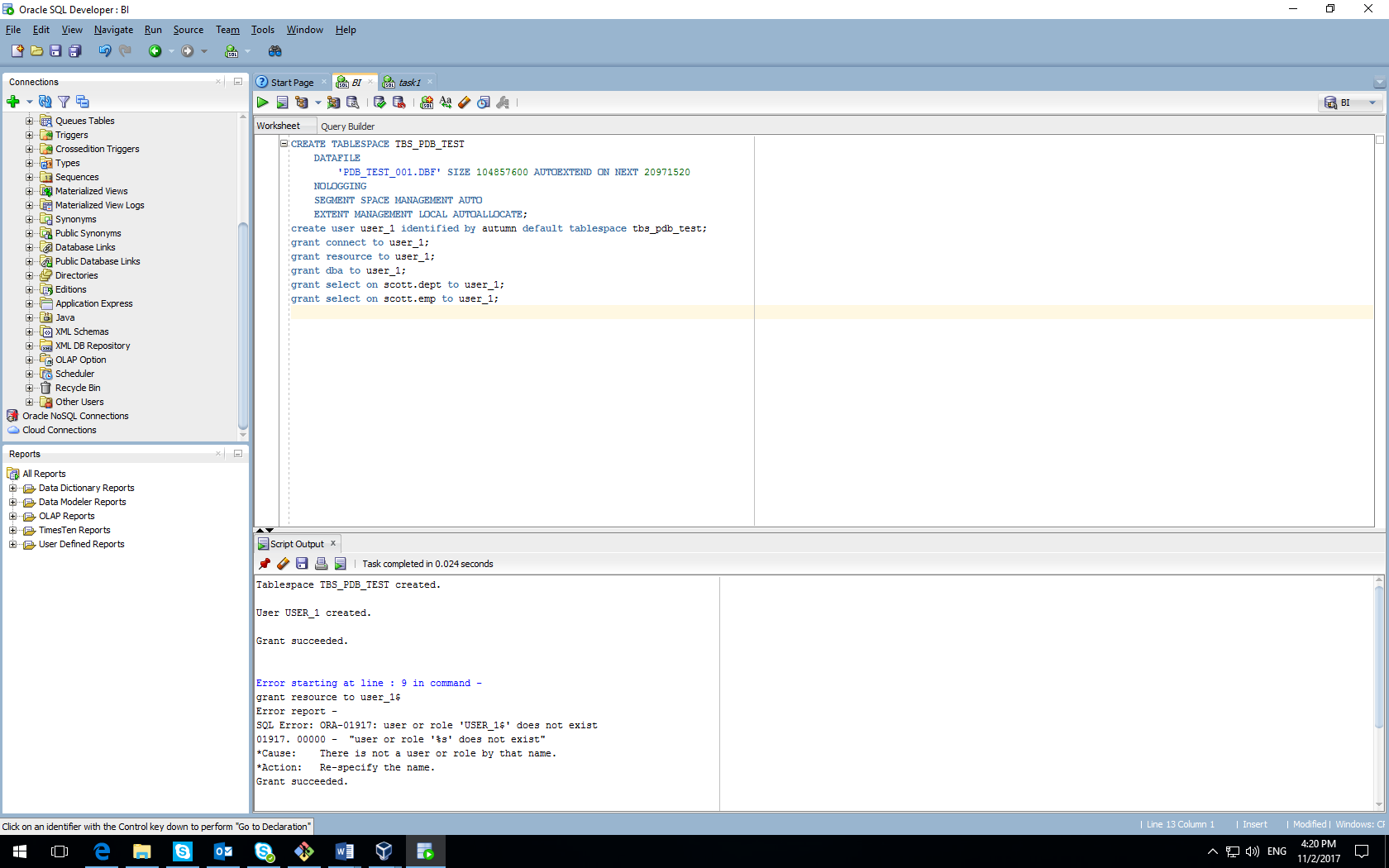
Ex1



Connected as system;

Created new tablespace with new data file pdb\_test\_001.dbf;

Created new user user\_1;

Grant Connect Role and Resource Role;

