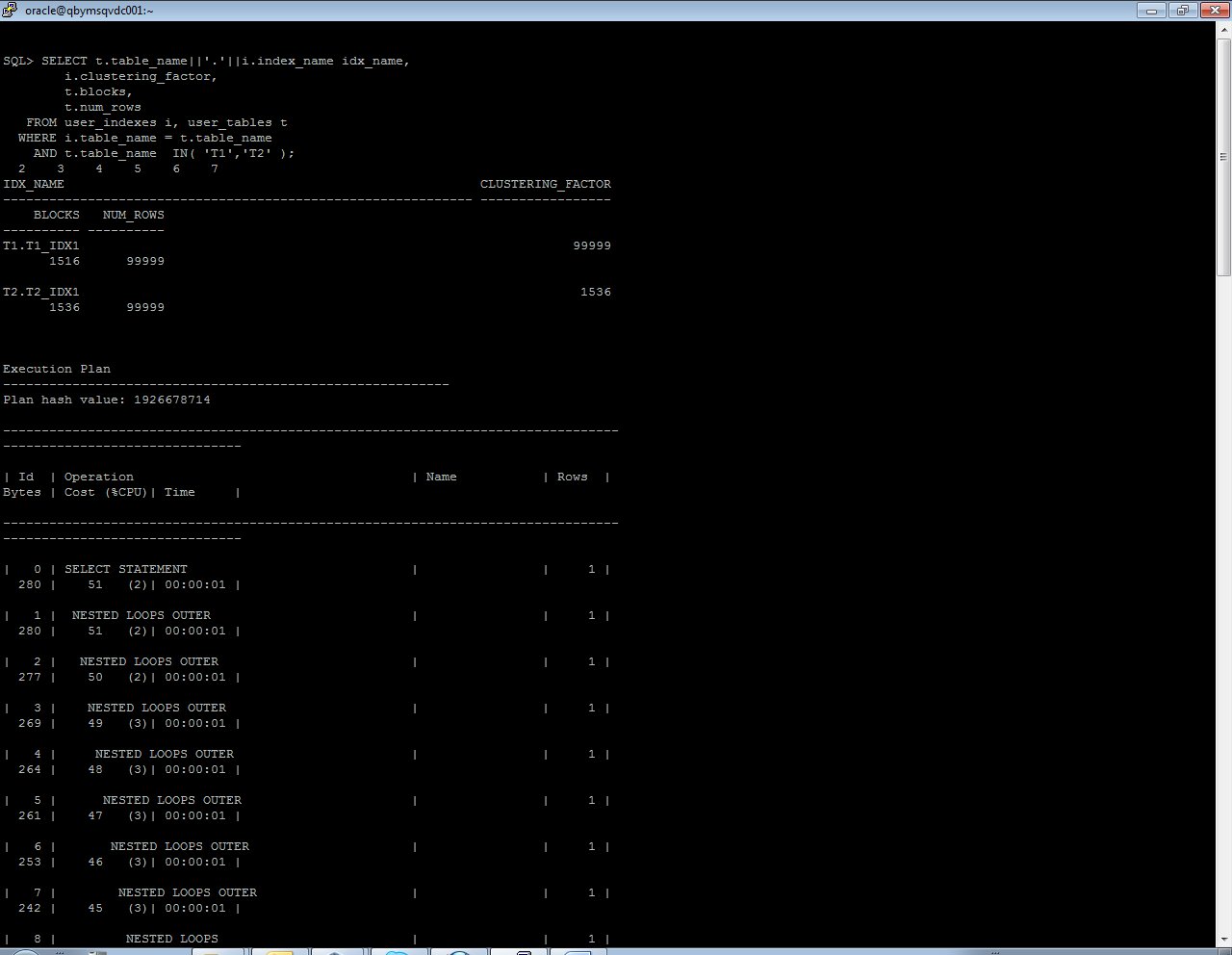
Task 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 1664 | 1536 | 99999 | 1607 | 1614 blocks allocated for storing data. All rows are in 1536 of blocks. HWM set to 1607 |
| 2 | 1664 | 0 | 0 | 1607 | We delete data but HWM is still 1607 |
| 3 | 1664 | 1 | 1 | 1607 | After scanning we see that used only 1 block, but HWM is not changed despite the fact that blocks are emty. |
| 4 | 8 | 0 | 0 | 5 | Only after truncate table HWM was reseted. |

Task 2.

