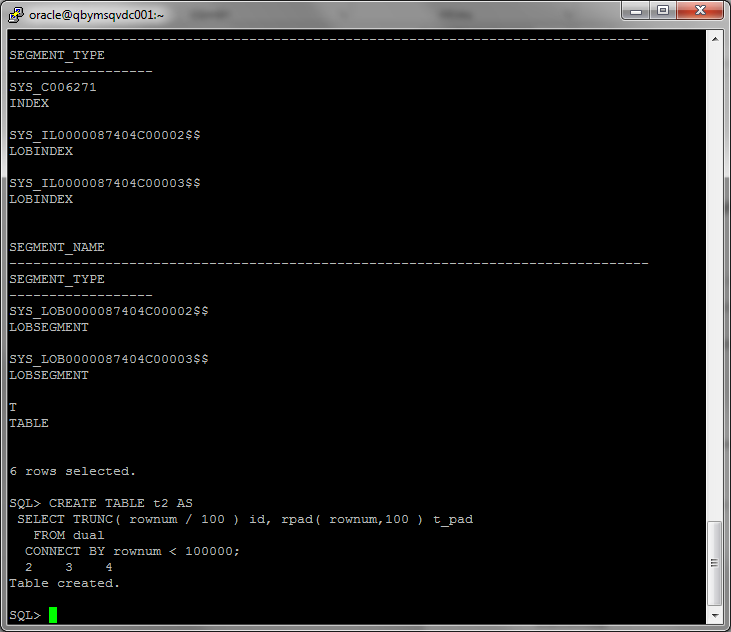
**Report “Lab 4”**

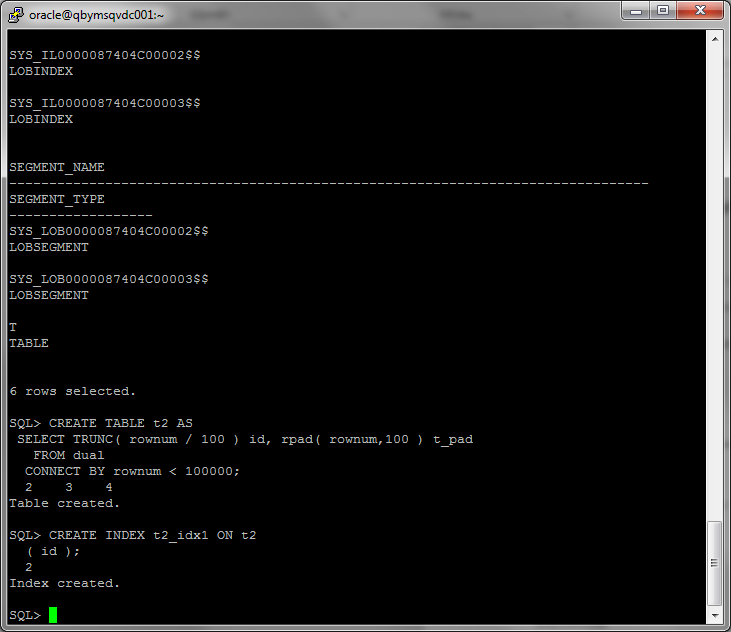
**Anton Tserakhau**

# Table access full scan

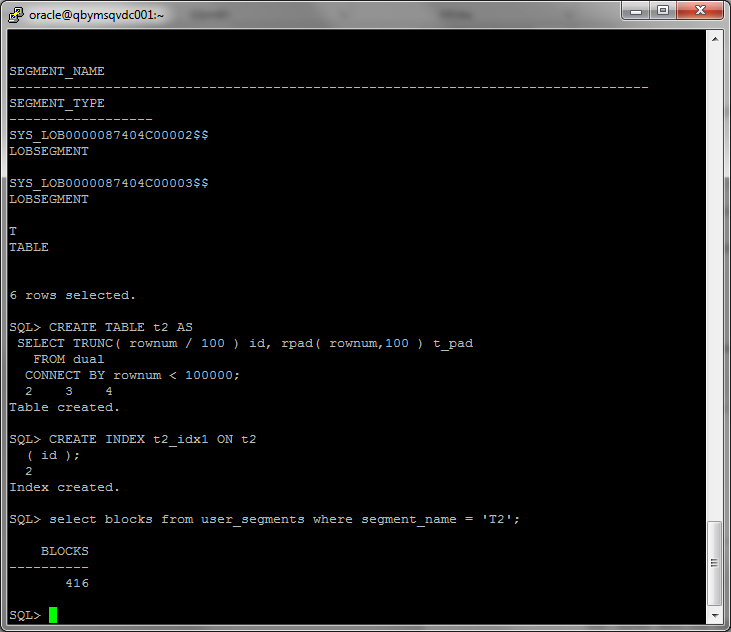
## Task 1: Full Scans and the High-water Mark and Block reading

1. I have created table and index:

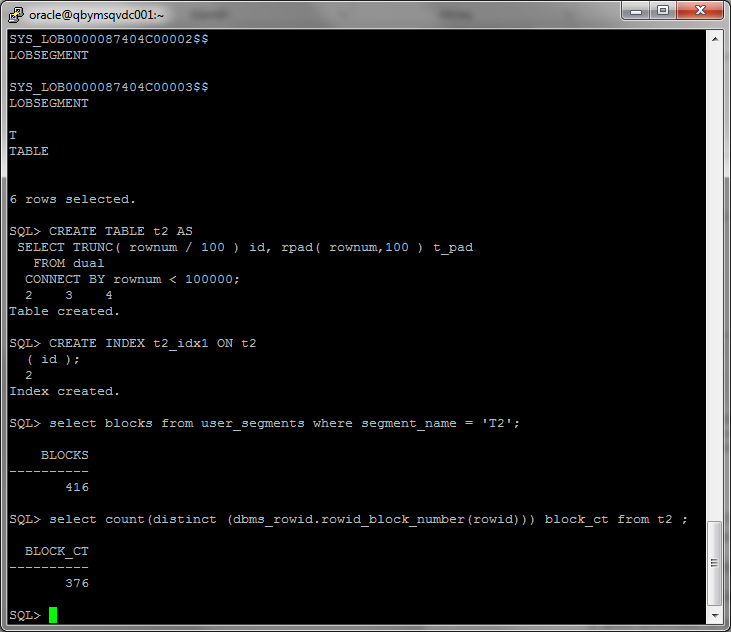




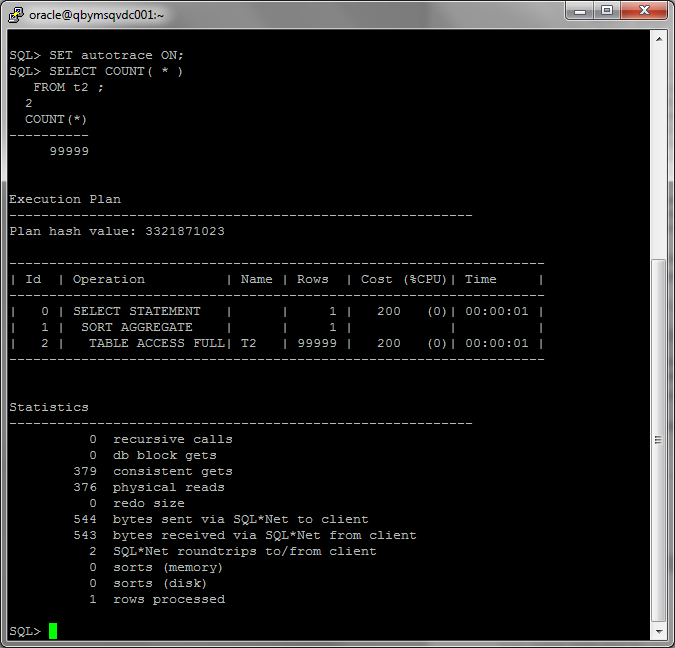
Block count:



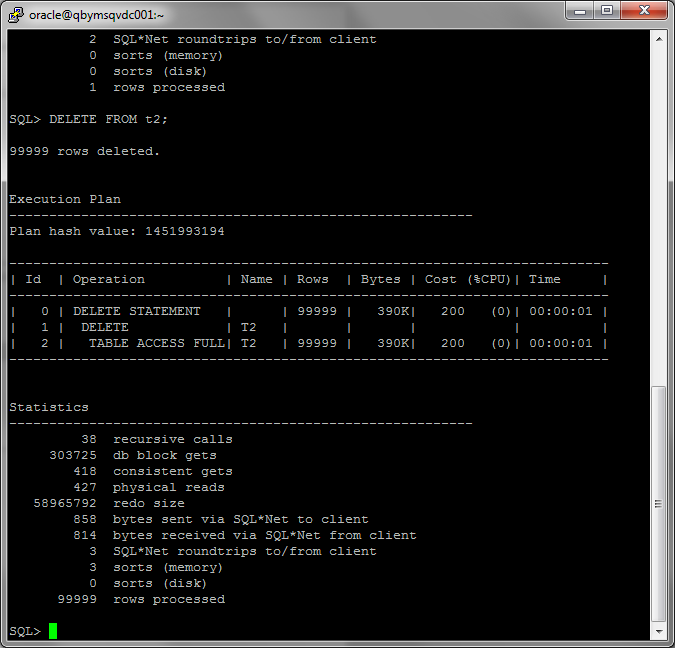
Used block count:



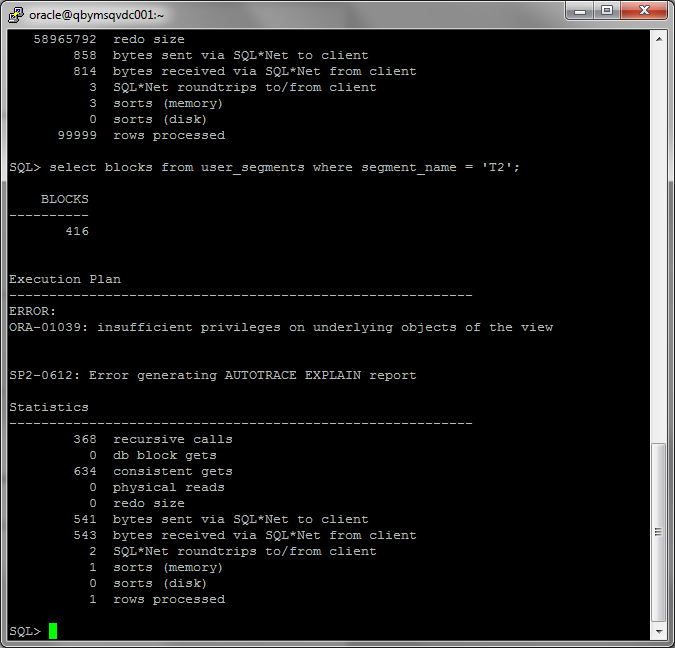
Result of query (count rows into table) and statistics by this query:



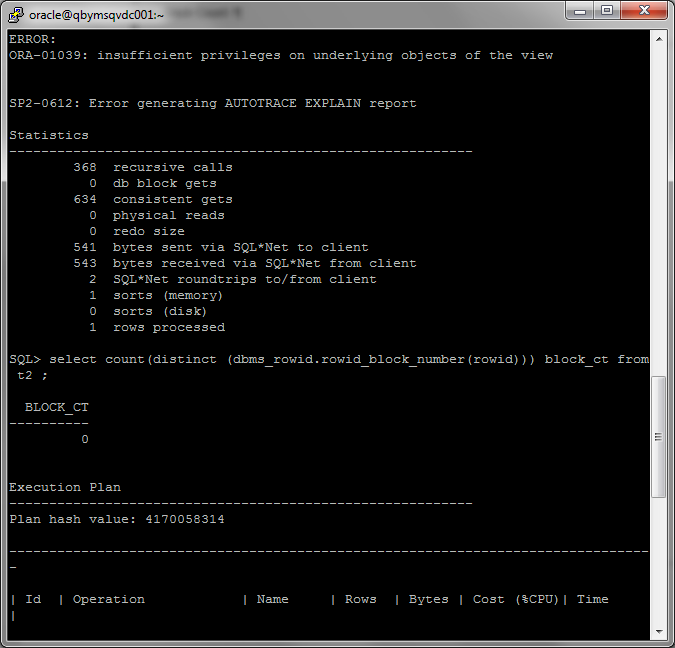
1. I have deleted all rows from table:



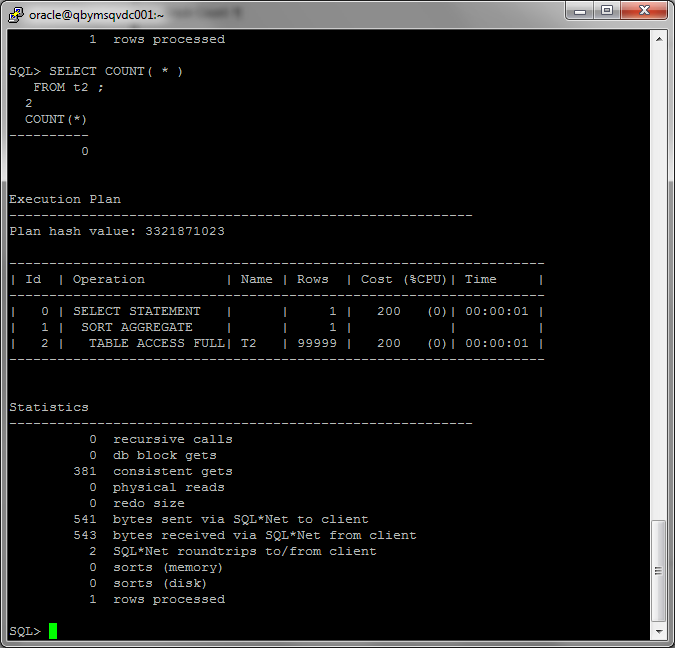
Block count:



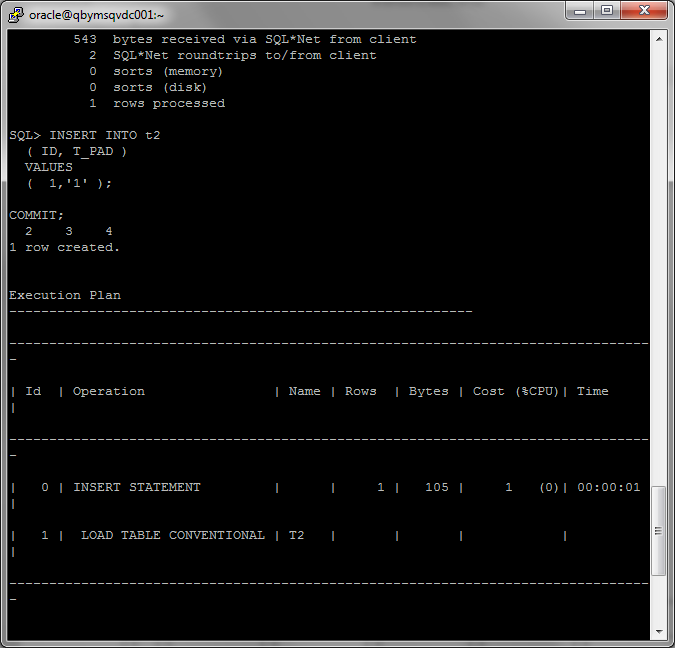
Used block count:



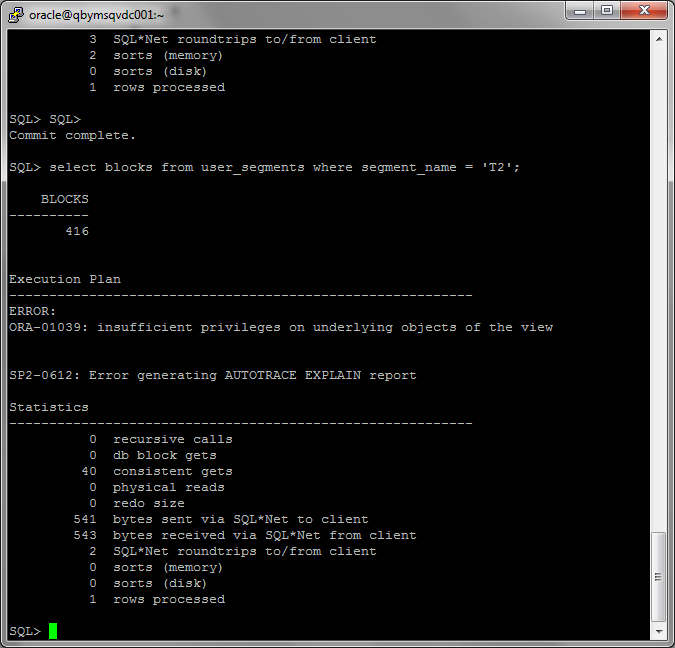
Result of query (count rows into table) and statistics by this query:



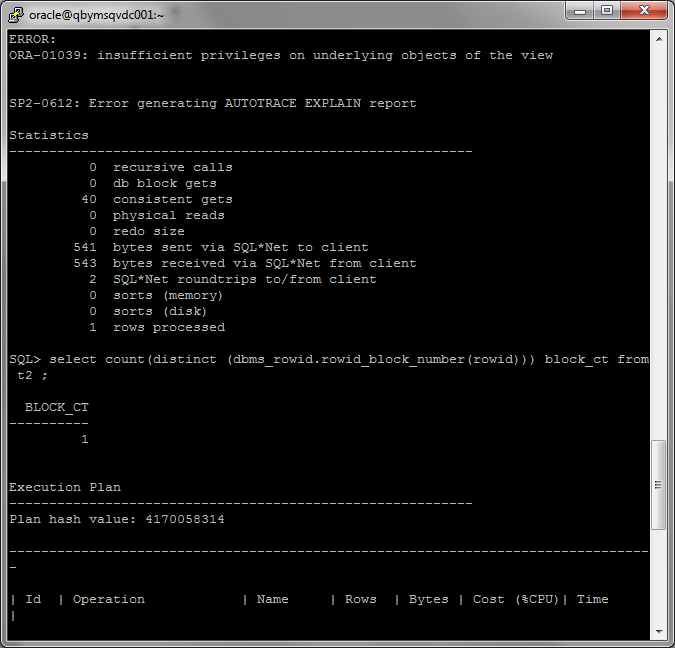
1. I have inserted 1 row into table:



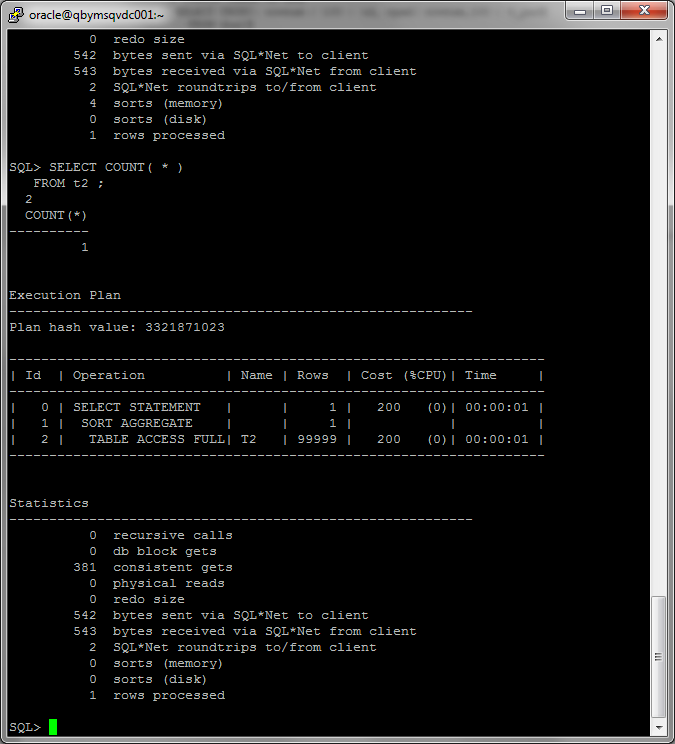
Block count:



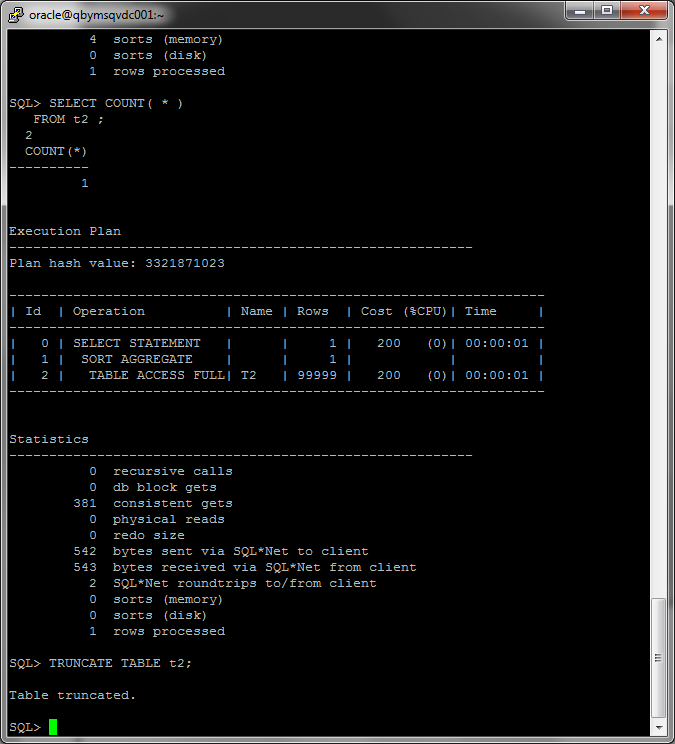
Used block count:



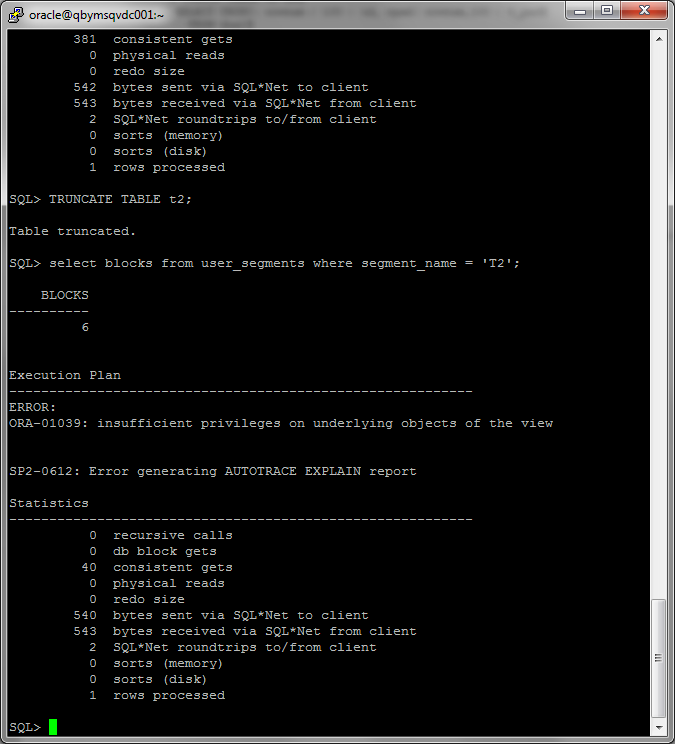
Result of query (count rows into table) and statistics by this query:



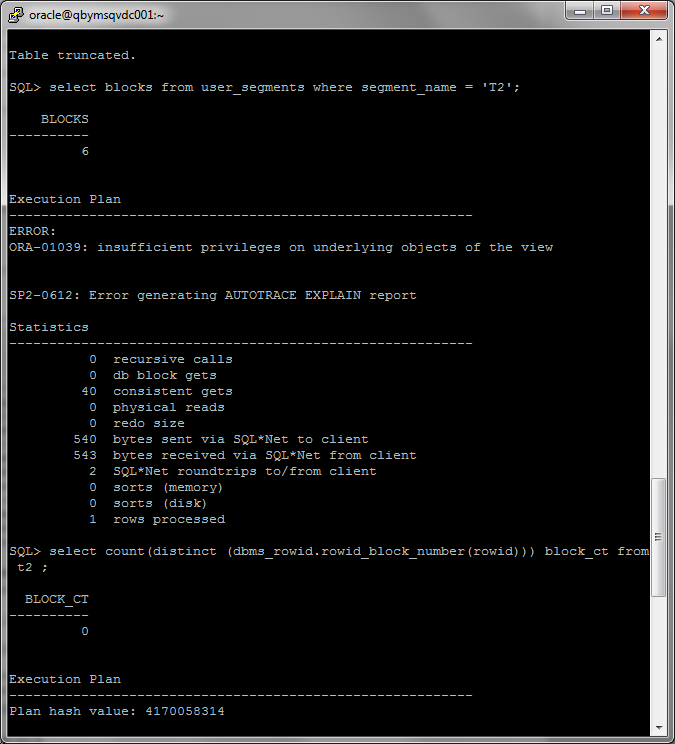
1. I have truncated table:



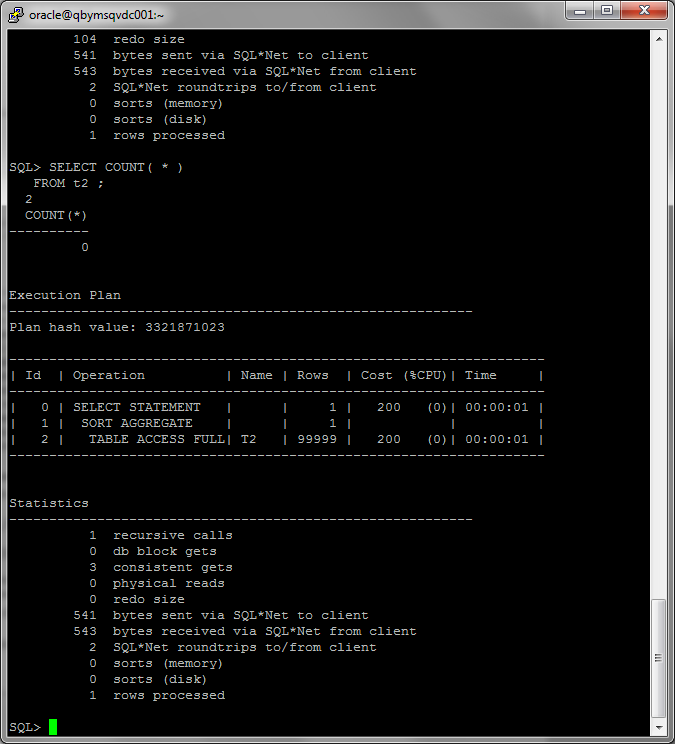
Block count:



Used block count:



Result of query (count rows into table) and statistics by this query:



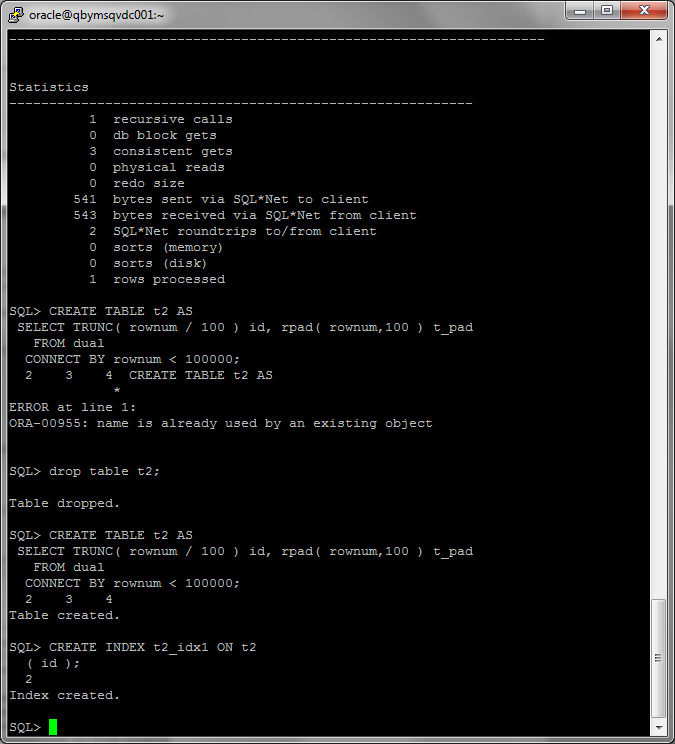
Summary table with all result and text description of analyses this results.