Ex2.1

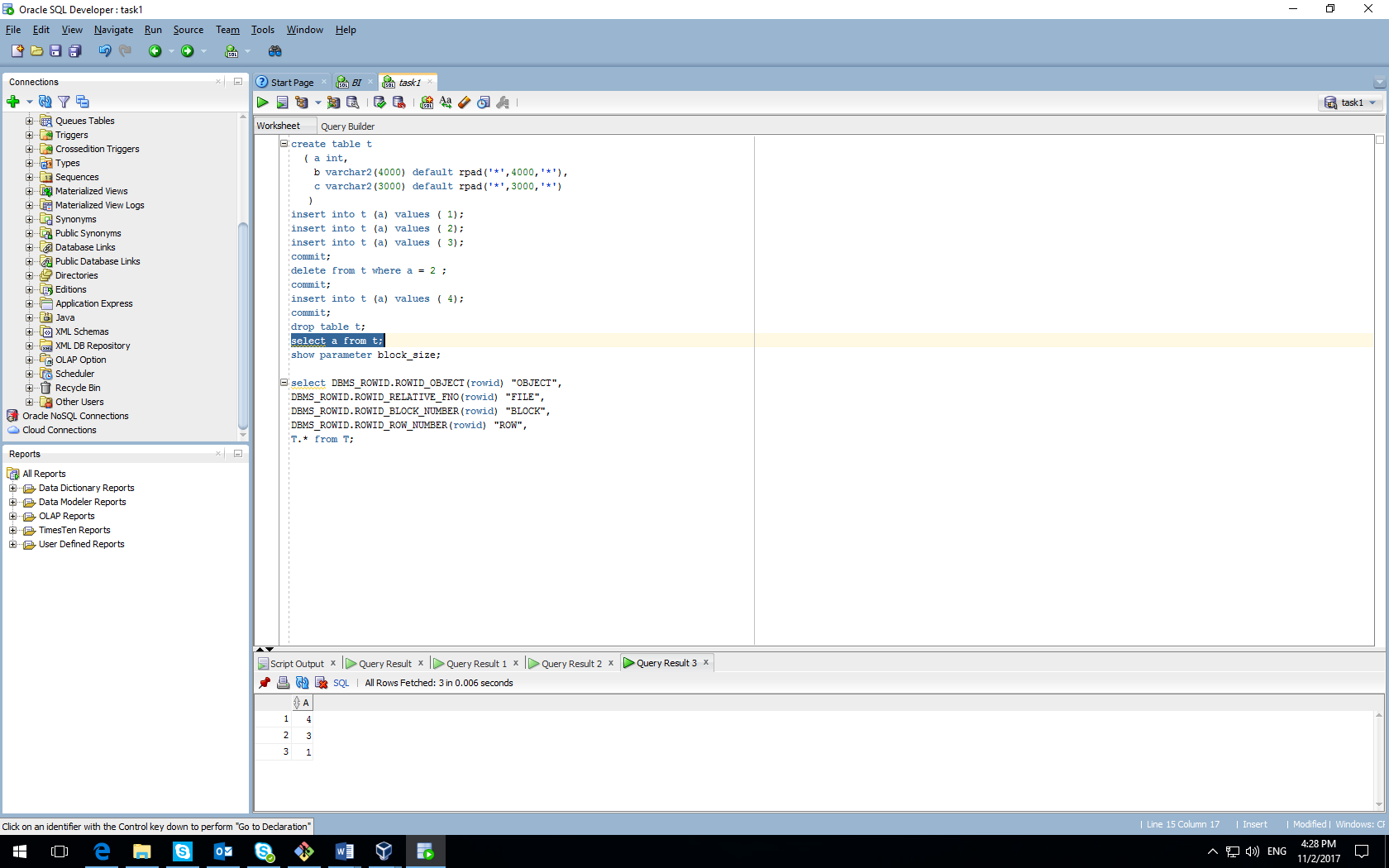
Created table t with values 4000 for column b and 3000 for column c.

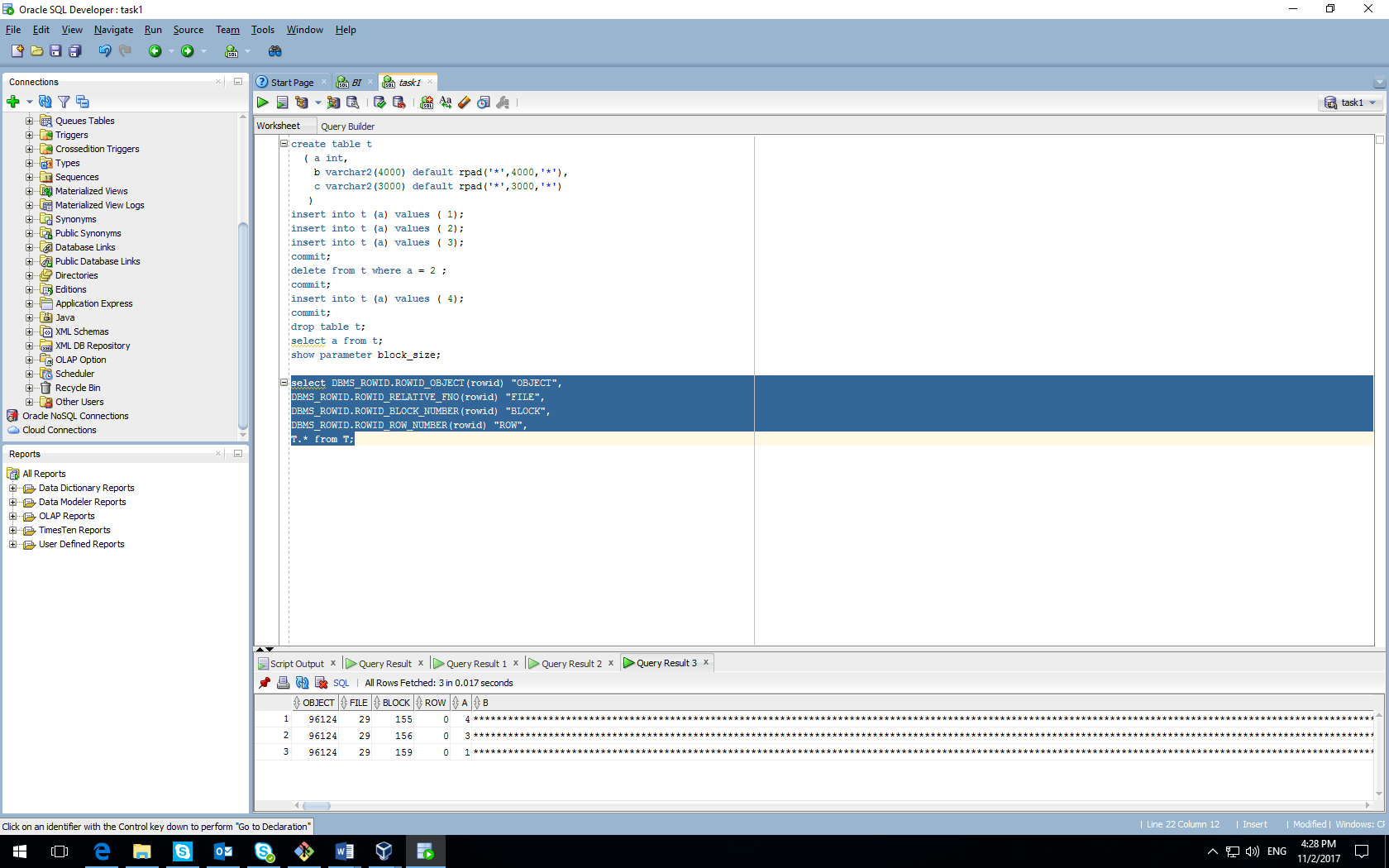
As a result we can see that values inserted into different blocks (155,156,159).

