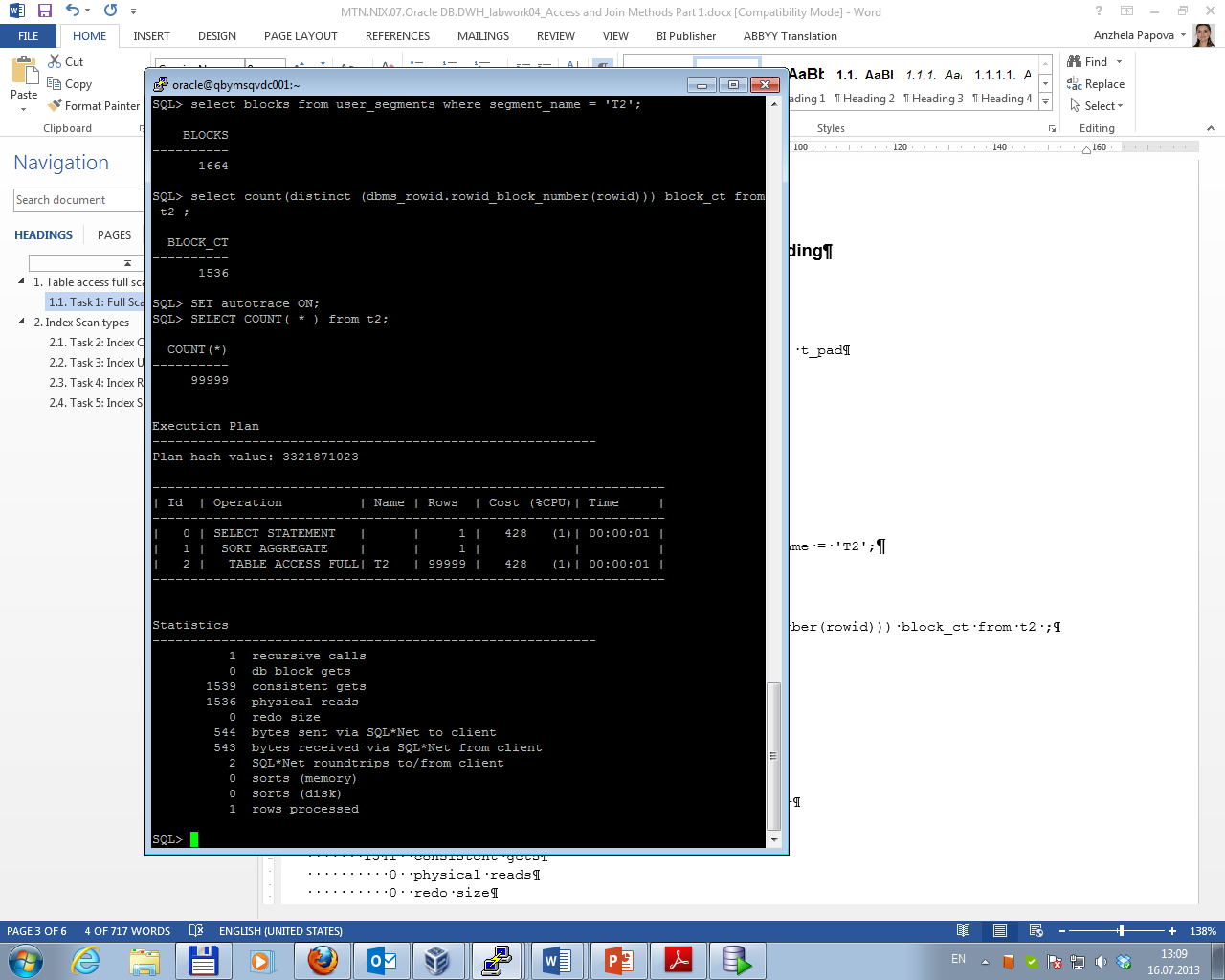
Here are the results of labwork 04:

