# Table access full scan

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

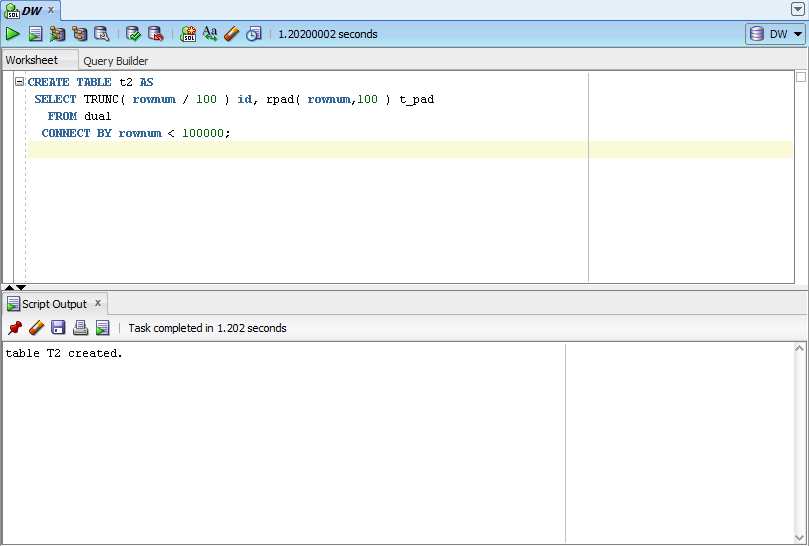
Step 1:

# CREATE TABLE t2 AS

SELECT TRUNC( rownum / 100 ) id, rpad( rownum,100 ) t\_pad

FROM dual

CONNECT BY rownum < 100000;



Step 2:

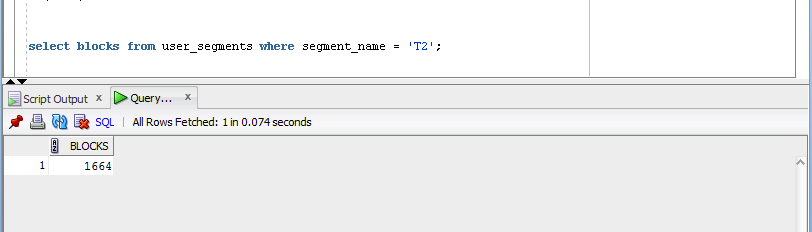
# CREATE INDEX t2\_idx1 ON t2

( id );

**Step 3:**

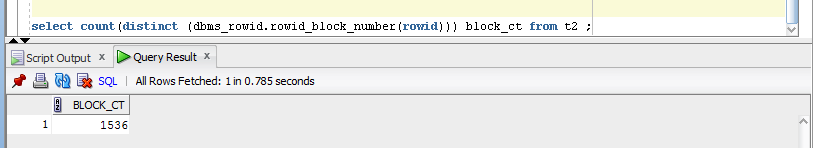
Block count:

# select blocks from user\_segments where segment\_name = 'T2';



Used Block Count:

# select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from t2 ;

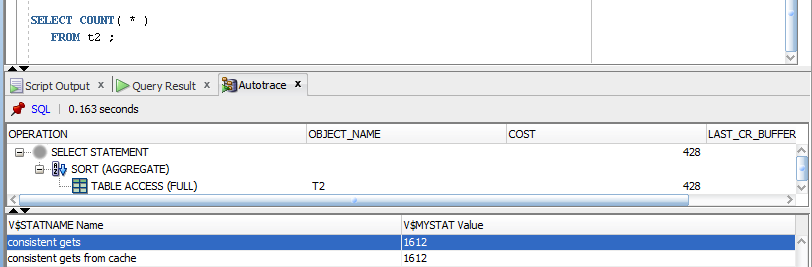


Explain Plan:

# SET autotrace ON;

# SELECT COUNT( \* )

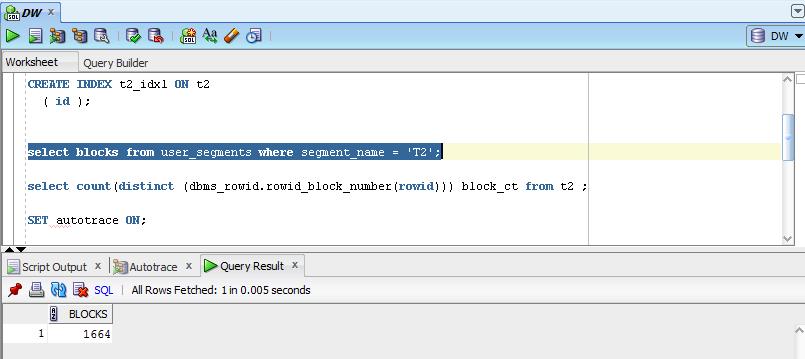
FROM t2 ;

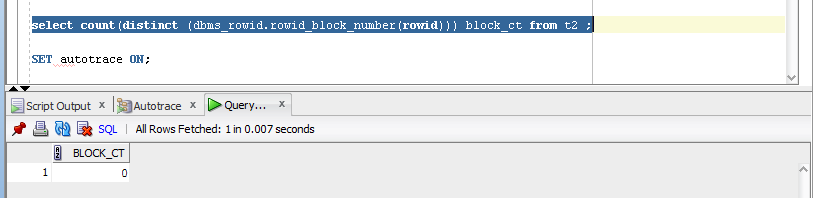


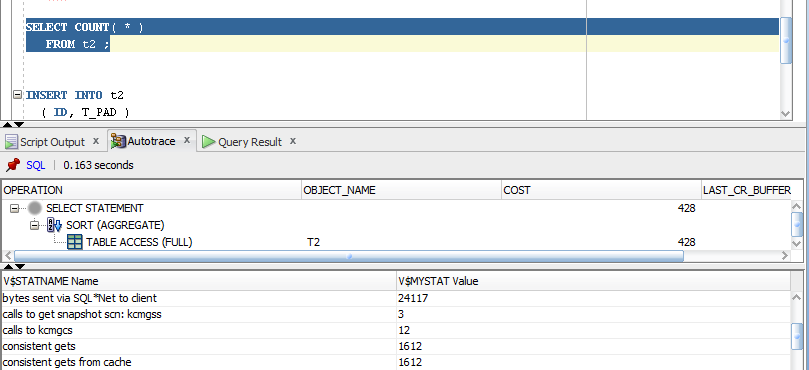
Step 4: Delete All Rows from table

# DELETE FROM t2;

**Step 5:** Repeat Step 3 and collect results.







Step 6: Insert 1 row

# INSERT INTO t2

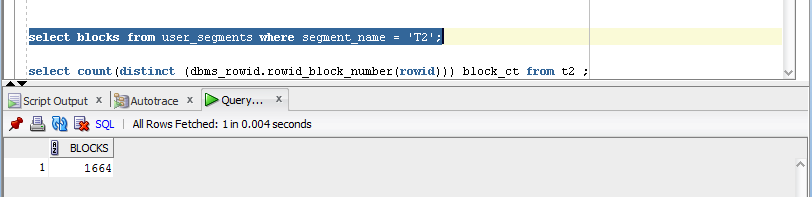
( ID, T\_PAD )

