|  |
| --- |
| EPAM Systems, RD Dep. |
| MTN.BI.07 Oracle Data Access and Optimizer |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Hanna Klimovich](mailto:Kiryl_Bucha@epam.com) | 7-NOV-2017 |  |  |

# 1 Table access

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 1664 | 1536 | 99999 | 1541 | Table was created. There are 99999 rows. |
| 2 | 1664 | 0 | 0 | 1619 | Deleting all rows. 0 blocks are used. |
| 3 | 1664 | 1 | 1 | 1542 | Inserting one row, but blocks are still not empty. This row inserted into one block. |
| 4 | 8 | 0 | 0 | 5 | Table was truncated and blocks are cleaned. |

# The autotrace report is generated after successful SQL DML statements. It is useful for monitoring and tuning the performance of these statements.

# Index Scan types

## Task 2: Index Clustering Factor

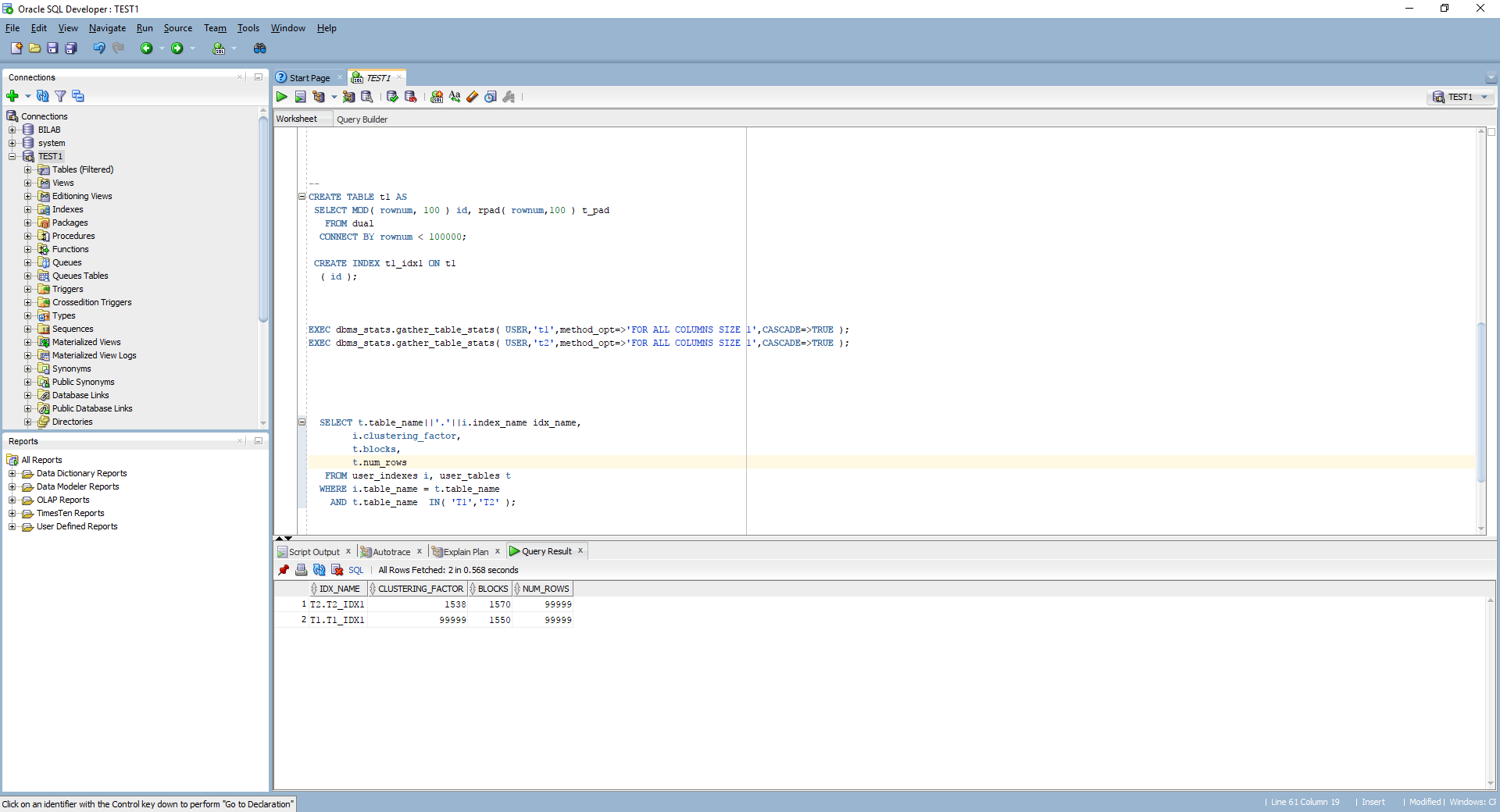


Figure 1 Index clustering factor

The index clustering factor is a measure of how many I/Os the database would perform if it were to read every row in that table via the index in index order or how many time the database should go to the datablocks to retrieve all needed data.

To compare table 1 and table 2.