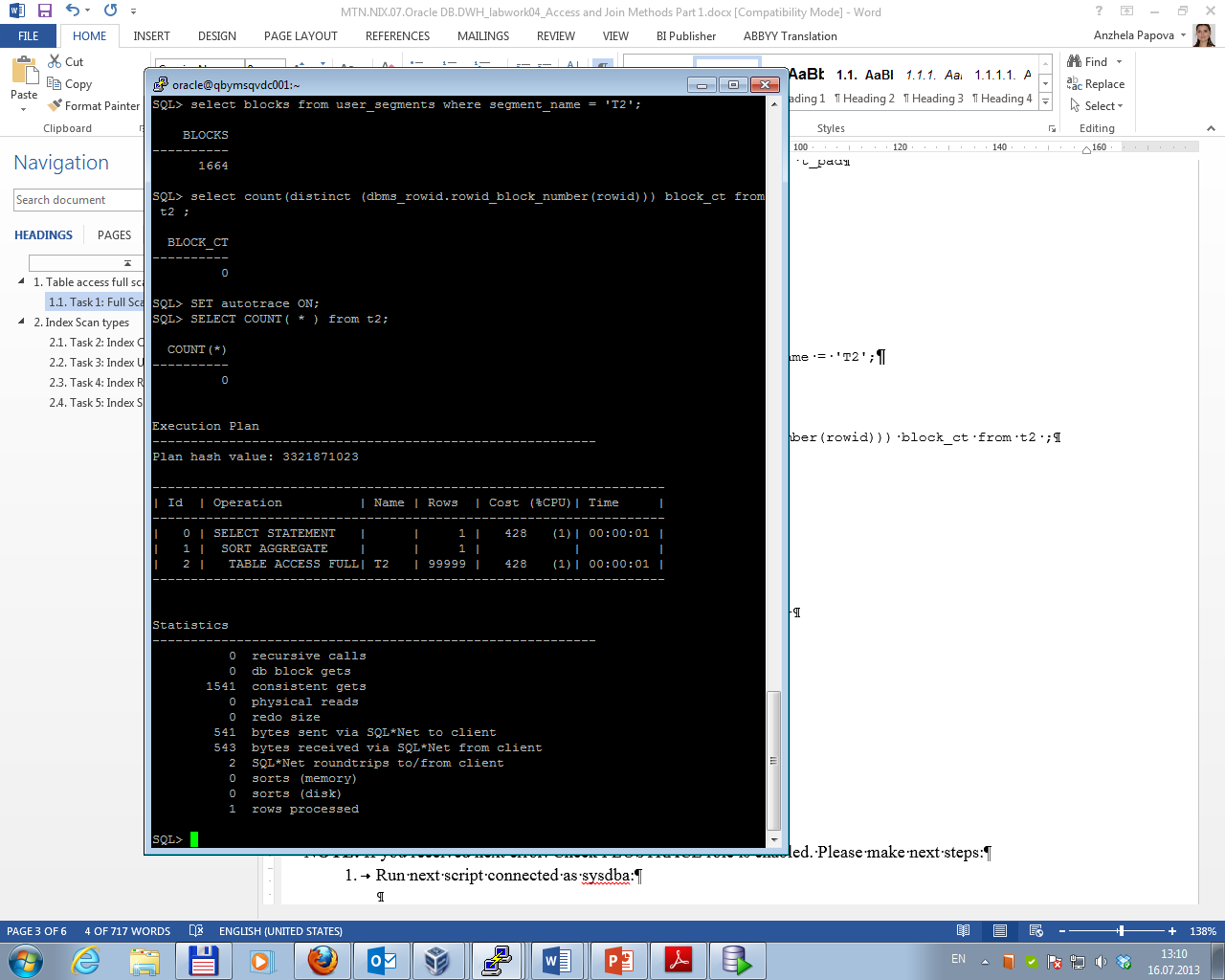
**Table access full scan**

Task 1 Full Scans and the High-water Mark and Block reading

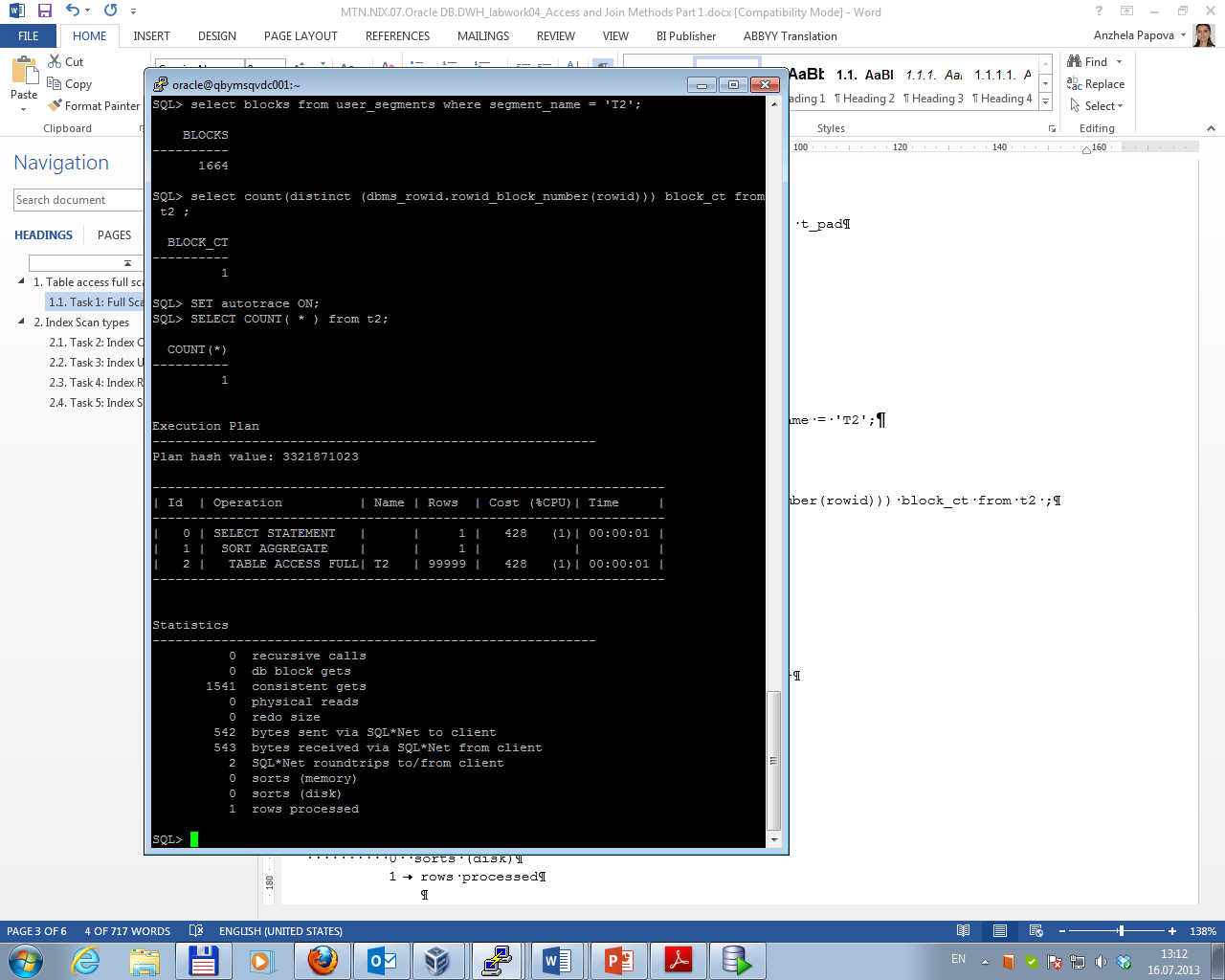
After creating table t2 and index t2\_idx1 on it the next results were gotten:



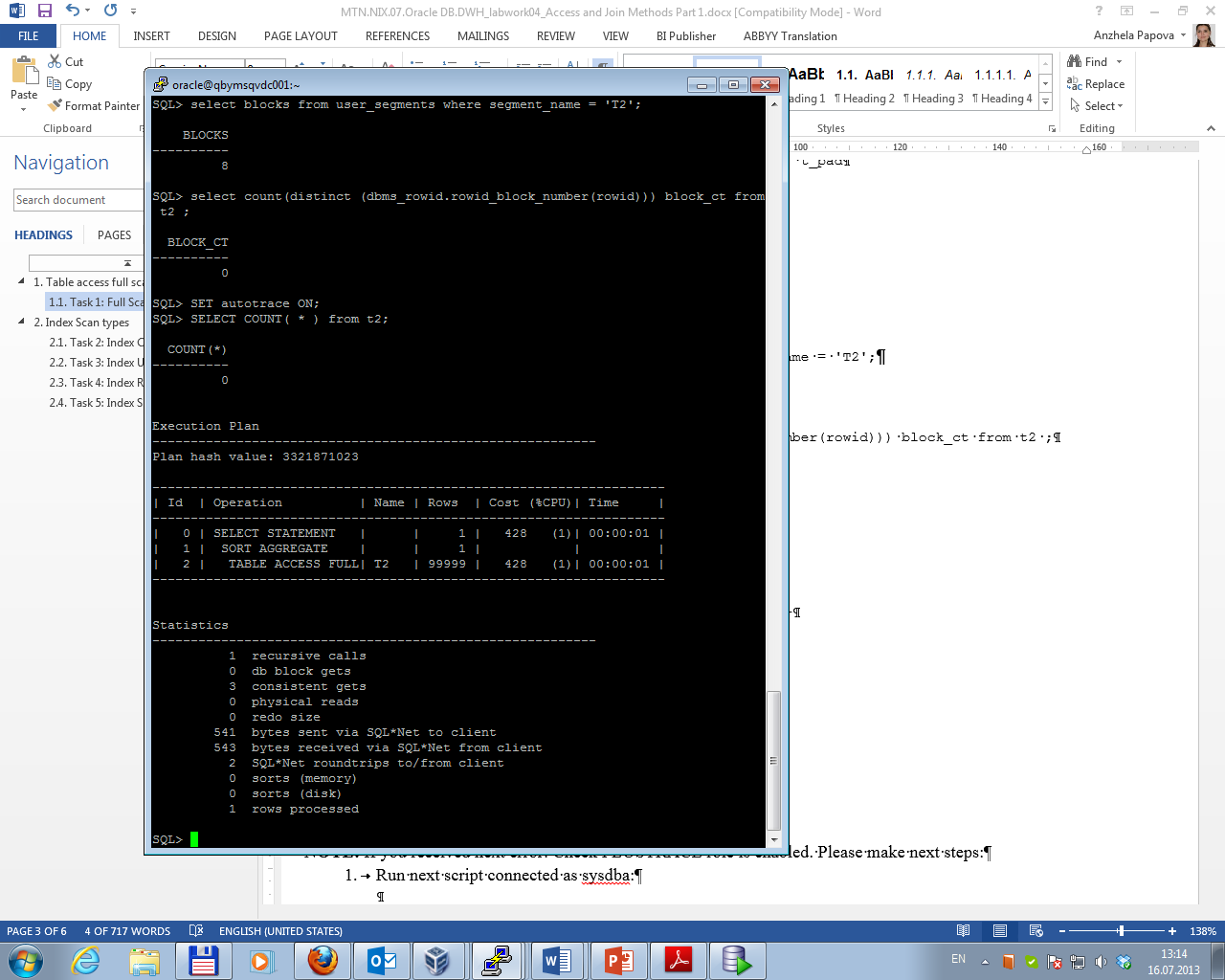
After deleting all rows from table the next results were gotten:



After inserting 1 row table the next results were gotten:



After truncating table the next results were gotten:



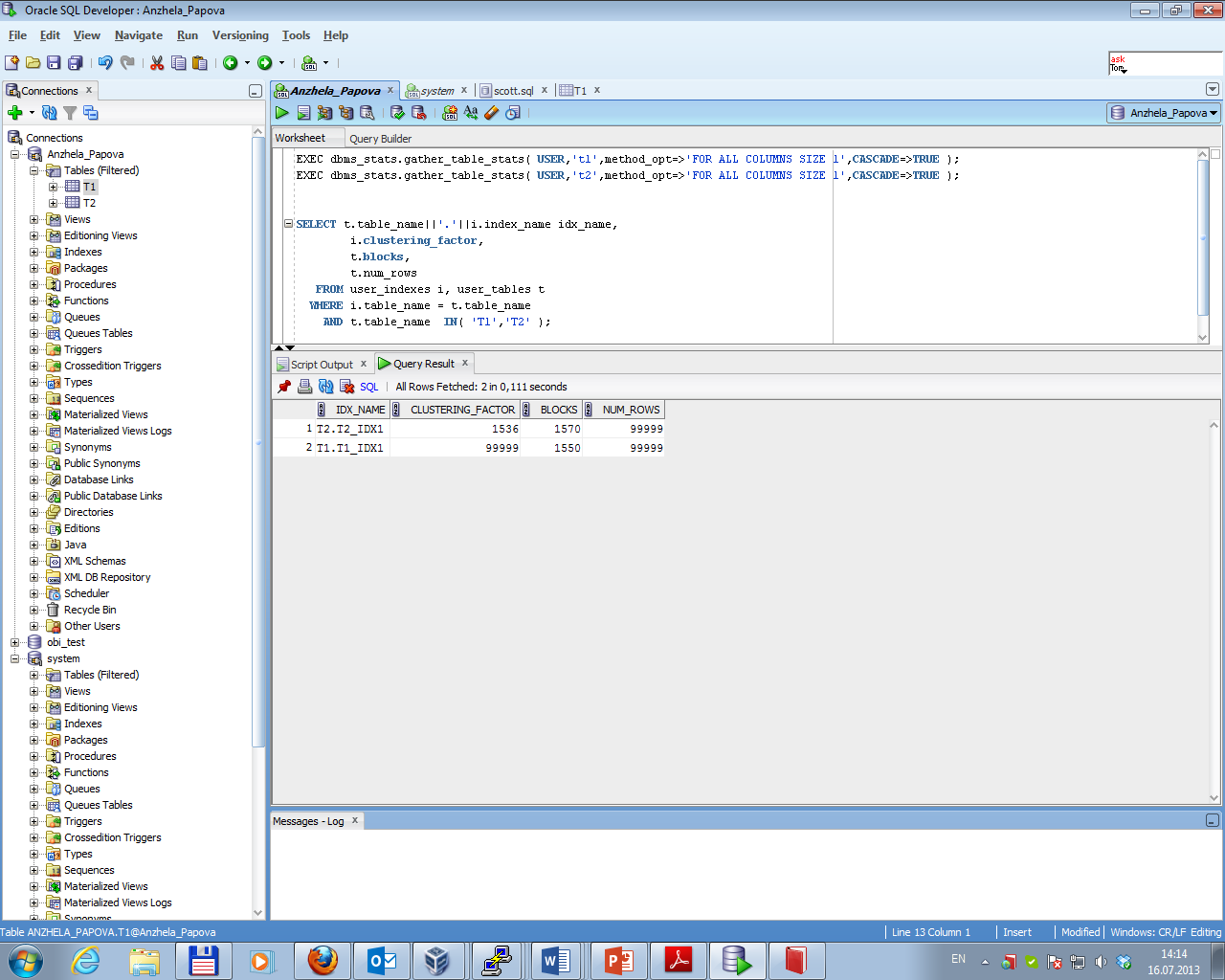
Total results are presented below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 1664 | 1536 | 99999 | 1539 | After creating table 1664 blocks were allocated, 1536 from which contained the data. A full scan is an effective choice to count all rows in the table. Oracle has read blocks up to the highwater mark in the table. It was necessary to make 1539 consistent gets. |
| 2 | 1664 | 0 | 0 | 1541 | After deleting rows in the table the highwater mark is still at the end of table. That’s why all blocks up to the highwater mark were read in and scanned. Although there is no date in table, it was necessary to make 1541 consistent gets. |
| 3 | 1664 | 1 | 1 | 1541 | After inserting one row in the table (at the end of table) the highwater mark remained at the end of last (the same) block. So, it was necessary to make 1541 consistent gets. |
| 4 | 8 | 0 | 0 | 3 | After truncating table there are no block with data in it (only blocks with metadata remained); highwater mark is at the end of last block. It was necessary to make 3 consistent gets. |

**Index Scan types**

Task 2 Index Clustering factor parameter

After creating tables t1, t2 and indexes on them statistic was calculated. The results are presented below.



Clustering factor indicates the amount of order of the rows in the table based on the values of the index:

• if the value is near the number of blocks, then the table is very well ordered.