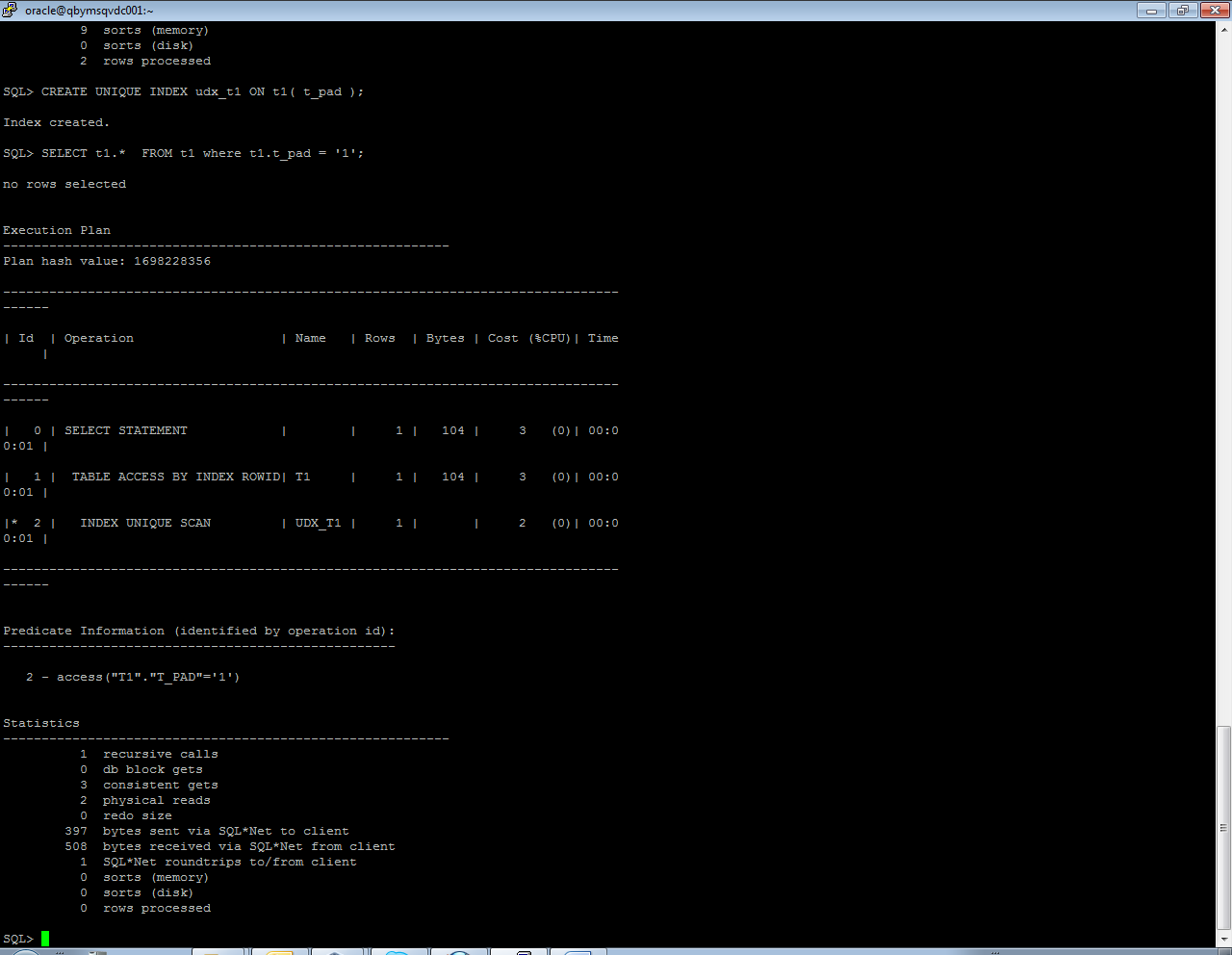


Clustering factor is a value that show how synchronized an index is with the data in a table.  A table with a high clustering factor is out-of-sequence with the rows and large index range scans will consume lots of I/O.

We have different values because in t2 we have a lot fo duplicate values. Because of that clustering factor of this table less than t1.

