****

**EPAM Training Center**

**DWH**

**Tasks 4**

**Report**

**Minsk, 2017**

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 03.11.2017 | 0.1 | Description of workflow (Tasks 4) | Olga Hilko |
|  |  |  |  |

Contests

[Introduction 4](#_Toc497490080)

[Purpose 4](#_Toc497490081)

[1 Table access 4](#_Toc497490082)

[1.1 Full table scan 4](#_Toc497490083)

[2 Index scan 4](#_Toc497490084)

[2.1 Index Clustering Factor 4](#_Toc497490085)

[2.2 Index Unique Scan 6](#_Toc497490086)

[2.3 Index Range Scan 6](#_Toc497490087)

[2.4 Skip scan 7](#_Toc497490088)

# Introduction

## Purpose

This document includes the results of completed task 4.

# Table access

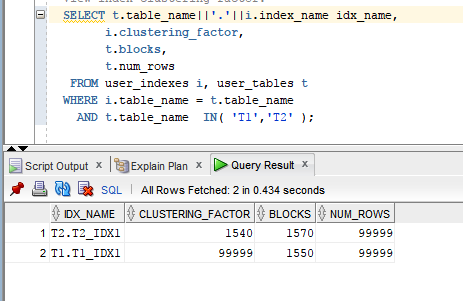
## Full table scan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 3 | 1664 | 1536 | 99 999 | 1539 | Created table with 99999 rows, it has unused blocks |
| 4 | 1664 | 0 | 0 | 1541 | Data were deleted, but not commited, they still occupy blocks in memory. The blocks are dirty. |
| 7 | 1664 | 1536 | 100 000 | 1541 | Data (1 row) were inserted and commited, trace canceled the delete, the blocks are still occupied. In fact there is only one row, but the system in enforced to read all |
| 9 | 8 | 0 | 0 | 3 | Table was cleaned entirely, only metadata is left. Blocks are cleaned. |

**The autotrace provides instantaneous feedback including the returned rows, execution plan, and statistics.** Consistent gets in our case means session logical reads.

# Index scan

## Index Clustering Factor



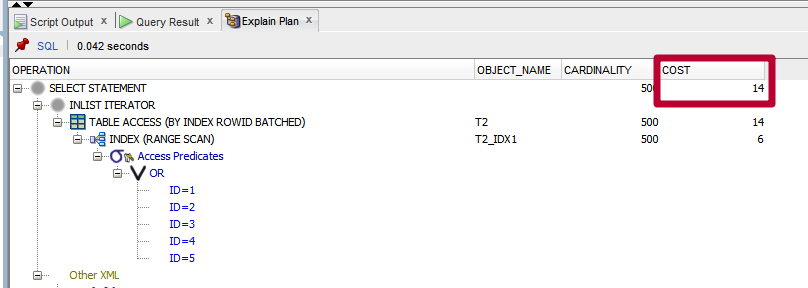
The **clustering factor** is a measure of the ordered-ness of an index in comparison to the table that it is based on. It means how perfect are the data ordered in comparison with the native table with data.

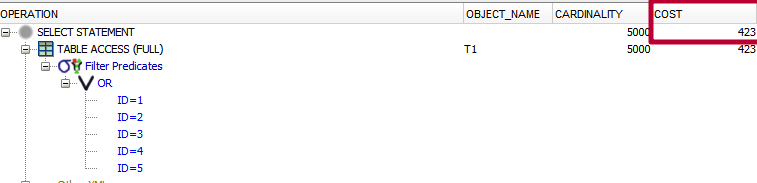
If the value **is close** to the total **number of** **blocks**, then the table is **very well ordered**. In this case, index entries in one leaf block usually indicate the rows that are in the same data blocks.