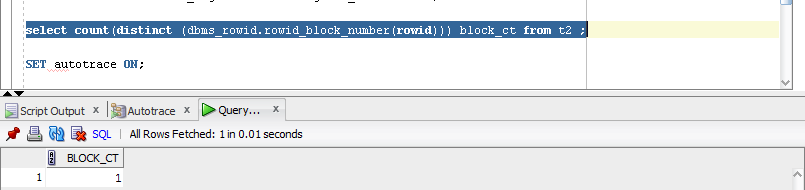
VALUES

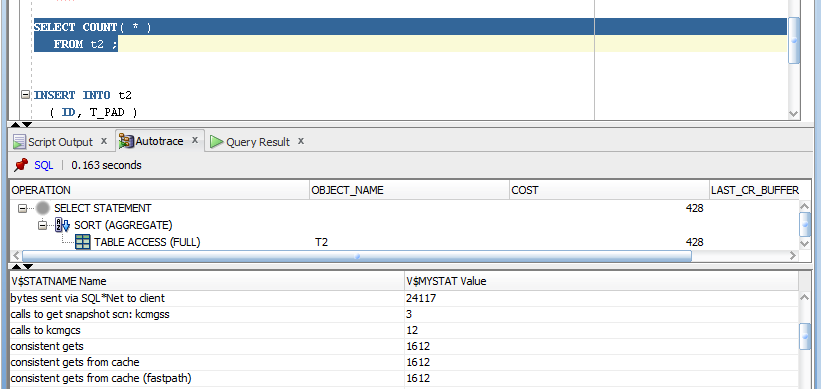
( 1,'1' );

COMMIT;

**Step 7:** Repeat Step 3 and collect results.

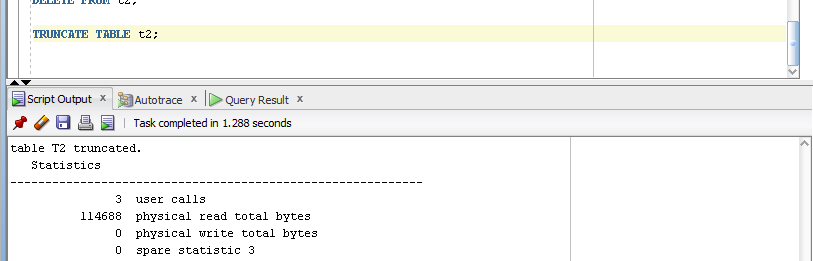




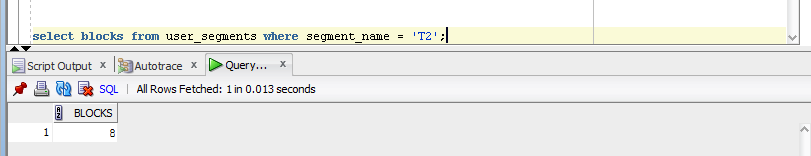


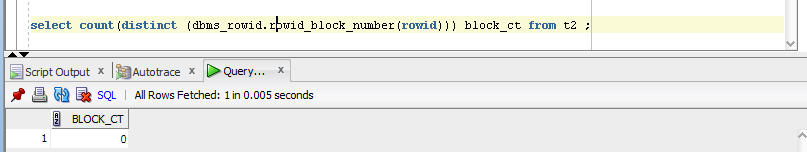
Step 8: Truncate Table

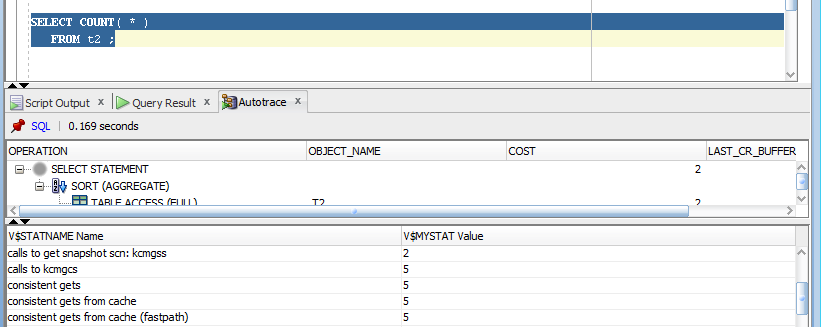
# TRUNCATE TABLE t2;



**Step 9:**  Repeat Step 3 and collect results.







**Task Results:**

Expected:

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 |
| 3 | 1664 | 1536 | 99 999 | 1612 | Create table T2 with 99 999 rows |
| 5 | 1664 | 0 | 0 | 1612 | Delete all from t2 |
| 7 | 1664 | 1 | 1 | 1612 | Insert 1 row to t2 |
| 9 | 8 | 0 | 0 | 5 | Truncate t2 |

After creation of the table with 99 999 rows of random data were allocated 1664 blocks. 1536 of them were used and High-Water mark was on the level of 1612.

