)

**SELECT \***

**FROM employees e left join departments d on e.department\_id=d.department\_id;**

* Oracle syntax ( left +)

**SELECT \***

**FROM employees e, departments d**

**WHERE e.department\_id=d.department\_id(+);**

Right Join. Таблица employees содержит 107 записей, таблица departments 22 записей. При применении inner join получаем 106 записей, т.е. один сотрудник не имеет департамента. Количество департаментов, в которых есть сотрудники равняется 6, т.е. еще 16 департаментов без сотрудников. При выполнении right join к employees получаем 122 записей, т.к. количество сотрудников, имеющих департаменты – 106 + 16 департаментов без сотрудников.

* ANSI syntax (right join)

**SELECT \***

**FROM employees e right join departments d on e.department\_id=d.department\_id;**

* Oracle syntax (right +)

**SELECT \***

**FROM employees e, departments d**

**WHERE e.department\_id(+)=d.department\_id;**

## Task 7: Full Outer Join

Полное внешнее соединение объединяет записи двух таблиц целиком, т.е. сюда попадают записи, которые имеют соединения, а также записи, которые не имеют соединения, там будет стоять Null.

Для нашего примера таблица employees имеет 107 записей, departments 22, 106 сотрудников имеют департаменты, 6 department имеют сотрудников.

Результирующее количество будет следующим:

106(соединение по ключу)+1(сотрудник без департамента)+(22-6)(количество департаментов не имеющих сотрудников)=123 записи.

* ANSI syntax (outer join)

**SELECT \***

**FROM employees e full join departments d on e.department\_id=d.department\_id;**

* Oracle syntax (+)

**SELECT \***

**FROM employees e, departments d**

**WHERE e.department\_id=d.department\_id(+);**

**UNION**

**SELECT \***

**FROM employees e, departments d**

**WHERE e.department\_id(+)=d.department\_id;**

## Task 8: Results

* Make a list of combination of different tables (see examples below) and write a description for the join between them. Express your opinion on why the selected join type was chosen by an optimizer. Create at least 10 variants.

|  |  |  |
| --- | --- | --- |
| Table “A” | Table “B” | Join type description |
| Небольшая таблица с индексом по ключу | Небольшая таблица с индексом по ключу | Будет Nested Loop, если используется соединение по ключу, у которого есть индекс, и условие равенства любому значению из любого столбца (т.е. будет на выходе одна строка) |
| Таблица с индексом по ключу | Таблица с индексом по ключу | Таблицы имеют примерно одинаковое количество строк, и имеют индексы на ключах, будет использоваться Merge Join, т.к. индекс уже хранит сортировку по ключу и Oracle останется только сделать merge, а он не сильно дорогой. |
| Небольшая таблицы с индексом на поле | IOT | Был примене Merge join, т.к. 2 таблицы имеют индексы => они отсртированы, осталось только применить Merge |
| Небольшая таблица без индекса | Небольшая таблица с индексом на поле | Тут используется Hash join, т.к. тут не указано условие равенства значения поля (искл. Nested Loop), а так же одна из таблиц не имеет индекса, т.е. не отсортирована, а вторая имеет и отсортирована. В Этом случае применится Hash Join. |
| Большая таблица с индексом на поле | Большая таблица без индекса на поле | Hash Join, такая же ситуация как и описана выше с сортировкой таблиц, да и еще большое количество возвращаемых значений. |
| Большая таблица с равенством по значению поля | Небольшая таблица с индексом по полю | Nested Loop, т.к. из большой таблицы будет возвращены определенные строки, с помощью которых мы будем ходить по маленькой таблице |
| Любая таблица с наличием индексов или нет | Любая таблица с наличием индексов или нет | Без условия применения будет выдаваться Cartesian Join |
| Большая таблица с индексом на ключе | Маленькая таблица без индекса | Будет использоваться Nested Loop, при выполнении условия равенства в большой таблицу |