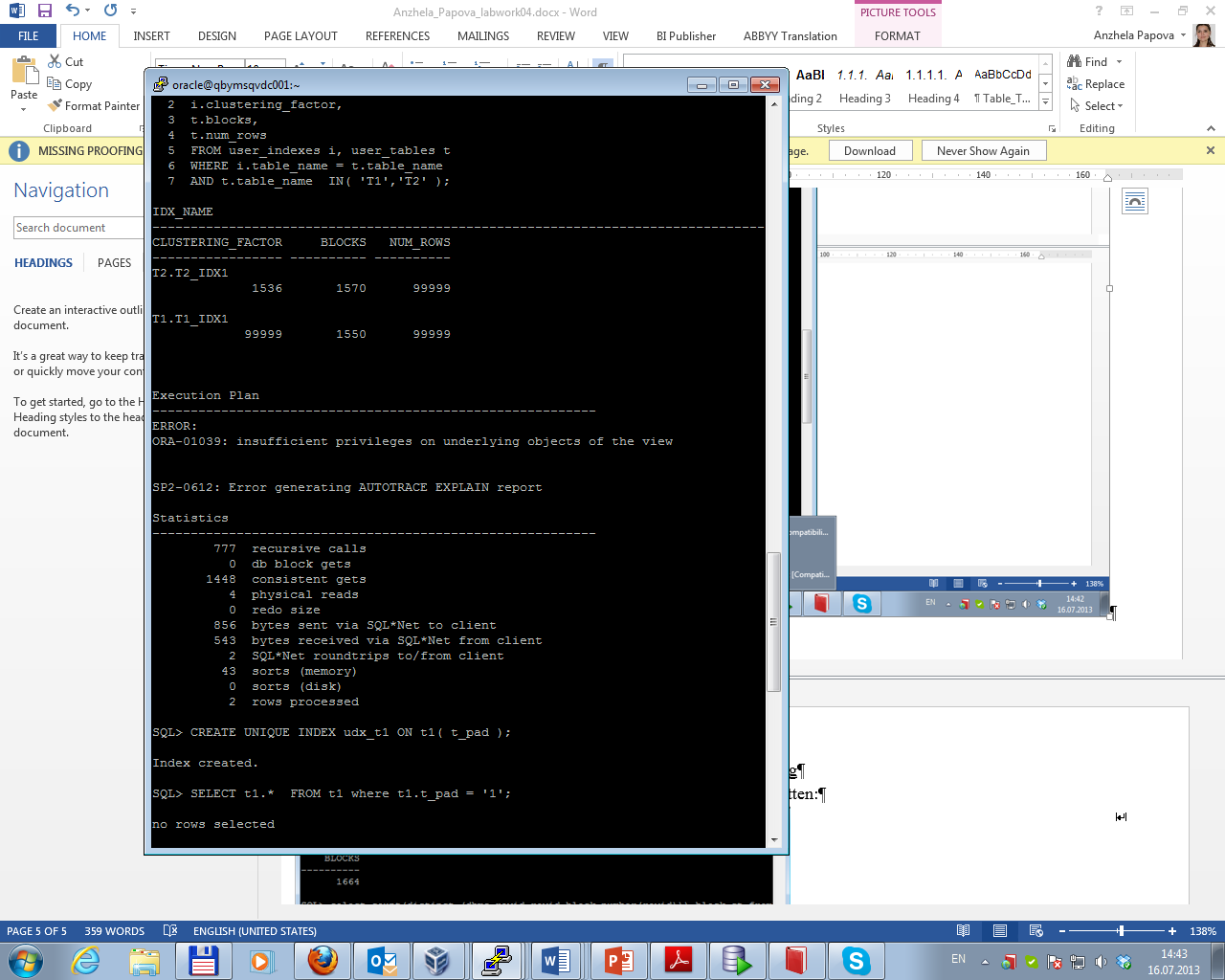
• if the value is near the number of rows, then the table is very randomly ordered.

So, as clustering factor of t1\_idx1 is equal the number of rows than table t1 is very randomly ordered; as clustering factor of t2\_idx1 is near the number of blocks than table t2 is well ordered. Besides, for the table t2 clustering factor is less than the number of blocks in the table. The reason of this is that there are unformatted blocks in the table (that do not contain data).

The best selective performance in execution Select clause filtered by IN (, list of values,) has index t2\_idx1.