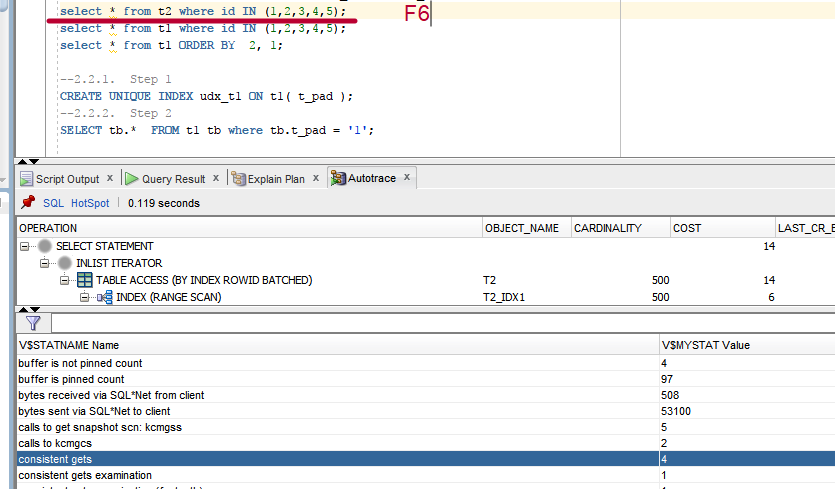
If the value **is close** to the total **number of** **rows**, then the table is **very unordered**. In this case, it is unlikely that the index entries in one leaf block point to the same data blocks.

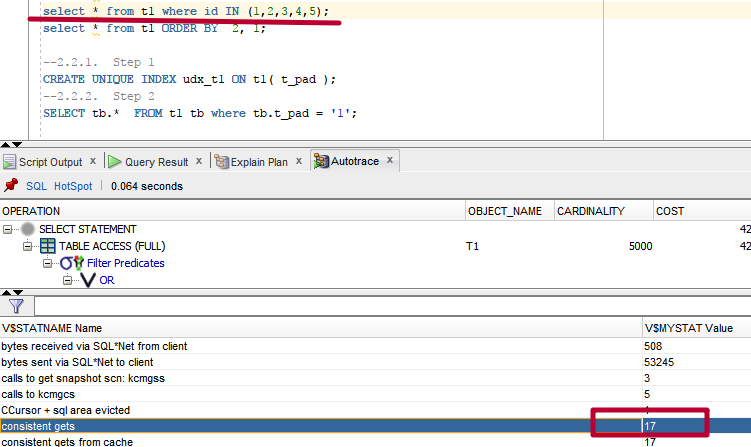
**Conclusion** t2 table is well ordered, t1 is unordered. T2 has repeatable, but consequential id values and they are native ordered by insert. T1 has different id values in the same or close blocks, the same values are located in separate far away blocks

The best performances has t2 (ordered table). In the t1 index in not used at all. Optimizer do not use it, because it is not effective.

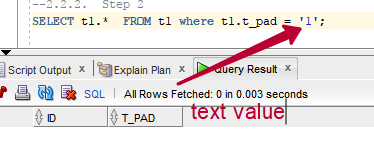
.





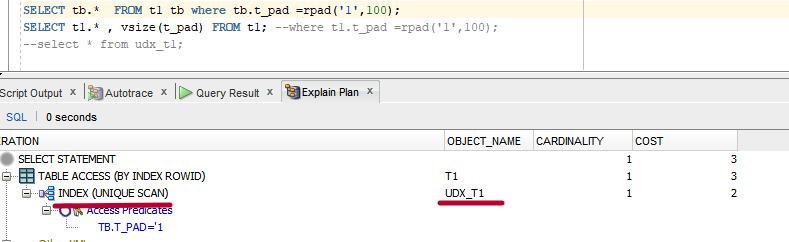


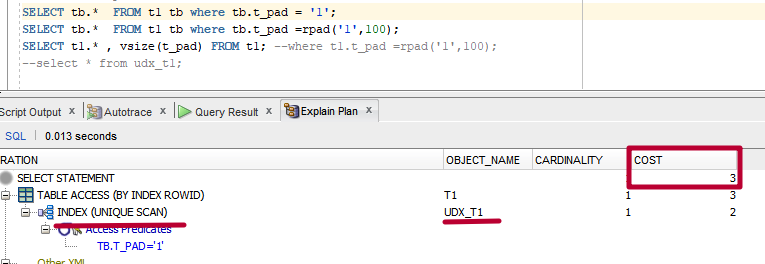
## Index Unique Scan



Result set is empty, because the searched value is ‘1’ and fact value is ‘1’+<99 spaces>. But we need to investigate the search method.

