|  |
| --- |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Anastasiya](mailto:Kiryl_Bucha@epam.com) Khilko | 03-NOV-2017 |  |  |
|  |  |  |  |  |  |

Contents

[1. Table access 3](#_Toc497518942)

[1.1. Task 1: Full Scan, High-Water Mark and Consistent Gets 3](#_Toc497518943)

[1.1.1. Task Result 3](#_Toc497518944)

[2. Index Scan types 3](#_Toc497518945)

[2.1. Task 2: Index Clustering Factor 3](#_Toc497518946)

[2.1.1. Task Result 3](#_Toc497518947)

[2.2. Task 3: Index Unique Scan 4](#_Toc497518948)

[2.2.1. Task Result 4](#_Toc497518949)

[2.3. Task 4: Index Range Scan 5](#_Toc497518950)

[2.3.1. Task Result 5](#_Toc497518951)

[2.4. Task 5: Index Skip Scan 5](#_Toc497518952)

[2.4.1. Task Result 5](#_Toc497518953)

# Table access

## Task 1: Full Scan, High-Water Mark and Consistent Gets

### Task Result

Возникли проблемы с пониманием что такое Consistent gets. То, что удалось понять, прочитав несколько источников: это количество логических чтений блока. К одному блоку могут быть несколько чтений.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 1664 | 1536 | 99,999 | 1539 | Создали таблицу и индекс на ней. HWM на позиции последнего записанного блока. |
| 2 | 1664 | 0 | 0 | 1541 | Затем удаляем все строки из таблицы. Делаем commit. Consistent gets увеличился, так как процесс удаления строк затронул согласованный механизм чтения. До срабатывания autotrace HWM должен оставаться на той же позиции, что и на предыдущем этапе. После – изменить свою позицию. |
| 3 | 1664 | 1 | 1 | 1541 | Вставка одной строки. HWM не изменил своей позиции |
| 4 | 8 | 0 | 0 | 3 | Выполнили Truncate, который является ddl операцией. Блоки очищены. 3 оставшихся блока – метаданные. |

# Index Scan types

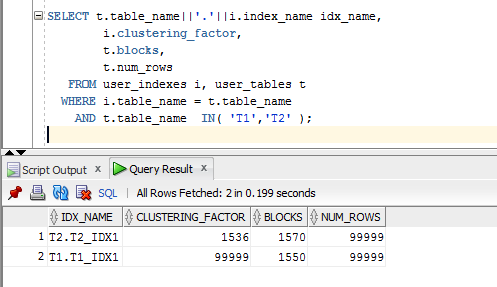
## Task 2: Index Clustering Factor

### Task Result

Фактор кластеризации – мера того, как организованы данные в сравнении с индексным столбцом.

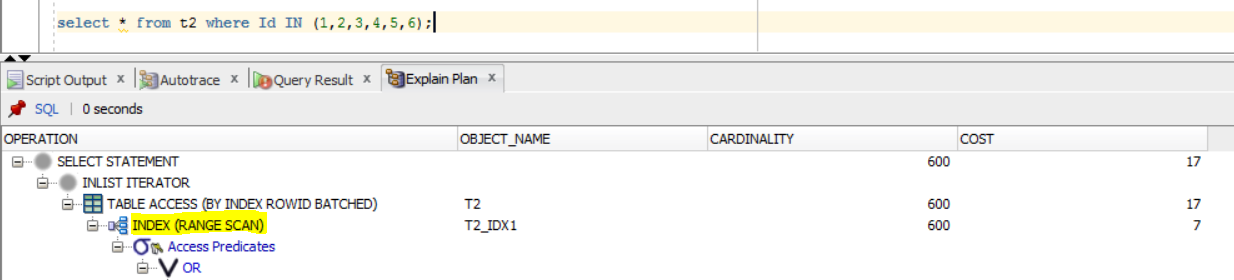
Минимальное значение фактора кластеризации всегда равно количеству блоков, максимальное – количеству строк.   
Причина, почему отличаются значения. В таблице T2 индекс отсортирован, таким образом данные объединены в блоки, где значение индекса одинаково. При запросе select \* from T2 where t\_pad = 1(t\_pad – поле, на котором построен индекс) не придется ходить по всем блокам, чтобы найти необходимые значения.

В случае второй таблицы T1 индекс неупорядочен, поэтому данные разбросаны по блокам с разными значениями индекса. Что увеличит в разы количество чтений из таблицы.

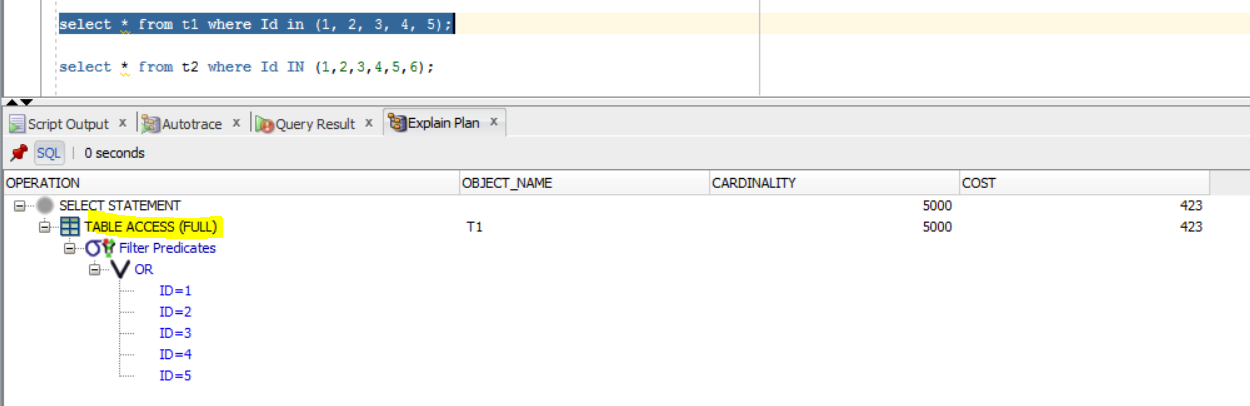


Which Index has better performance while executing **select** clause filtered by **IN ( list of values )?**

В таблице T2 индекс упорядочен, выражение IN обеспечивает операции сравнения найденных значений со значениями из диапазона, указанных в IN. Задействован range scan. Такой запрос отработает лучше.