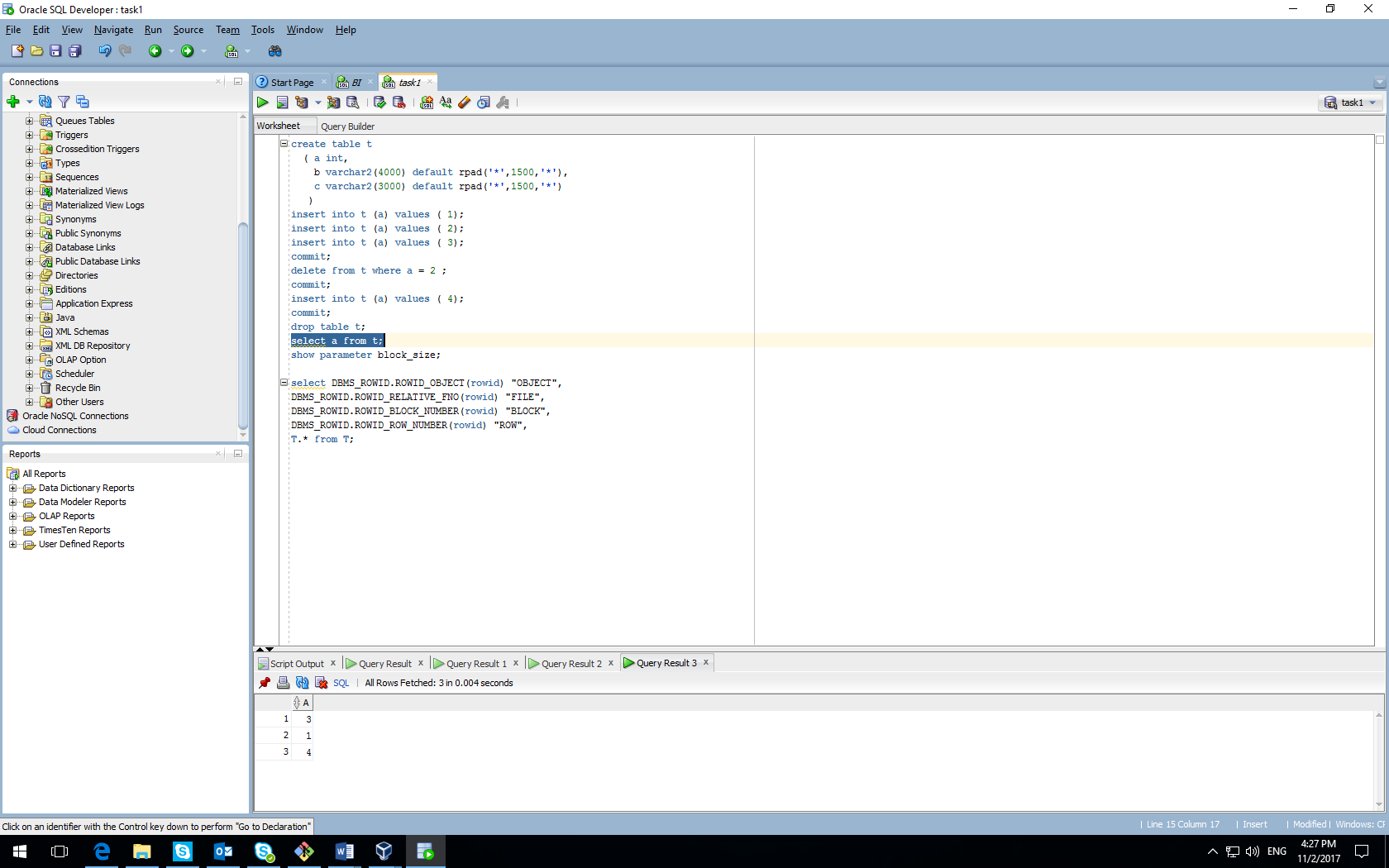
Let’s change values from 4000 to 1500 for column b and from 3000 to 1500 for column c.