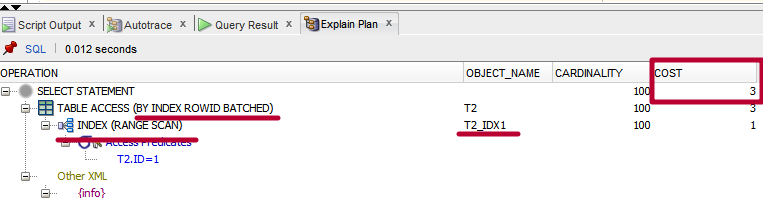
Let’s change the query and investigate both of them (native and modified). The execution plans are the same.



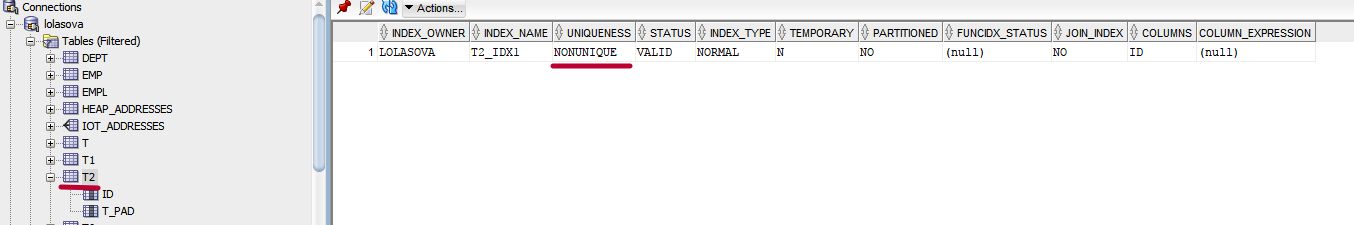


We are going to search by field, which is the base for the unique index. It means, that the optimizer can take a look through unique index structure (the index structure will be viewed from root to leaf block to a single index-record), find the row id of the SINGLE row (index is unique – the value must be single) and get all necessary fields (id, t\_pad) from the table. In the case if the proper index value is not found there is no need to go to the table and take the whole row.

## Index Range Scan



In this case we have non-unique index and the optimizer uses it because we have equality-clause on the indexed field in the where predicate.