After command delete blocks became unused, the data was deleted, but the High-Water mark stayed on the previous level.

After insert Count of Blocks were the same, 1 block was used for data, High-Water mark the same. Only after truncate operation blocks were released and High-Water mark became only 5.

# Index Scan types

## Task 2: Index Clustering factor parameter

Step 1: Create table t2 as on task 1 step 1-2

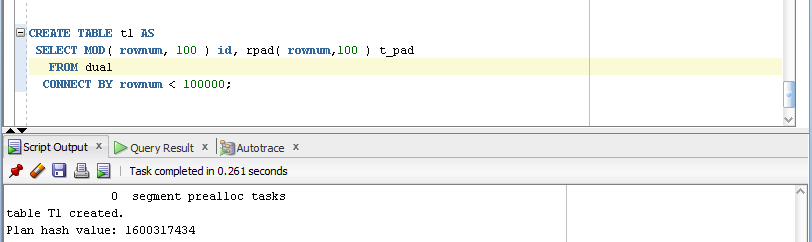
Step 2: Create table t1 as listed below

# CREATE TABLE t1 AS

SELECT MOD( rownum, 100 ) id, rpad( rownum,100 ) t\_pad

FROM dual

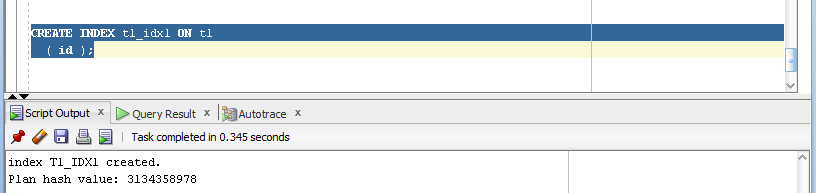
CONNECT BY rownum < 100000;



Step 3:

# CREATE INDEX t1\_idx1 ON t1

( id );



Step 4: Calculate statistic for both tables:

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

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

**Step 5:** Select Clustering Factor

# SELECT t.table\_name||'.'||i.index\_name idx\_name,

i.clustering\_factor,

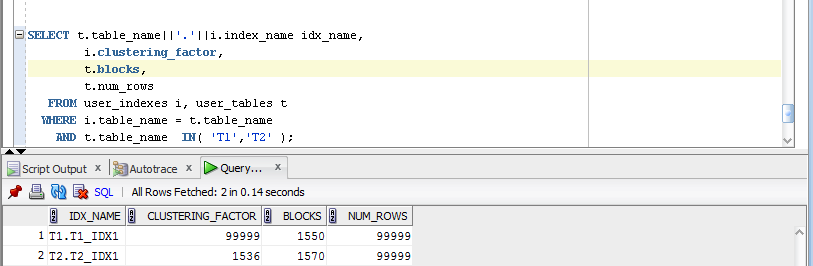
t.blocks,

t.num\_rows

FROM user\_indexes i, user\_tables t

WHERE i.table\_name = t.table\_name

AND t.table\_name IN( 'T1','T2' );