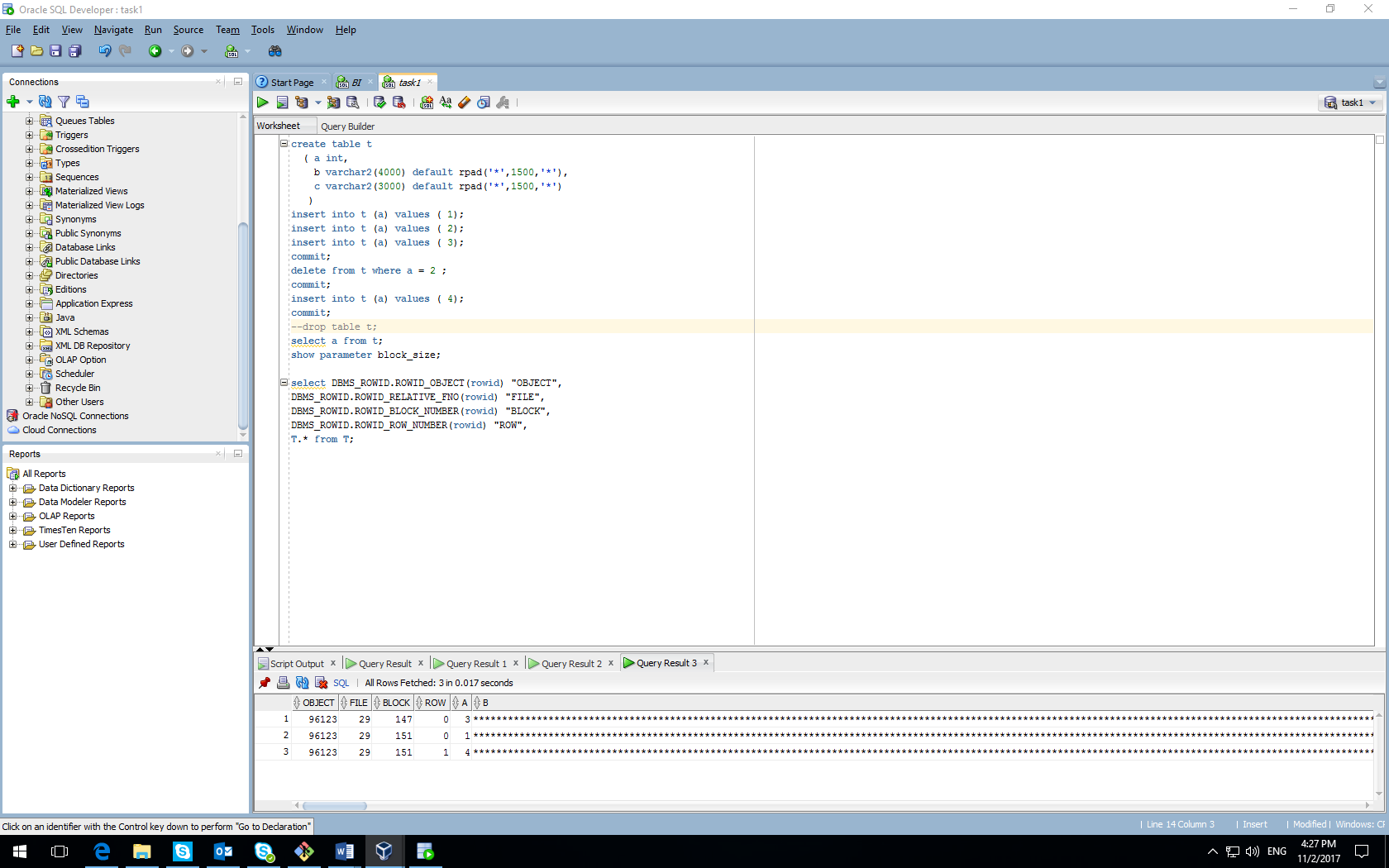
As a result we can see that values 1 and 4 inserted into one block (151).