Task 3 Index Unique Scan

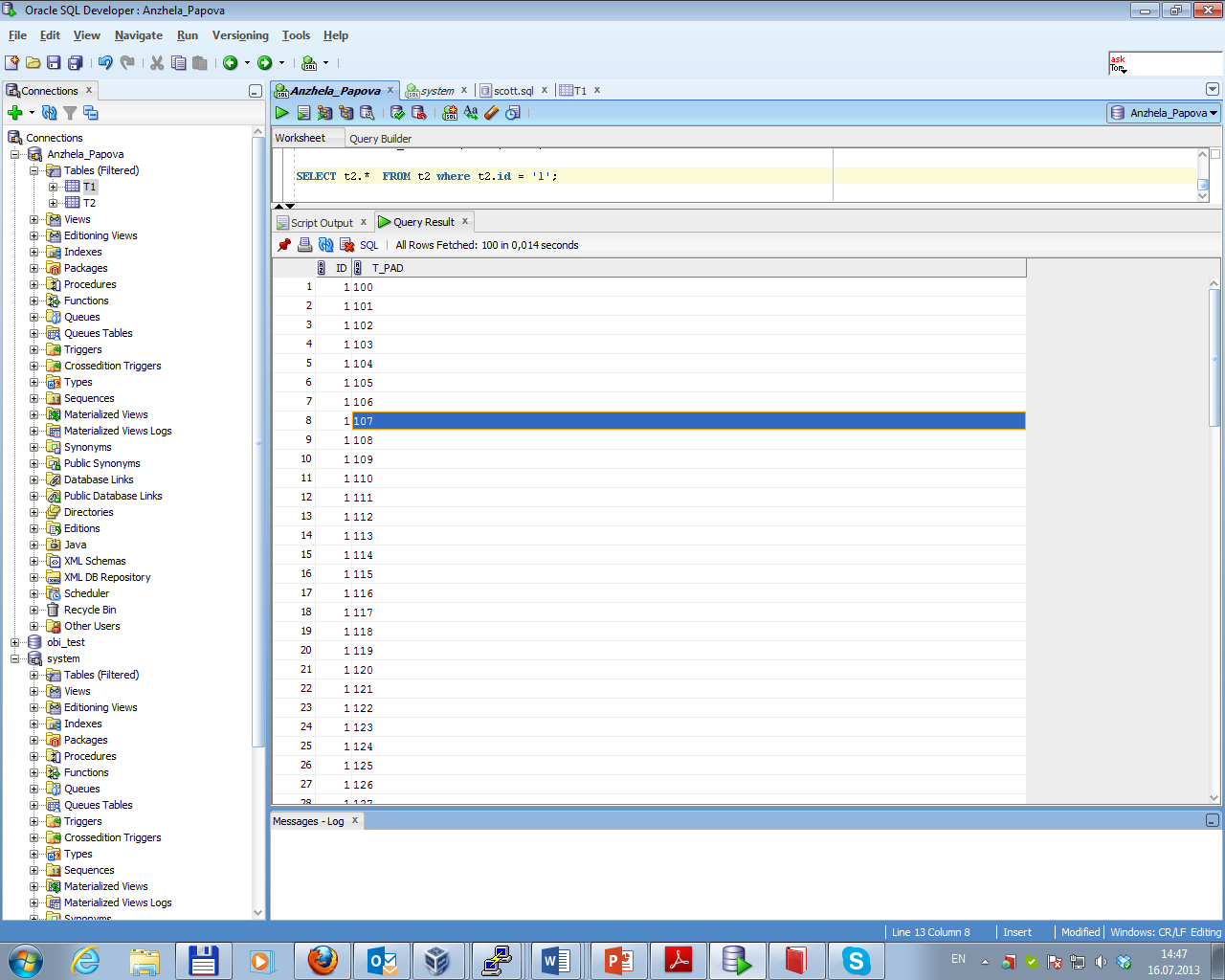
After creating unique index necessary SELECT command was run. The result is presented below.



Index unique scan must have 0 or 1 rowid associated with an index key. The database performs a unique scan when a predicate references all of the columns in a UNIQUE index key using an equality operator. An index unique scan stops processing as soon as it finds the first record because no second record is possible.