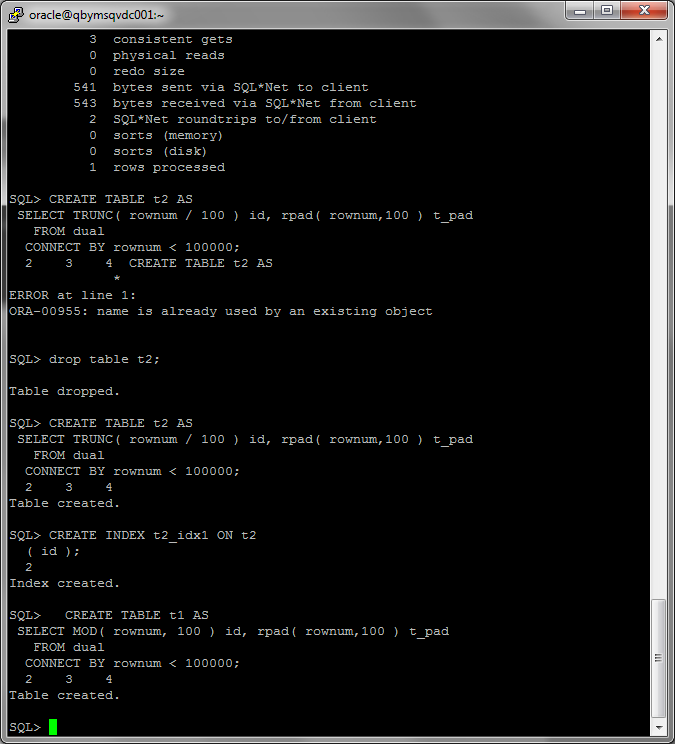
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 1 | 416 | 376 | 99999 | 379 | Count of Blocks ≈ Count of Used Blocks, because we use all blocks for storage 99999 rows. We needed Consistent gets ≈ Count of Blocks, because system have allocated 416 blocks. |
| 2 | 416 | 0 | 0 | 381 | Count of Used Blocks = 0, because we store 0 rows. We needed Consistent gets ≈ Count of Blocks, because system have allocated 416 blocks and these blocks are still allocated system. |
| 3 | 416 | 1 | 1 | 381 | Count of Used Blocks = 1, because we store 1 rows, the block is not completely filled. We needed Consistent gets ≈ Count of Blocks, because system have allocated 416 blocks and these blocks are still allocated system. |
| 4 | 6 | 0 | 0 | 3 | Count of Blocks = 6 (for system information), Count of Used Blocks = 0, because we store 0 rows. We needed Consistent gets (3) ≈ Count of Blocks, because system have released unused blocks. |

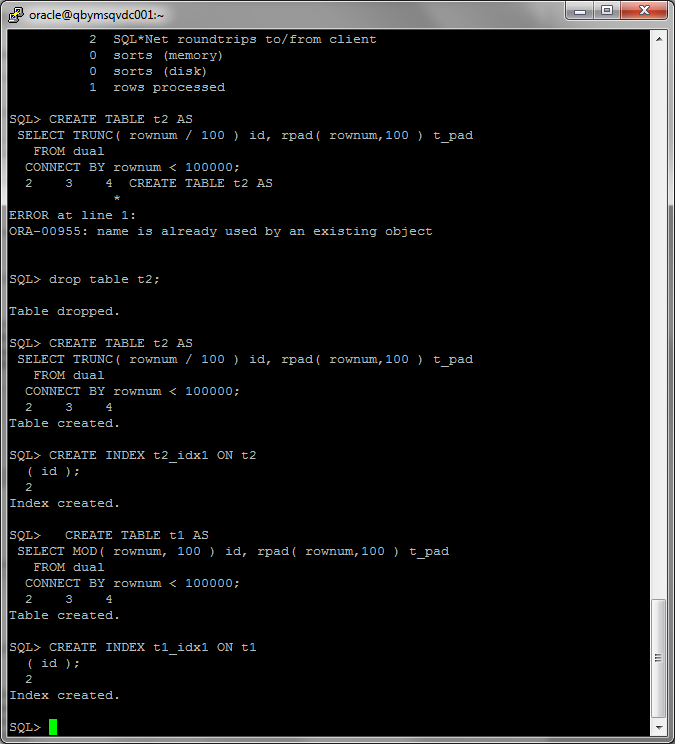
# Index Scan types

## Task 2: Index Clustering factor parameter

I have created tables and indexes:



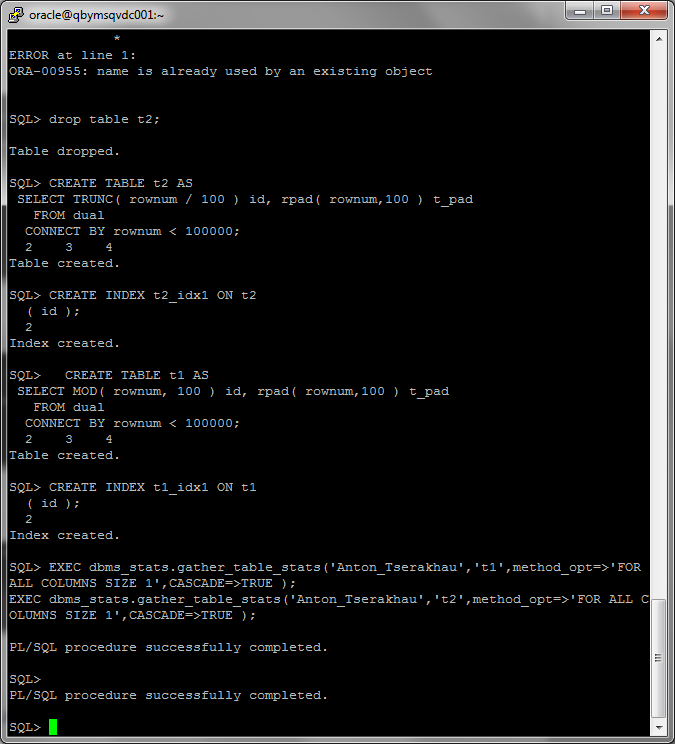




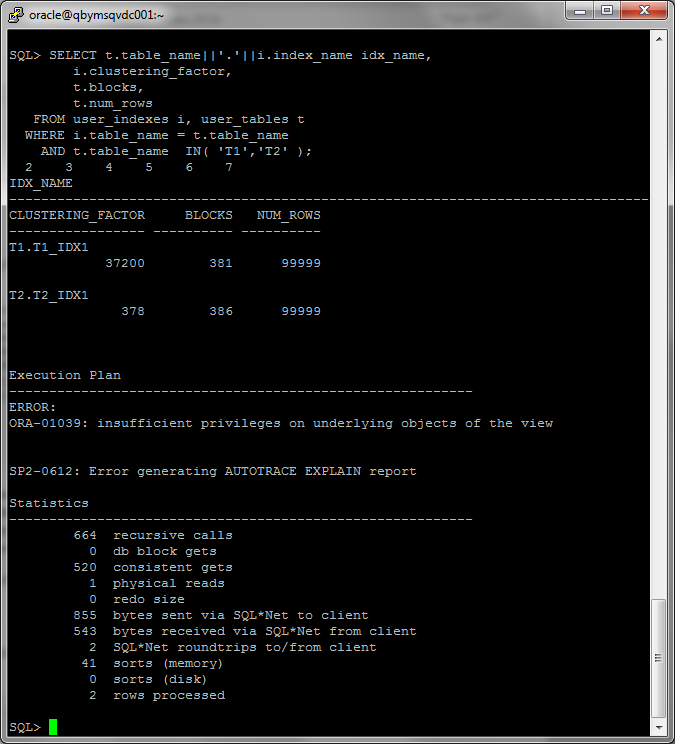
I have calculated statistic for both tables:

EXEC dbms\_stats.gather\_table\_stats('Anton\_Tserakhau','t1',method\_opt=>'FOR ALL COLUMNS SIZE 1',CASCADE=>TRUE );

EXEC dbms\_stats.gather\_table\_stats('Anton\_Tserakhau','t2',method\_opt=>'FOR ALL COLUMNS SIZE 1',CASCADE=>TRUE );



I have selected Clustering Factor:



Description of the parameter clustering factor:

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. In this case, the index entries in a single leaf block tend to point to rows in the same data blocks.
* If the value is near the number of rows, then the table is very randomly ordered. In this case, it is unlikely that index entries in the same leaf block point to rows in the same data blocks.