Because of this changes row order is changed from 4,3,1 to 3,1,4





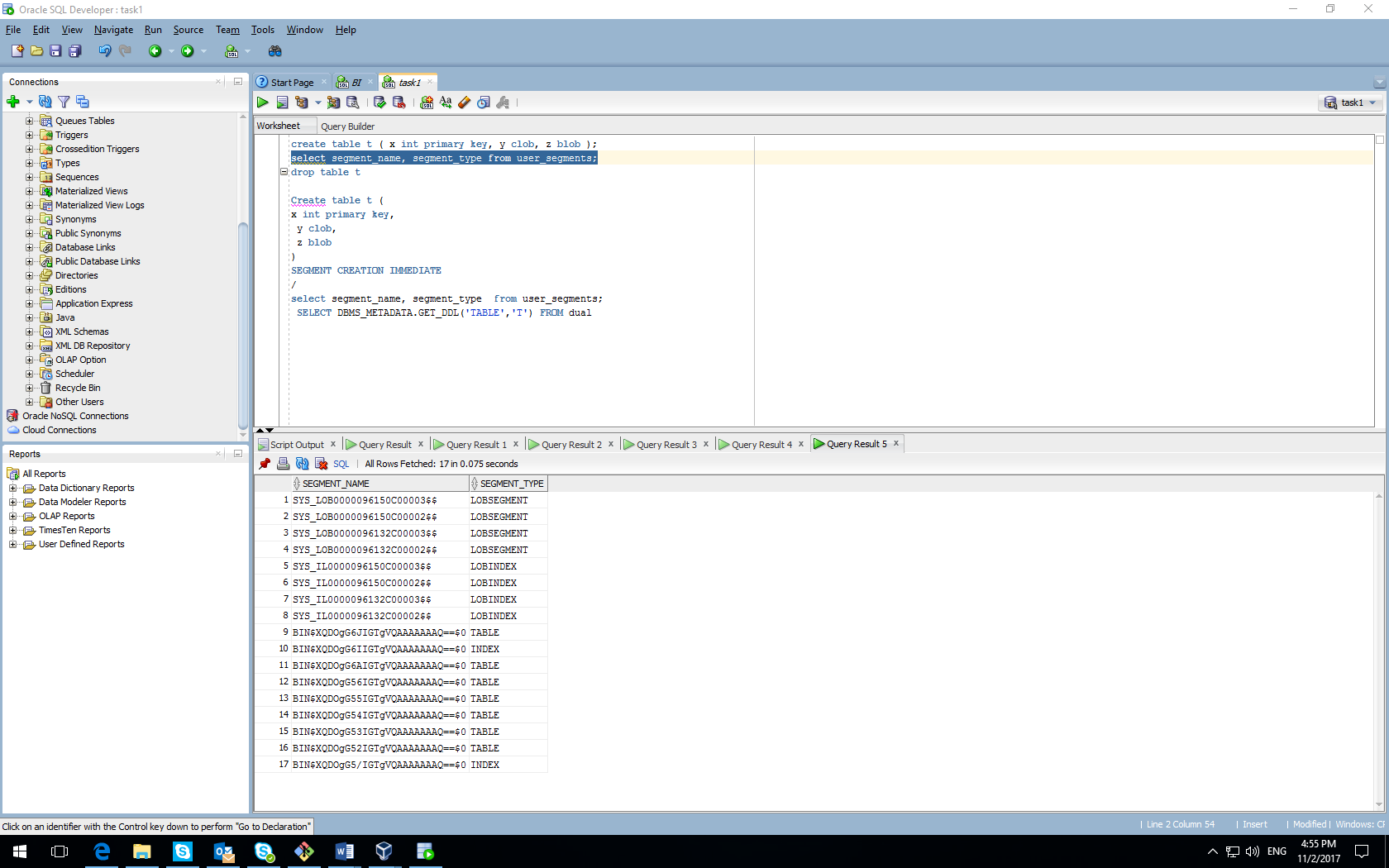