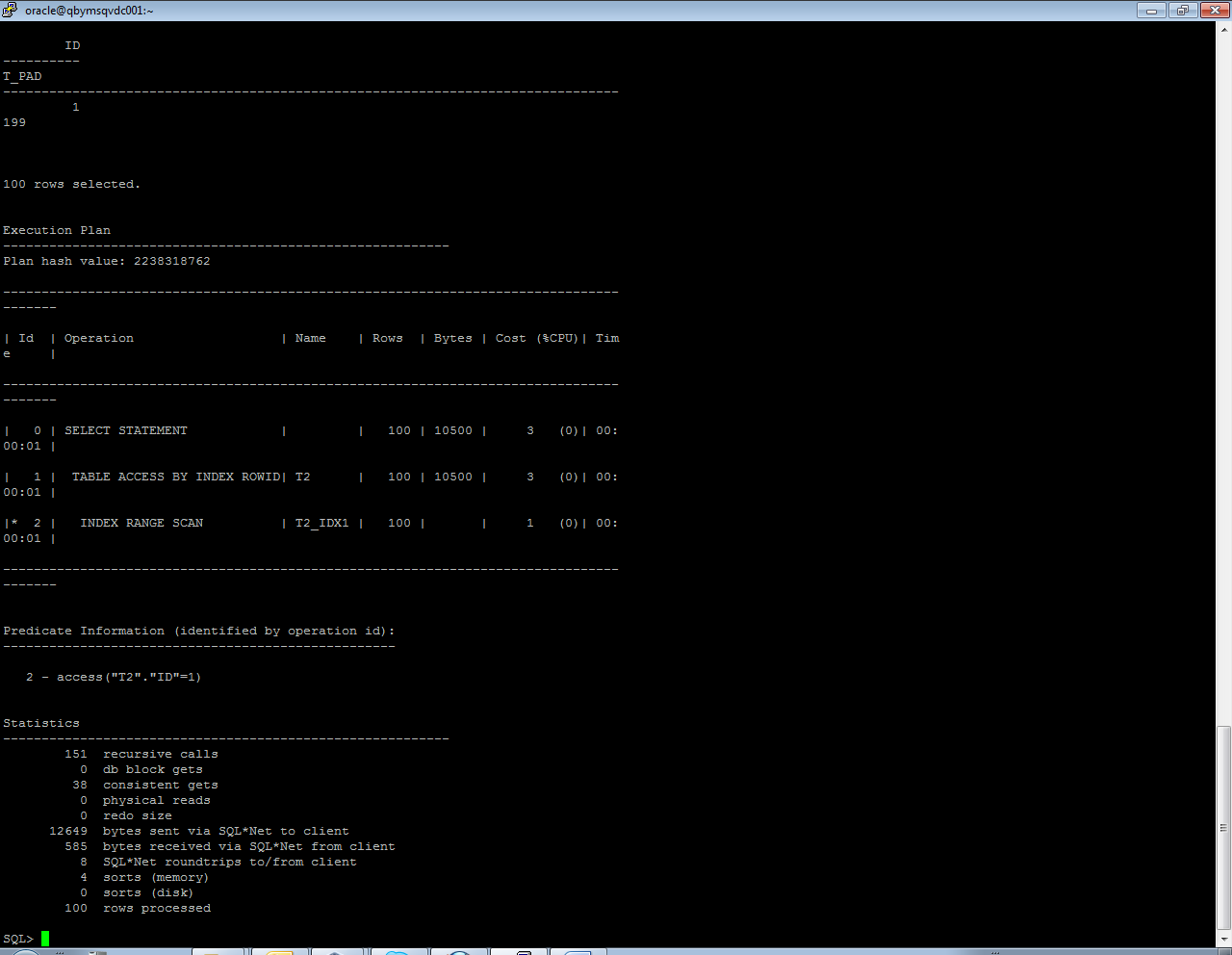
Index of table 2 has best perfomance. Cause – clustering factor lower than on table 2.

Task 3.



An index unique scan is chosen when a predicate contains a condition using a column defined with a UNIQUE or PRIMARY KEY index. These types of indexes guarantee that only one row will ever be returned for a specified value. In this cases, the index structure is traversed from root to leaf block to a single entry, retrieve the rowid, and use it to access the table data block containing the one row.