Task 4 Index Range Scan

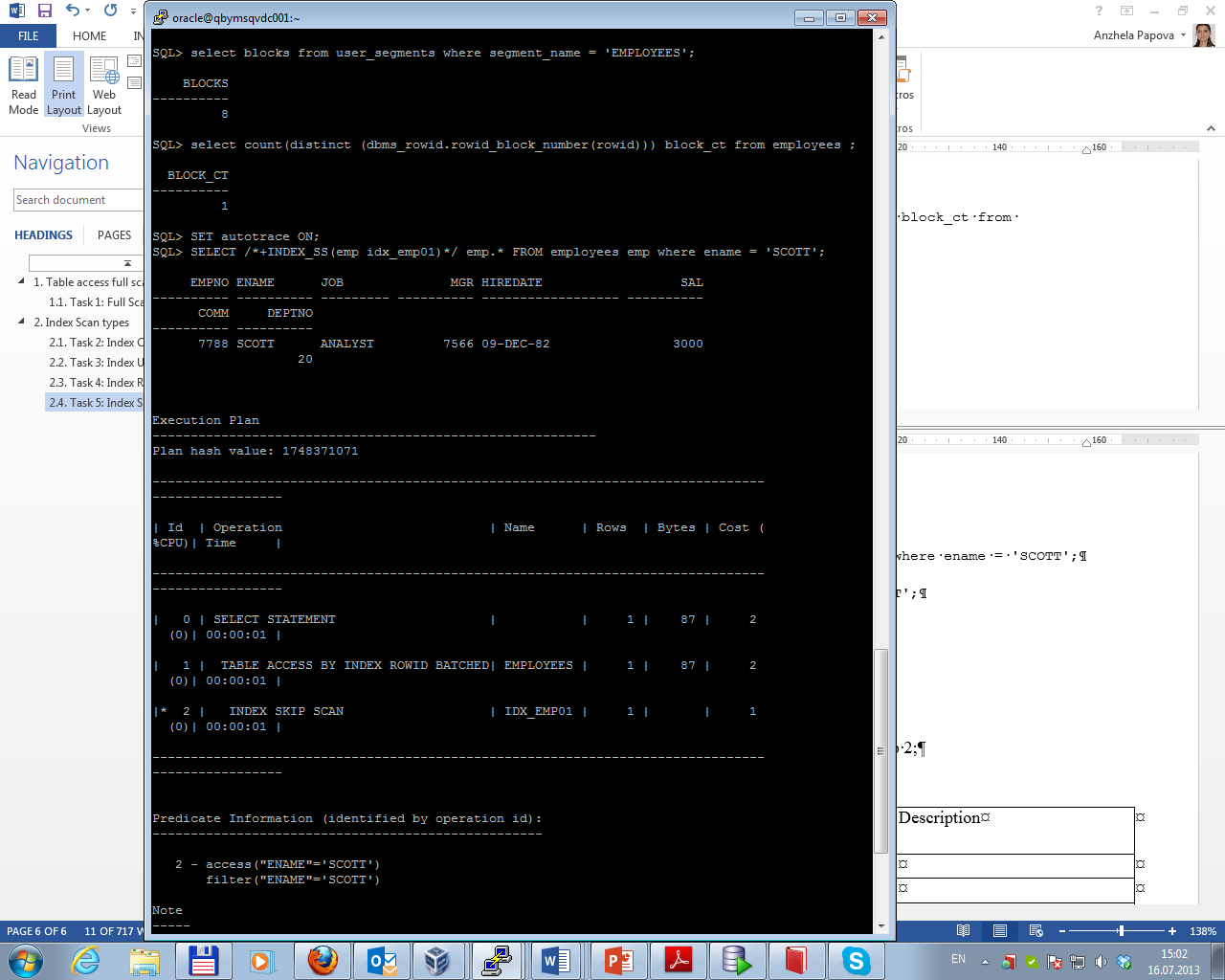
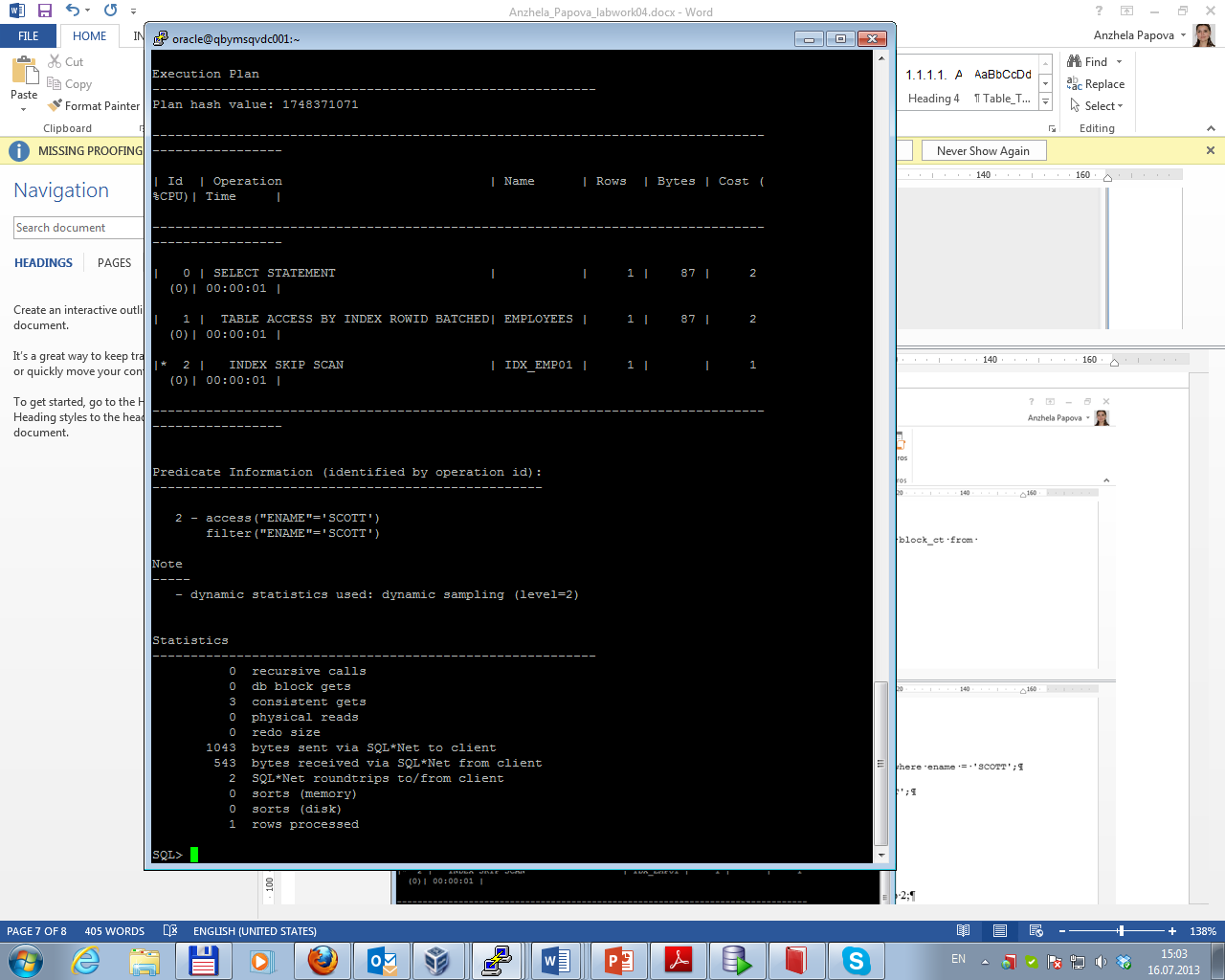
Result of running SELECT command is presented below.