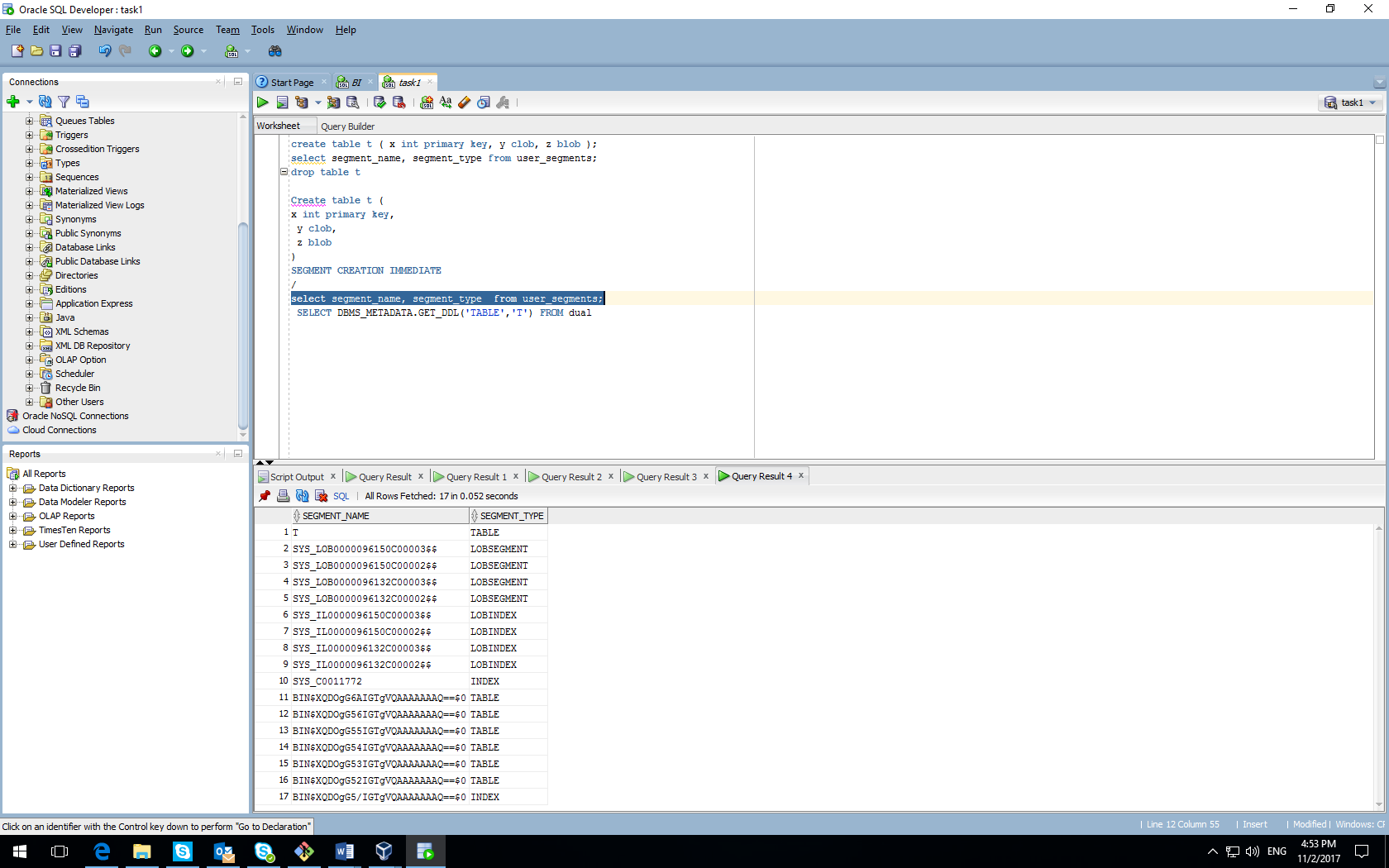


Ex 2.3

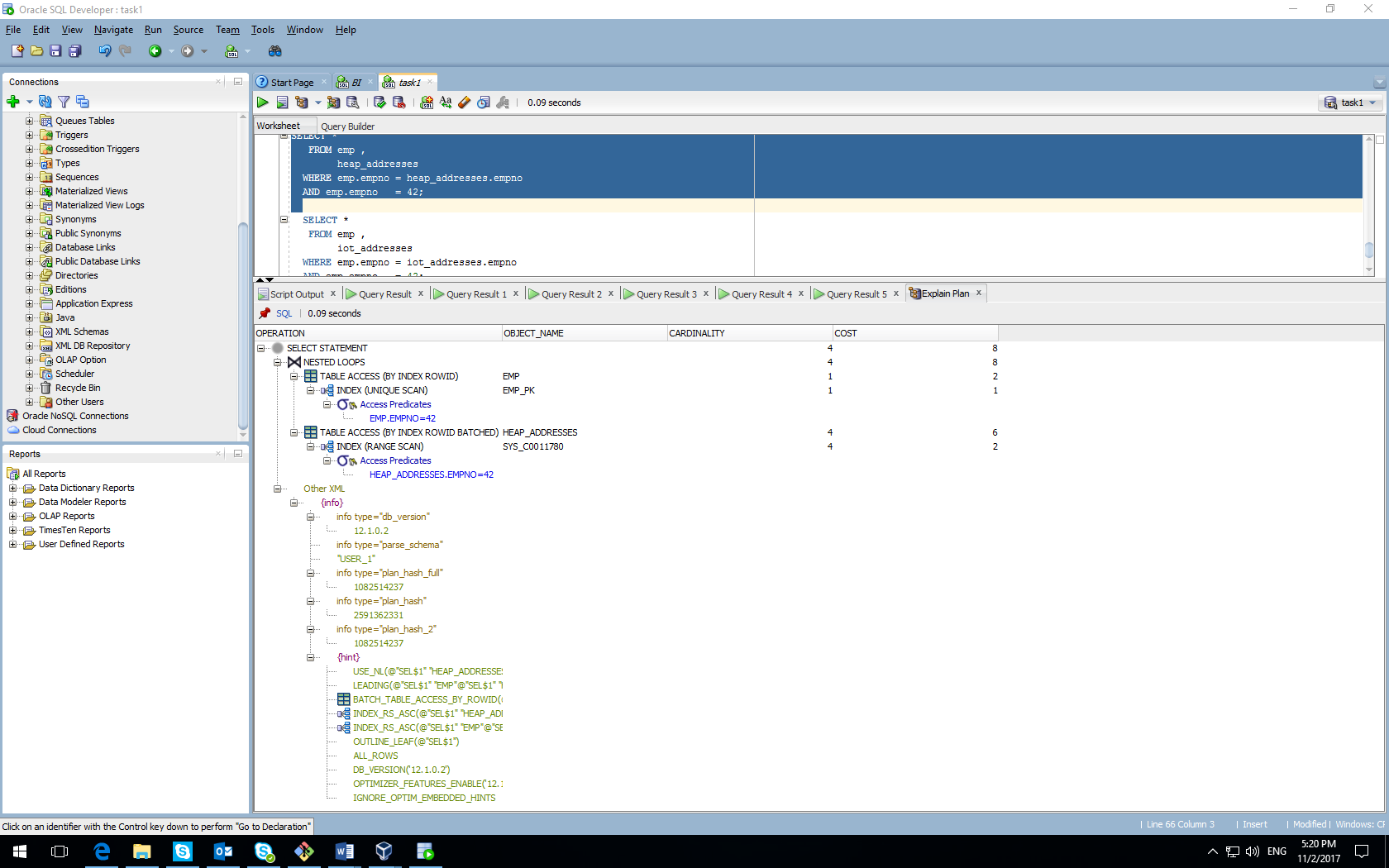
Created table t (in this case memory set before insertion )