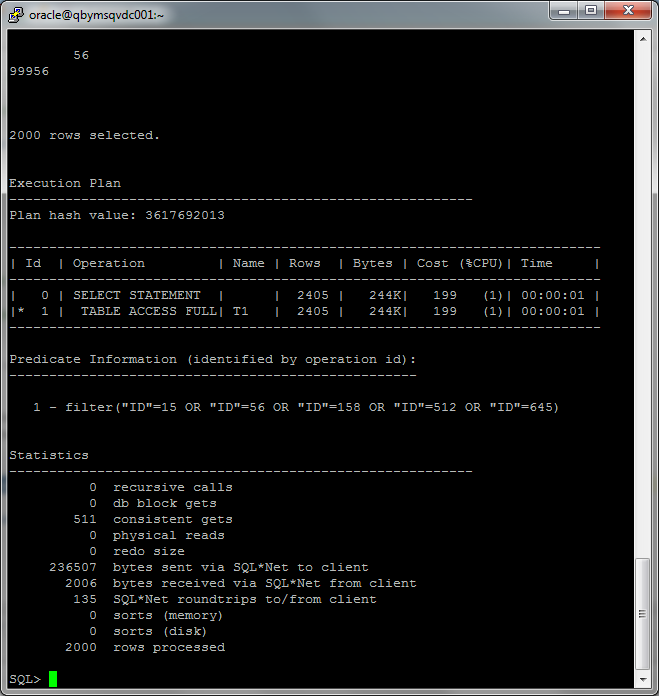
The Clustering Factor measures how synchronized an index is with the data in a table. A table with a high clustering factor is out-of-sequence with the rows and large index range scans will consume lots of I/O. Conversely, an index with a low Clustering Factor is closely aligned with the table and related rows reside together of each data block, making indexes very desirable for optimal access.

Explanation:

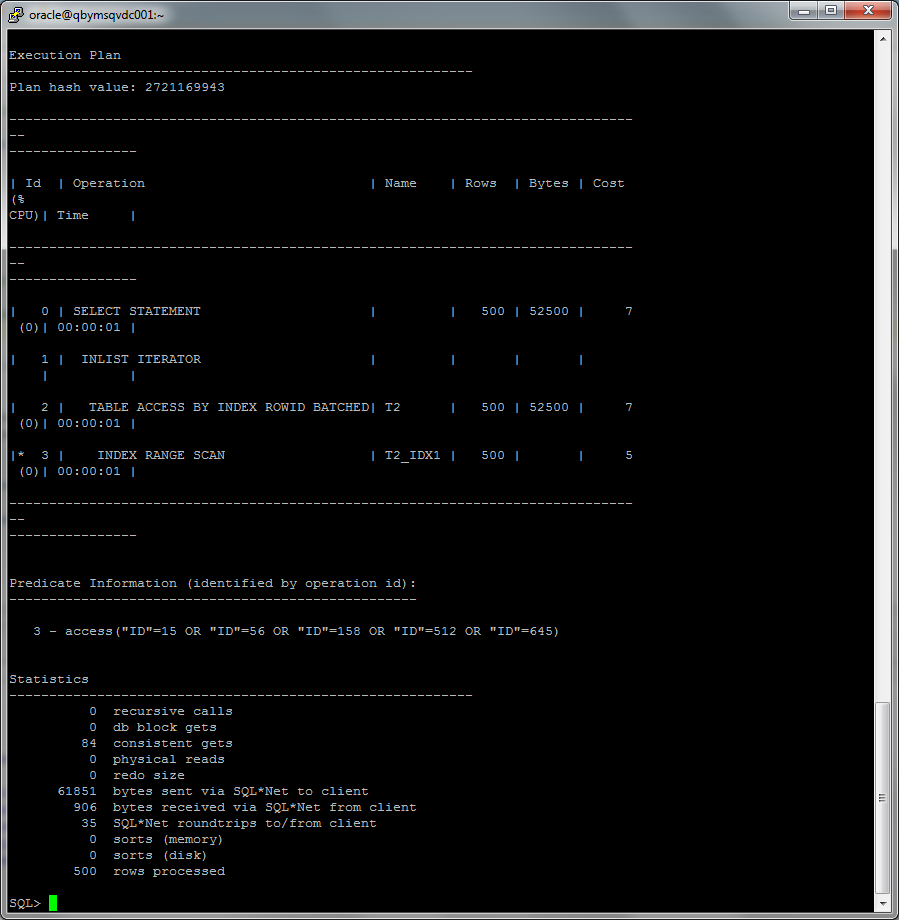
For indexes t1\_idx1 and t2\_idx1 we have different values, because data in first index have stored sorted by groups (0,1,2,…100,0,1,2…..100), data in second index have stored sorted (0,0,0,……,0,1,1,1…..100). So range scan is fast for second index, because range scans through the index structure, if it discovers the next row in the index is on the same database block as the prior row, it does not perform another I/O to get the table block from the buffer cache

Which Index has best selective performance in execution *Select clause filtered by IN ( , list of values, ):*

Select \* from t1 where id in (15, 56, 158, 512, 645);



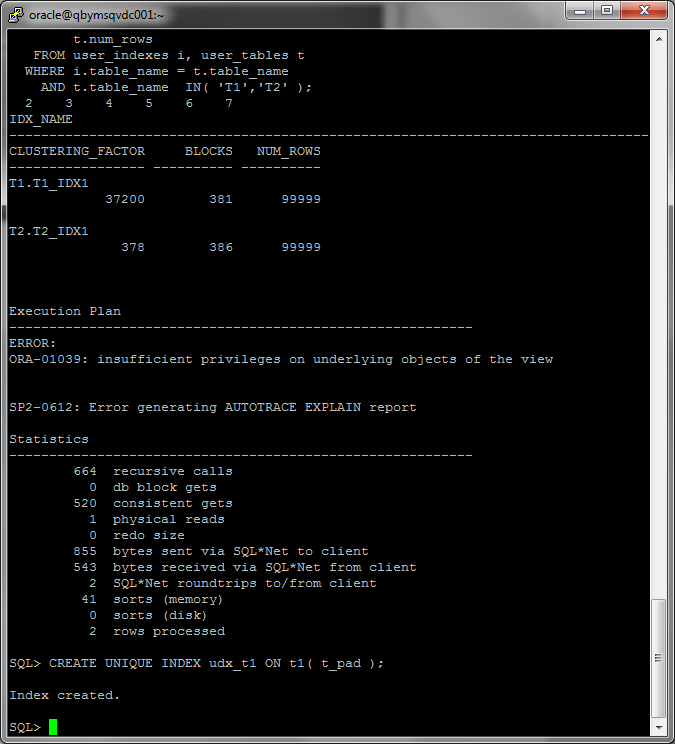
Select \* from t2 where id in (15, 56, 158, 512, 645);