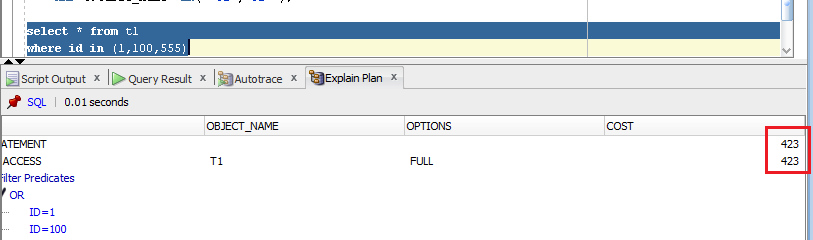
**Task Results:**

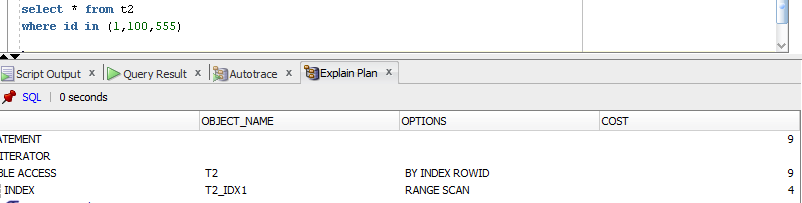
Expected:

* Screenshot of the step 5;
* Description of the parameter clustering factor;
* Explanation: why for indexes t1\_idx1 and t2\_idx1 we have different values ;

We have different values for indexes t1\_idx1 and t2\_idx1 because we use different expressions to calculate id. In t1 we have all id unique, but it is disadvantage because the search of the specified id value row will take more time.

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





t2\_idx1 have better perfomance in searching by id, because we select range of values in fact.

## Task 3: Index Unique Scan

Step 1:

# CREATE UNIQUE INDEX udx\_t1 ON t1( t\_pad );

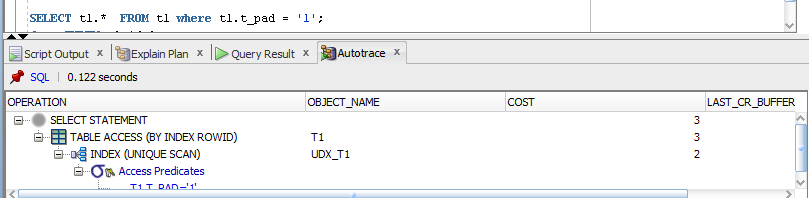
**Step 2**

# SELECT t1.\* FROM t1 where t1.t\_pad = '1';

**Task Results:**

Expected:

* Screenshot of the step 2;



* Description of process: How oracle read block on step 2;

This scan returns, at most, a single rowid. Oracle performs a unique scan if a statement contains a UNIQUE or a PRIMARY KEY constraint that guarantees that only a single row is accessed.

## Task 4: Index Range Scan

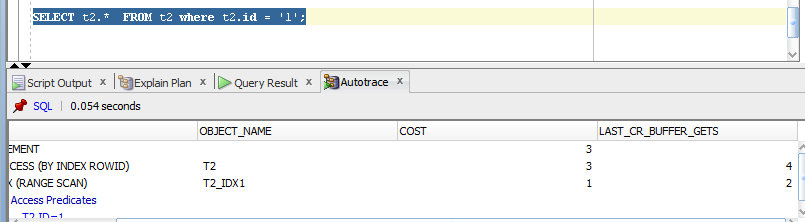
**Step 1:**

# SELECT t2.\* FROM t2 where t2.id = '1';

**Task Results:**

Expected:

* Screenshot of the step 1;



* Description of process: How oracle read block on step 1;

Oracle finds the first index block that contains 1 and then 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

Step 1:

# CREATE TABLE employees AS

SELECT \*

FROM scott.emp;

Step 2:

# CREATE INDEX idx\_emp01 ON employees

( empno, ename, job );