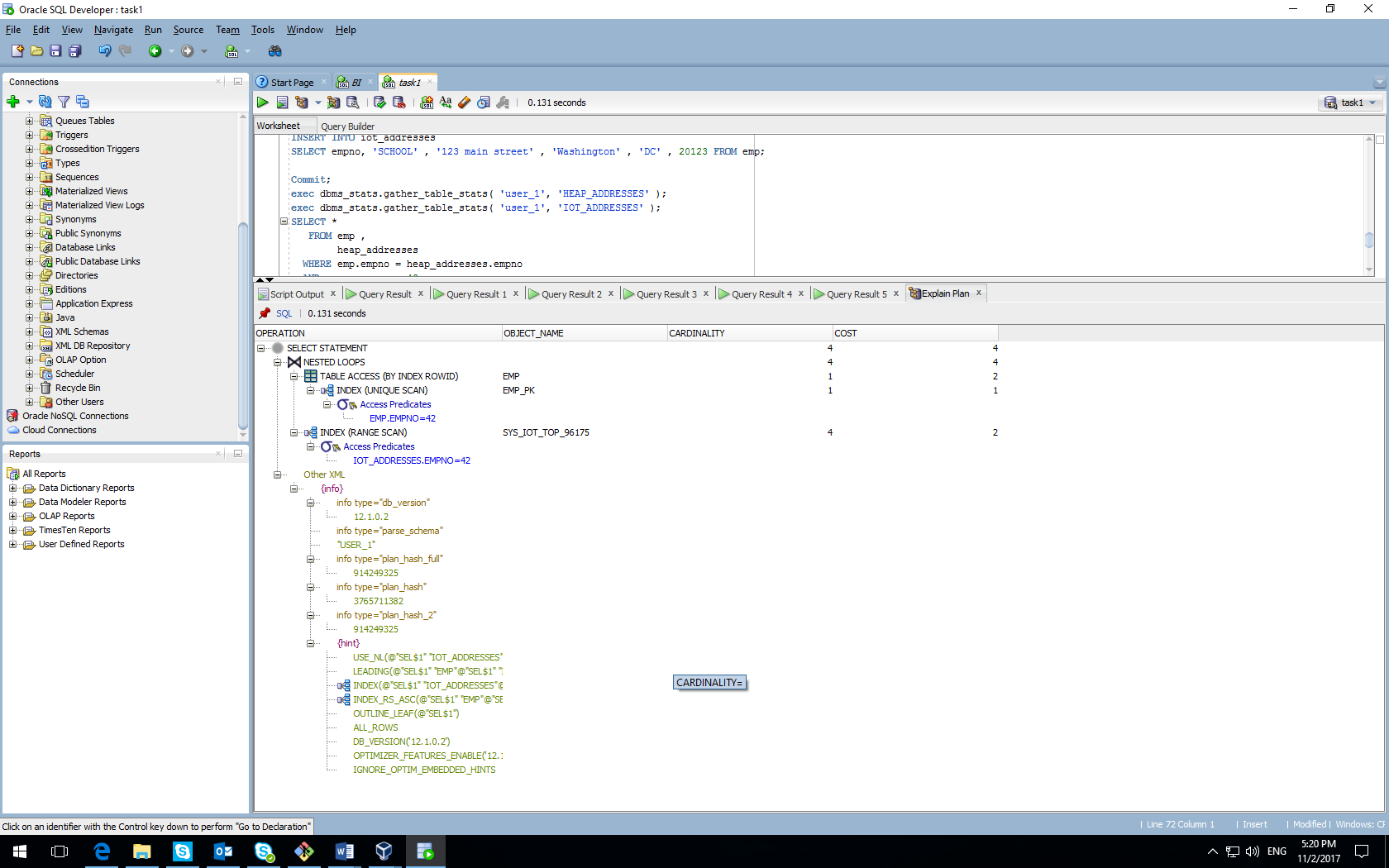
Created table t with immediate segment creation (in this case memory set immediately)