В таблице T1 индекс неупорядочен. Оптимизатор использует полный просмотр таблицы, так как ему неэффективно подключать индексный метод доступа.

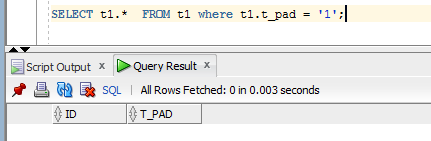


## Task 3: Index Unique Scan

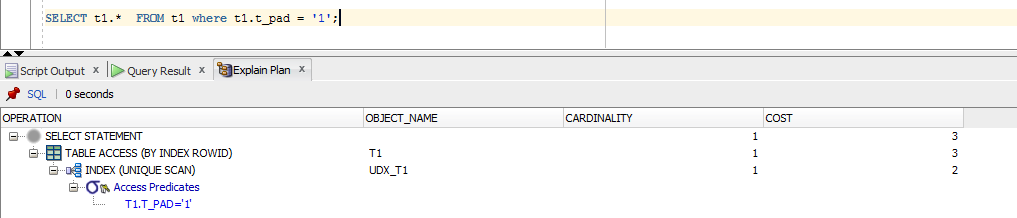
### Task Result

SELECT t1.\* FROM t1 where t1.t\_pad = '1';

Выполнив данный запрос получили пустое значение. Так как в ячейке содержится не только значение 1 но и пробелы.



На плане видно, что был задействован index unique scan .



Unique index scan выбирается, когда предикат содержит условие с использованием столбца, определенного с индексом UNIQUE или PRIMARY KEY. Эти типы индексов гарантируют, что только одна строка будет возвращена для указанного значения.

При проведении опыта несколько раз были случаи, что вместо индексного метода доступа был full scan. Если же изменить первоначальный запрос, добавив в выражении where t1.t\_pad = '1 ' некоторое количество пробелов , то срабатывает индексный метод.

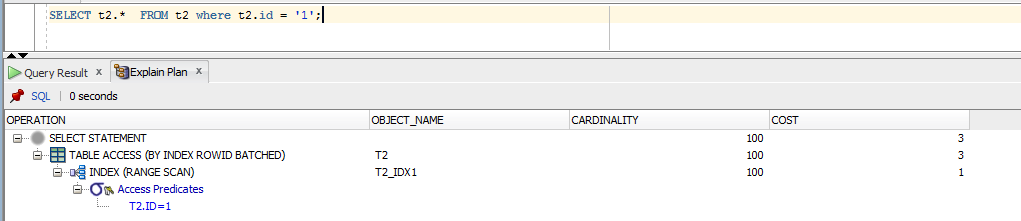
## Task 4: Index Range Scan

### Task Result

SELECT t2.\* FROM t2 where t2.id = '1';

Index range scan используется при возвращении диапазона значений. Выполняется таким образом, что для начала находится начальный листовой блок, и затем продвижение происходит только по листовым блокам. Нет необходимости просматривать весь индекс сверху вниз. Так как в запросе указан не только индексный столбец, то доступ к данным будет осуществляться по row\_id.

TABLE ACCESS (BY INDEX ROWID BATCHED) означает, что база данных извлекает несколько rowid из индекса.



## Task 5: Index Skip Scan

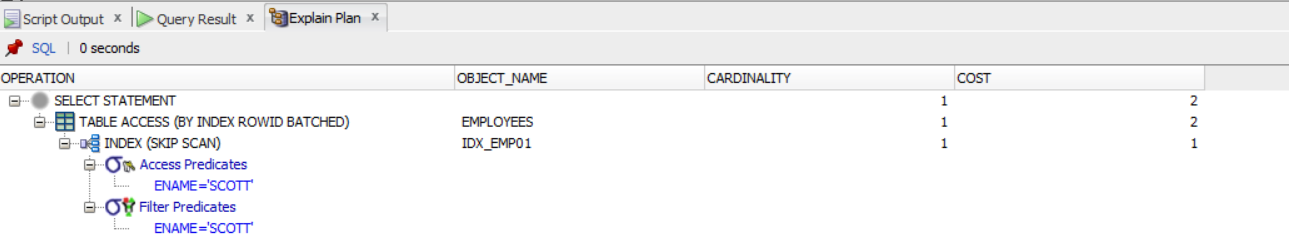
### Task Result

CREATE INDEX idx\_emp01 ON employees

( empno, ename, job );

1.SELECT /\*+INDEX\_SS(emp idx\_emp01)\*/ emp.\* FROM employees emp where ename = 'SCOTT';

2. SELECT /\*+FULL\*/ emp.\* FROM employees emp WHERE ename = 'SCOTT';



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 8 | 1 | 14 | 2 | Index |
| 2 | 8 | 1 | 14 | 3 | Full |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Index Skip Scan используется в случаях, когда первый столбец составного индекса не входит в условия запроса. Т.е. столбец пропускается при выполнении запроса. В данном методе доступа используется разделение составного индекса на логические части по числу неповторяющихся значений основного столбца составного индекса. Если таких логических частей будет мало, то как раз и сработает Skip Scan.