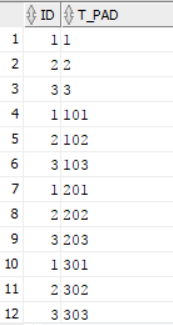
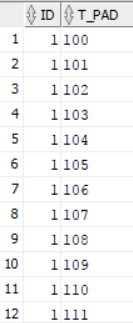
 

Figure 2 t1 Figure 3 t2

Table 2 has a better performance because the rows are sorted by index key, so the next row from the index key will be the next row in the table block. So it’s not needed to go through the whole blocks.

## Task 3: Index Unique Scan

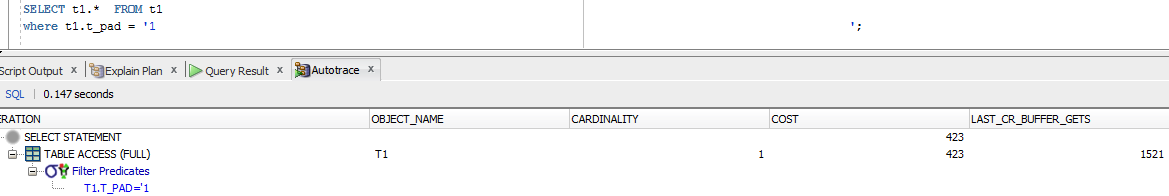


Figure 4 table without index

Without the index, database will go through the whole values it doesn’t matter where these values found or not.

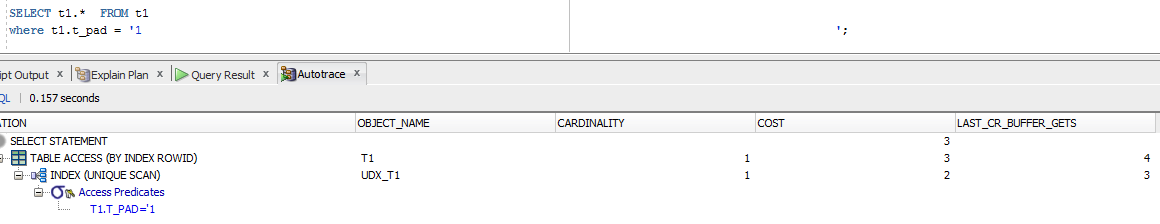
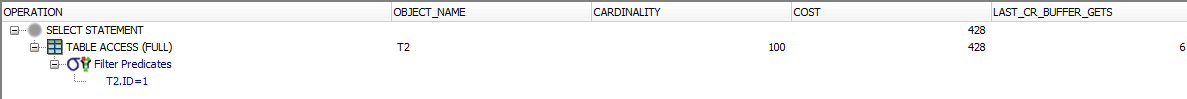
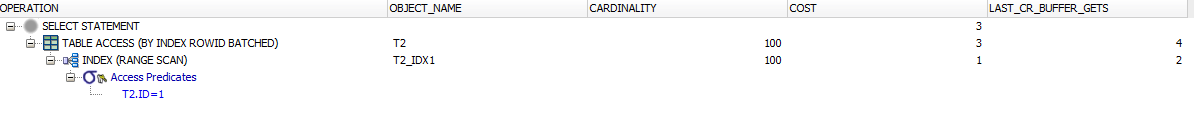


Figure 5 Table with unique index

With unique index when the database find the needed value in the tree – search stops. That’s why the cost in the second case is much smaller the in the first one.

## Task 4: Index Range Scan

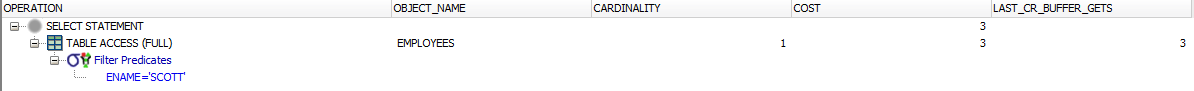


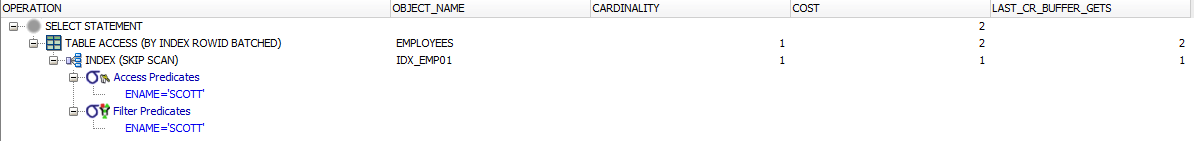


With index range scan there is a condition that return a range of values.

It is not necessary to search through the whole table because it is starts from the root block to the first leaf block containing an entry matching the specified condition. Here he retrieves a rowid and then retrieves a table data block. After that the next rowed will be retrieved.This back-and-forth between the index leaf blocks and the data blocks will continue until all the matching index entries have been read. If the range of entries matching the condition is large enough, it is likely that more than one leaf block will have to be accessed, so the next leaf block needed can be read using a pointer stored in the current leaf block that leads to the next leaf block (there’s also a pointer to the previous leaf block). Since these pointers exist, there is no need to go back up to the branch block to determine where to go next.

## Task 5: Index Skip Scan





Here we decided by ourselves which types of scan we would like to have. We use skip scan because predicate contains a condition on a non-leading column in an index and the leading columns are distinct.