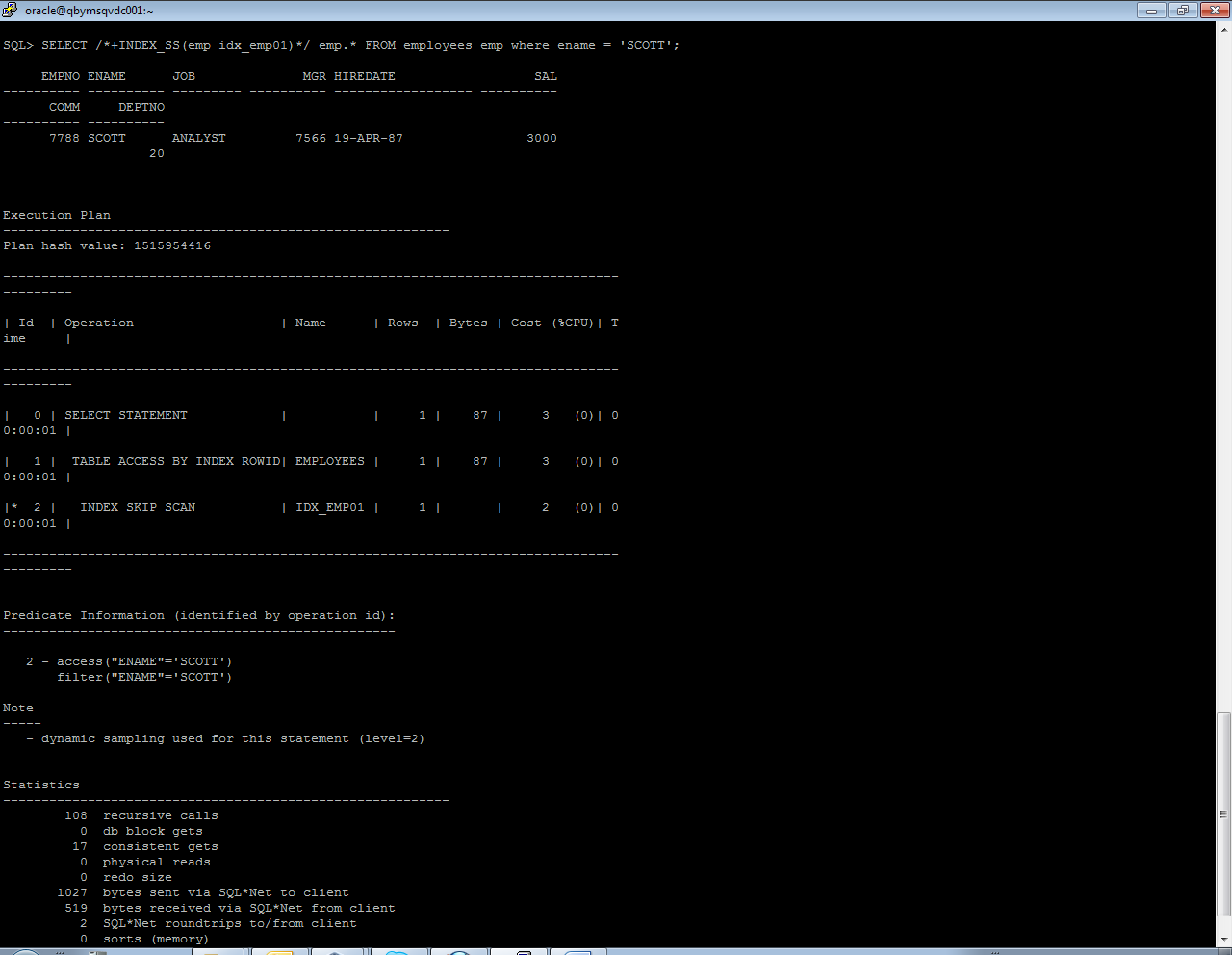
Task 4.