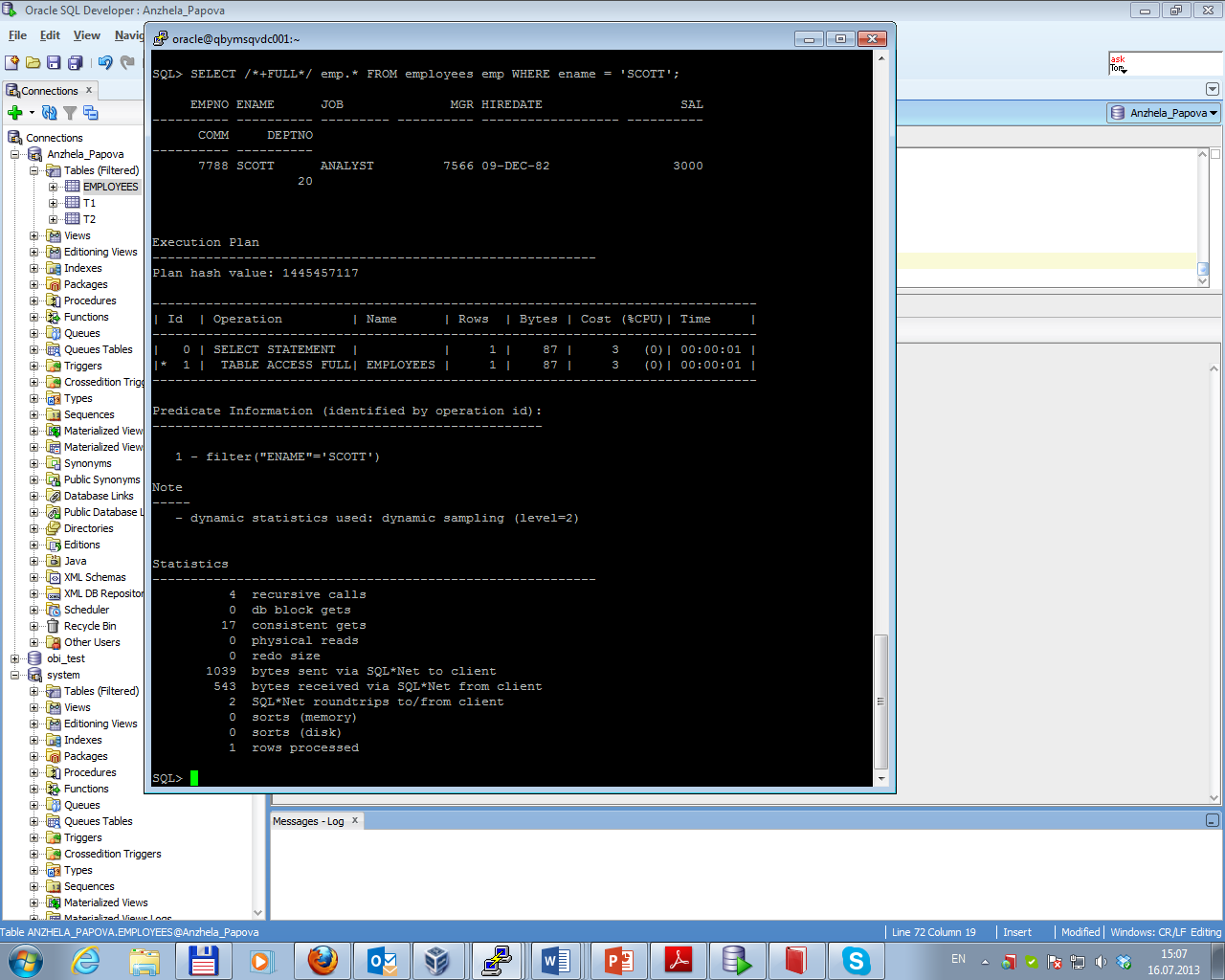
In this case index range scan was used. It’s mean that Oracle finds the first index leaf block that contains key value = 1 and then it just walks horizontally through the linked list of leaf nodes until it finally hits a value that is greater than 1.

Task 5 Index Skip Scan

After creating of table employees and index idx\_emp01 on it the next traces and statistics of explain plan were gotten.

In this case index skip scan was used. It’s mean that Oracle is going to skip throughout the first part of index (empno), looking for points where empno changes values and read down the tree from there looking for ename = ‘SCOTT’.



In this case full table access was used; index was skipped at all. That’s why the cost value became 3 (in the previous example it was 2) and 17 consistent gets were needed (in the previous example – only 3).

Total results are presented below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Consistent gets | Recursive calls | Cost | Rows processed | Description of Execution Plan |
| 1 | 3 | 0 | 2 | 1 | Table access by index, then Index skip scan were used  Small number of consistent gets because it’s not necessary to look through entire table  Cost value = 2  Predicates: 2 (one condition was used to locate rows in n access structure and one more – to filter rows after they have been accessed)  This case is quite effective |
| 2 | 17 | 4 | 3 | 1 | Full table access was used  Large number of consistent gets because it’s necessary to look through entire table  Cost value = 3  It took 4 recursive calls to parse that query (a hard parse is when we parse and cannot find the query in the shared pool)  Predicates: 1 (the only condition - to filter rows after they have been accessed by using full table access)  This case is not the effective one |