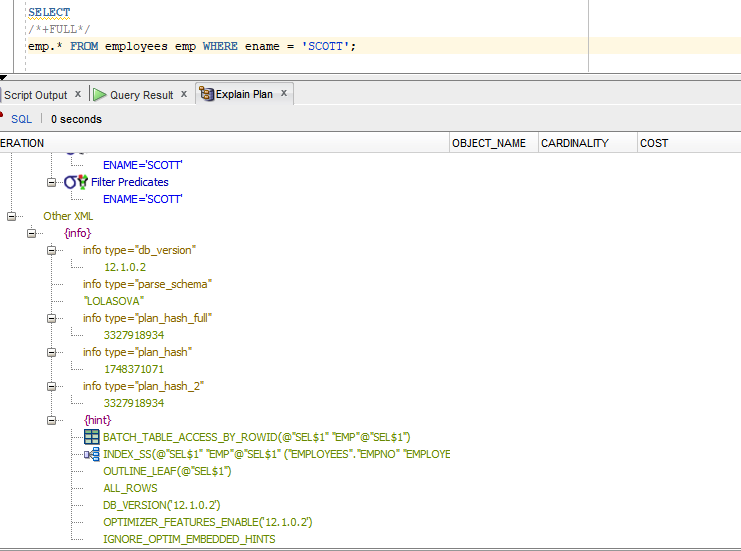


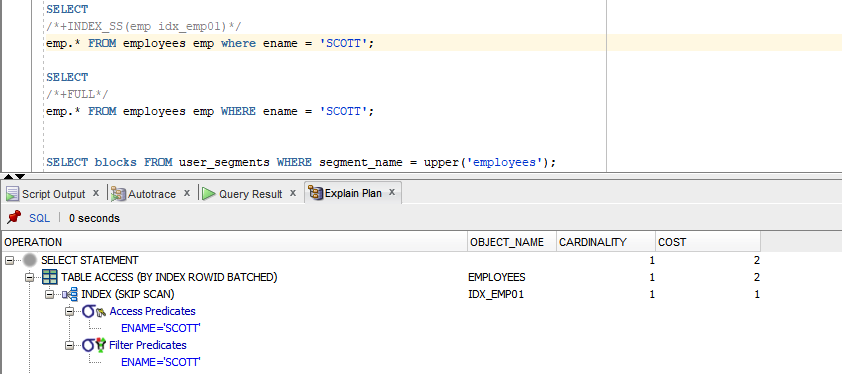
## Skip scan

The method is used in the case we have a composite index and the search is held by the second index-column value.

Execution plans are equal.







|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Select | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 8 | 1 | 1 | 2 | All fields in statistics are equal except red values (see in the bottom) |
| 2 | 8 | 1 | 1 | 2 |

|  |  |
| --- | --- |
| buffer is not pinned count | 2 |
| **bytes received via SQL\*Net from client** | **552** |
| bytes sent via SQL\*Net to client | 48037 |
| calls to get snapshot scn: kcmgss | 5 |
| calls to kcmgcs | 2 |
| consistent gets | 2 |
| consistent gets from cache | 2 |
| consistent gets pin | 2 |
| consistent gets pin (fastpath) | 2 |
| CPU used by this session | 3 |
| CPU used when call started | 3 |
| **DB time** | **4** |
| enqueue releases | 1 |
| enqueue requests | 1 |
| execute count | 2 |
| logical read bytes from cache | 16384 |
| no work - consistent read gets | 2 |
| non-idle wait count | 25 |
| opened cursors cumulative | 2 |
| parse count (hard) | 1 |
| parse count (total) | 2 |
|  |  |
| recursive calls | 1 |
| Requests to/from client | 25 |
| session cursor cache hits | 1 |
| session logical reads | 2 |
| sorts (memory) | 2 |
| sorts (rows) | 2356 |
| SQL\*Net roundtrips to/from client | 25 |
| table fetch by rowid | 1 |
| user calls | 27 |
| workarea executions - optimal | 5 |

|  |  |
| --- | --- |
| buffer is not pinned count | 2 |
| **bytes received via SQL\*Net from client** | **533** |
| bytes sent via SQL\*Net to client | 48037 |
| calls to get snapshot scn: kcmgss | 5 |
| calls to kcmgcs | 2 |
| consistent gets | 2 |
| consistent gets from cache | 2 |
| consistent gets pin | 2 |
| consistent gets pin (fastpath) | 2 |
| CPU used by this session | 2 |
| CPU used when call started | 2 |
| **DB time** | **5** |
| enqueue releases | 1 |
| enqueue requests | 1 |
| execute count | 2 |
| logical read bytes from cache | 16384 |
| no work - consistent read gets | 2 |
| non-idle wait count | 25 |
| opened cursors cumulative | 2 |
| parse count (hard) | 1 |
| parse count (total) | 2 |
| parse time elapsed | 1 |
| recursive calls | 1 |
| Requests to/from client | 25 |
| session cursor cache hits | 1 |
| session logical reads | 2 |
| sorts (memory) | 2 |
| sorts (rows) | 2356 |
| SQL\*Net roundtrips to/from client | 25 |
| table fetch by rowid | 1 |
| user calls | 27 |
| workarea executions - optimal | 5 |