**Step 3:**  Get trace and statistic of explain plan

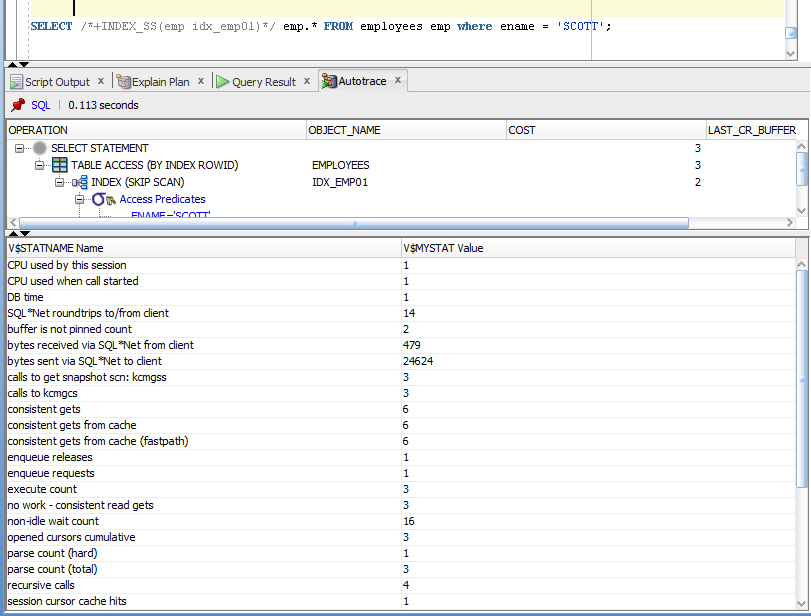
# SELECT /\*+INDEX\_SS(emp idx\_emp01)\*/ emp.\* FROM employees emp where ename = 'SCOTT';

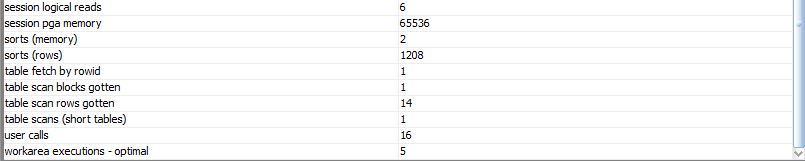
# SELECT /\*+FULL\*/ emp.\* FROM employees emp WHERE ename = 'SCOTT';

**Task Results:**

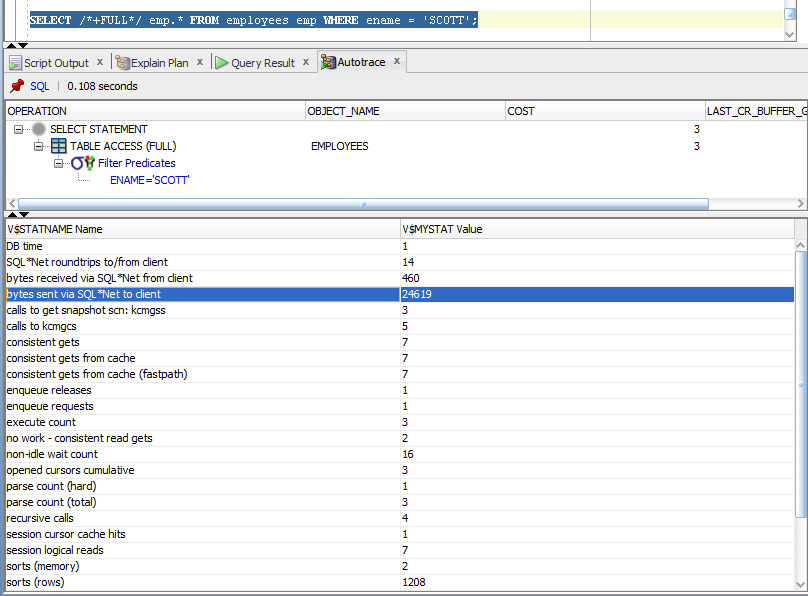
Expected:

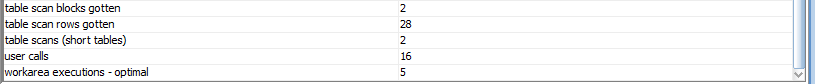
* 2 Screenshots of the step 3;





Skip Scans are initiated by probing the index for distinct values of the prefix column. Each of these distinct values is then used as a starting point for a regular index search. The result is several separate searches of a single index that, when combined, eliminate the affect of the prefix column.





* Description of process: How oracle analyses index that was created on step 2;