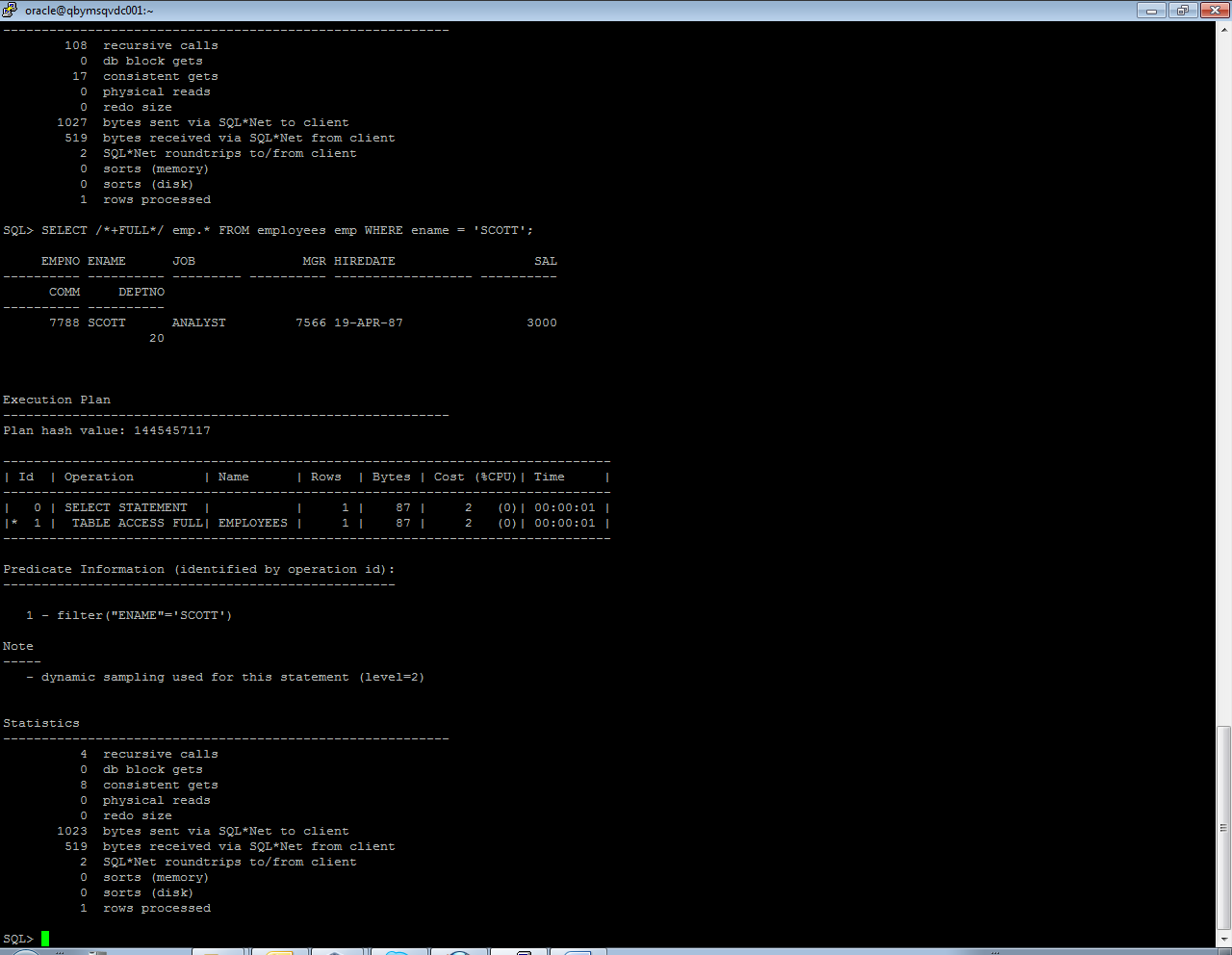


An index range scan is chosen when a predicate contains a condition that will return a range of data. The index can be unique or non-unique as it is the condition that determines whether or not multiple rows will be returned or not.

Task 5.

Screenshots of step 3





|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 1664 | 1536 | 99999 | 4 | Range scan |
| 2 | 1664 | 1536 | 99999 | 9 | Index unique |
| 3 | 1664 | 1536 | 99999 | 17 | Index skip |
| 4 | 1664 | 1536 | 99999 | 8 | Index full |

An index skip scan is chosen when the predicate contains a condition on a non-leading column in an index and the leading columns are fairly distinct. A skip scan works by logically splitting a multi-column index into smaller subindexes. The number of logical subindexes is determined by the number of distinct values in the leading columns of the index. Therefore, the more distinct the leading columns are, the more logical subindexes would need to be created. If too many subindexes would be required, the operation won’t be as efficient as simply doing a full scan.