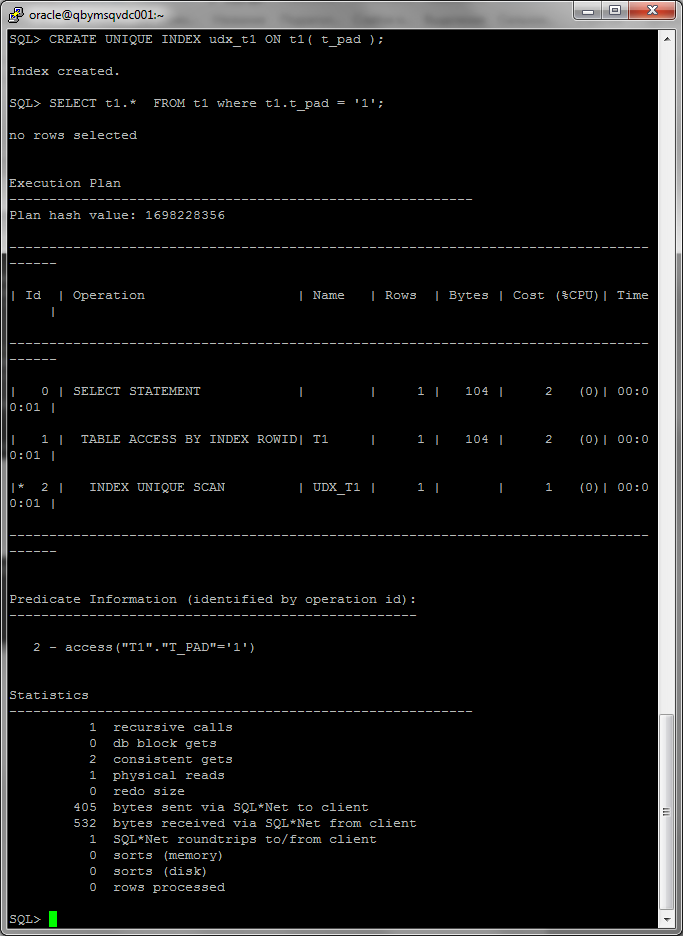
Cost query with index range scan << cost query with full access table, so index range scan has best selective performance in execution *Select clause filtered by IN ( , list of values, ).*

## Task 3: Index Unique Scan

I have created unique index:



Result of query that uses index unique scan:

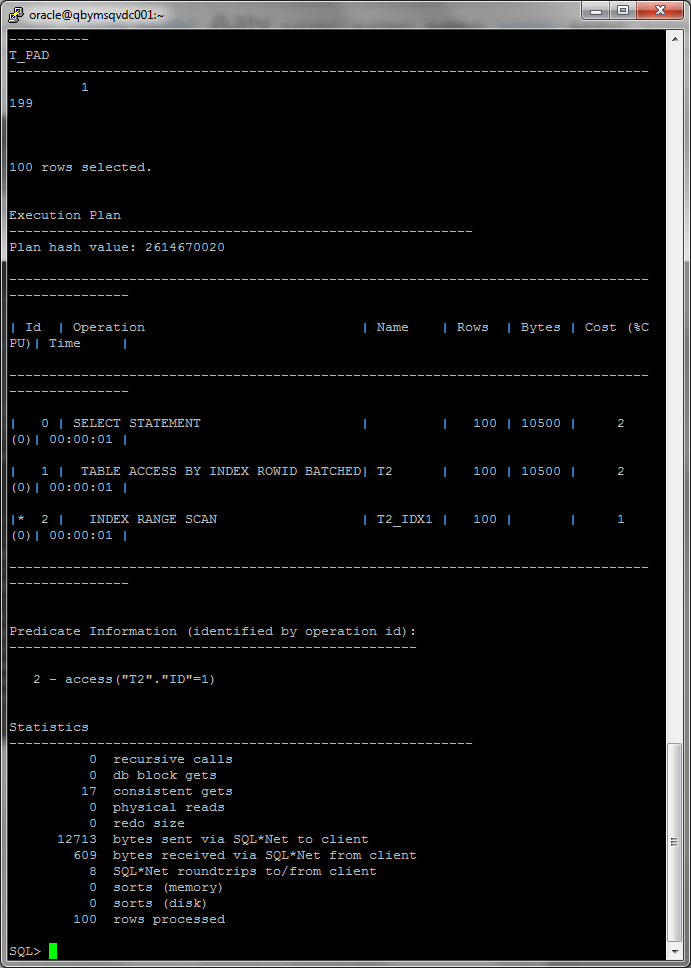


Description of process:

Index unique scan must have either 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

Execution plan and statistics of query that uses index range scan:



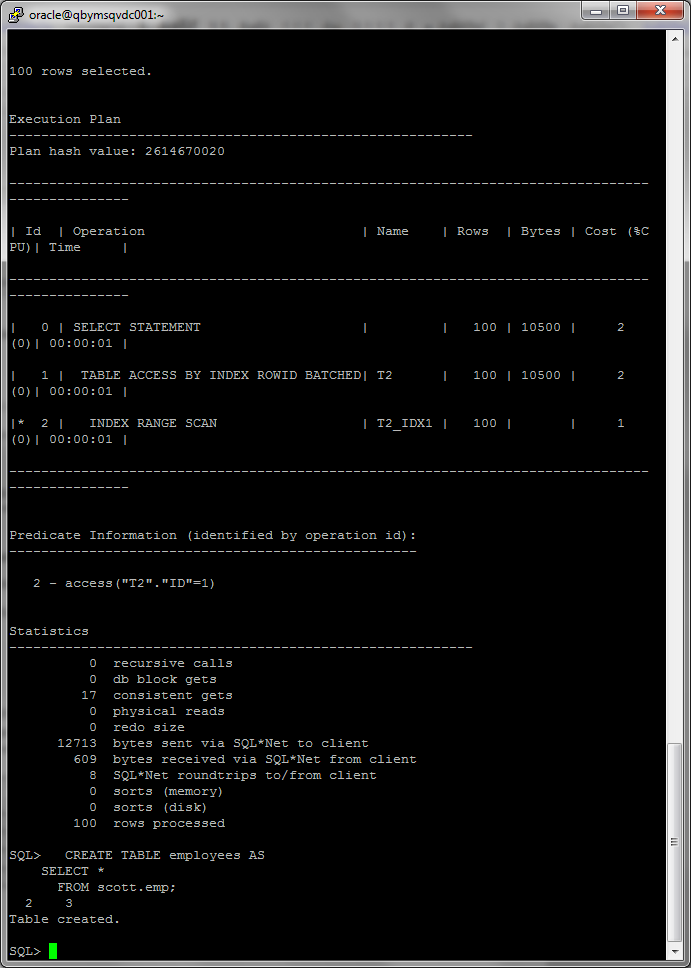
Description of process:

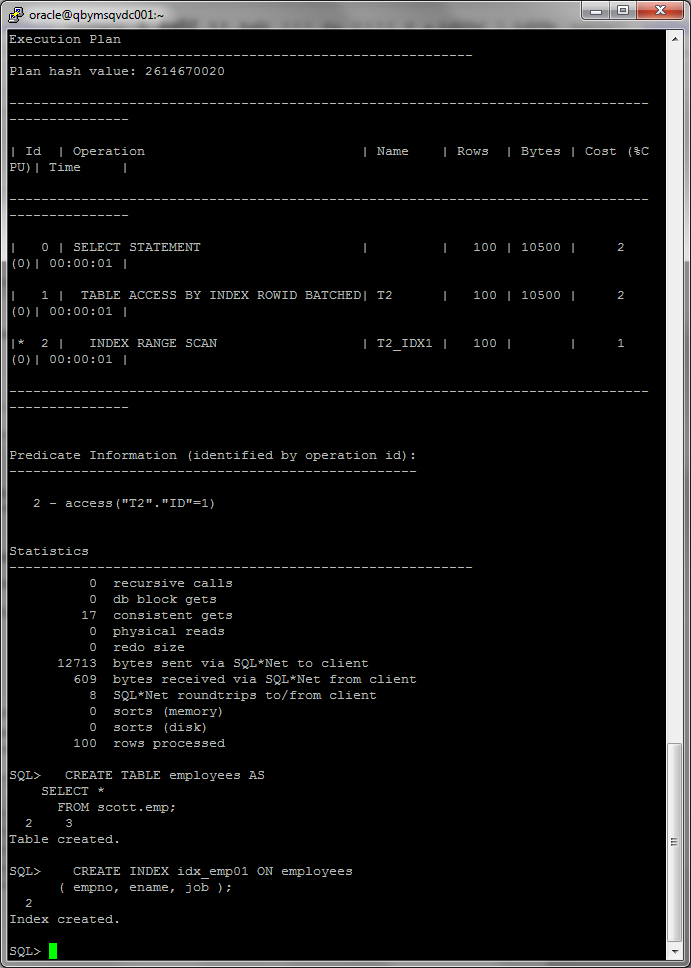
Is chosen when a predicate contains a condition that will return a range of data. There are cases when predicates that you might think should use index range scans do not.

An index range scan is an ordered scan of an index. To scan the index, the database moves backward or forward through the leaf blocks.

## Task 5: Index Skip Scan

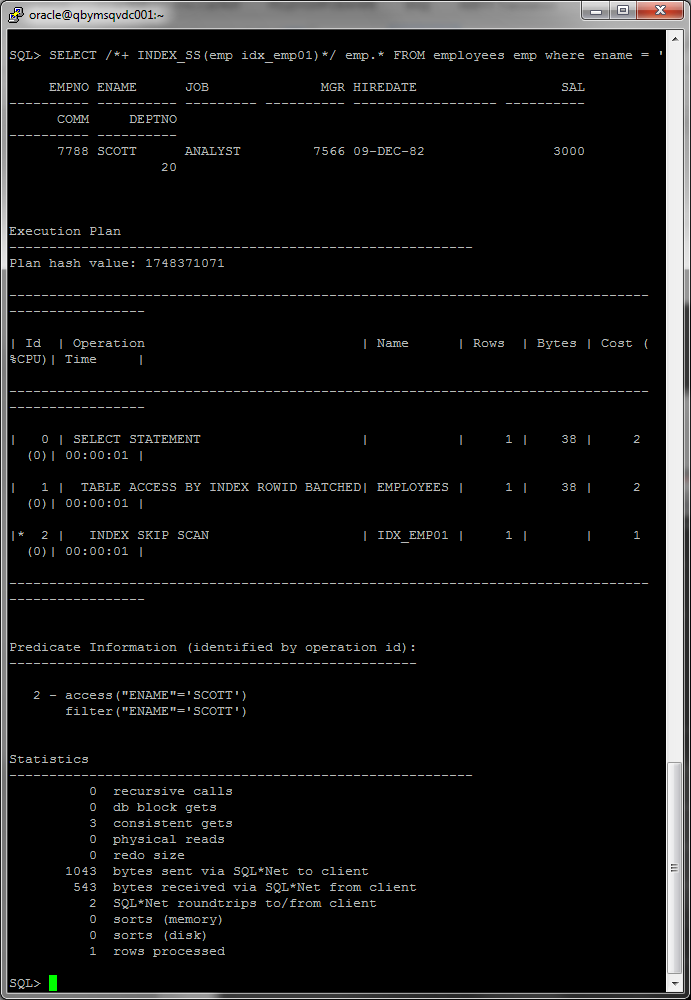
I have created table and index:





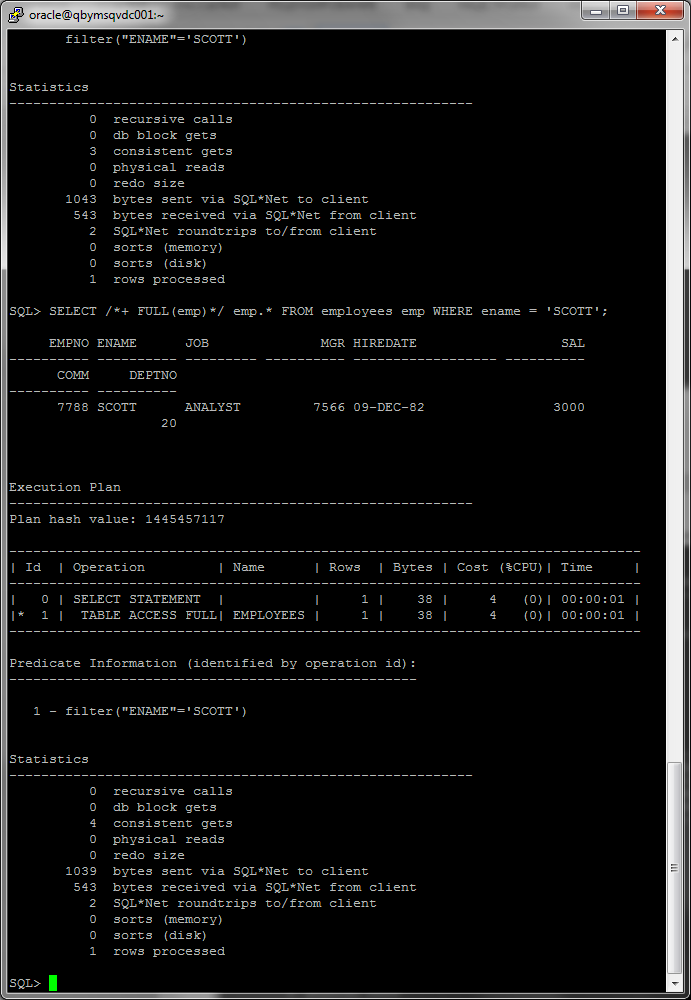
Query with using index:

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



Query with using full scan table

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



Description of process:

An index skip scan uses logical subindexes of a composite index. The database "skips" through a single index as if it were searching separate indexes. Skip scanning is beneficial if there are few distinct values in the leading column of a composite index and many distinct values in the nonleading key of the index.

Summary table with all result and text description of analyses this results:

|  |  |  |
| --- | --- | --- |
| Scan | Cost | Description |
| Index skip scan | 2 | 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. |
| Full scan table | 4 | When a full scan operation occurs, all blocks up to the highwater mark will be read in and scanned, even if they are empty. This means that many blocks that don’t need to be read because they are empty will still be read. |