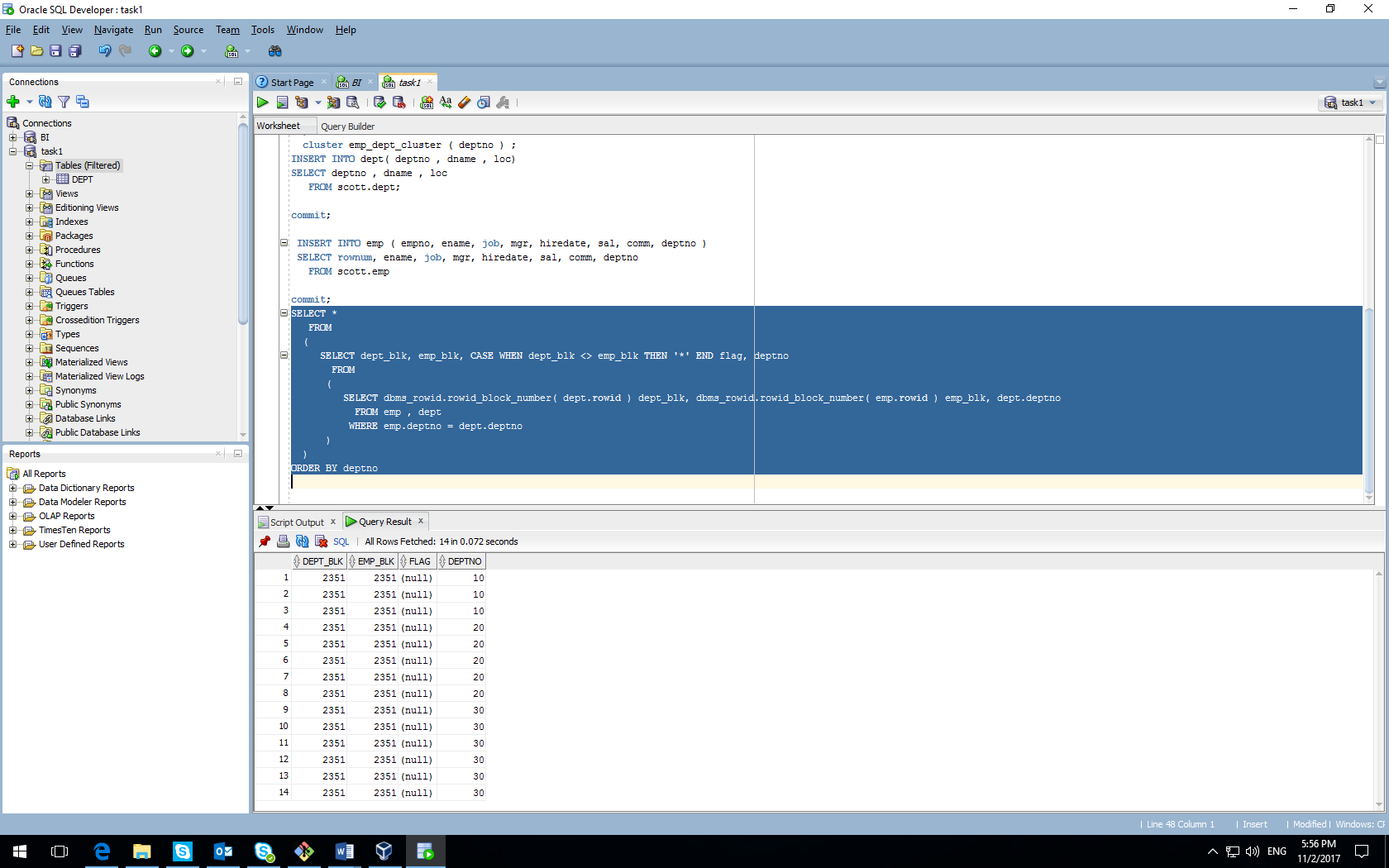
Ex3





Second query (execution) uses only index range scan while first query (execution) uses full table scan firstly and index range scan secondly

Ex4



Because of creation cluster our dept\_blk and emp\_blk are stored in the same block. We can see that they have the same clustered key.

Ex5

CREATE cluster emp\_dept\_cluster( deptno NUMBER( 2 ) )

SIZE 1024 hashkeys 15

STORAGE( INITIAL 100K NEXT 50K );

CREATE TABLE dept

(

deptno NUMBER( 2 ) PRIMARY KEY

, dname VARCHAR2( 14 )

, loc VARCHAR2( 13 )

)

cluster emp\_dept\_cluster ( deptno ) ;

CREATE TABLE emp

(

empno NUMBER PRIMARY KEY

, ename VARCHAR2( 10 )

, job VARCHAR2( 9 )

, mgr NUMBER

, hiredate DATE

, sal NUMBER

, comm NUMBER

, deptno NUMBER( 2 ) REFERENCES dept( deptno )

)

cluster emp\_dept\_cluster ( deptno ) ;

INSERT INTO dept( deptno , dname , loc)

SELECT deptno , dname , loc

FROM scott.dept;

commit;

INSERT INTO emp ( empno, ename, job, mgr, hiredate, sal, comm, deptno )

SELECT rownum, ename, job, mgr, hiredate, sal, comm, deptno

FROM scott.emp

commit;

SELECT \*

FROM

(

SELECT dept\_blk, emp\_blk, CASE WHEN dept\_blk <> emp\_blk THEN '\*' END flag, deptno

FROM

(

SELECT dbms\_rowid.rowid\_block\_number( dept.rowid ) dept\_blk, dbms\_rowid.rowid\_block\_number( emp.rowid ) emp\_blk, dept.deptno

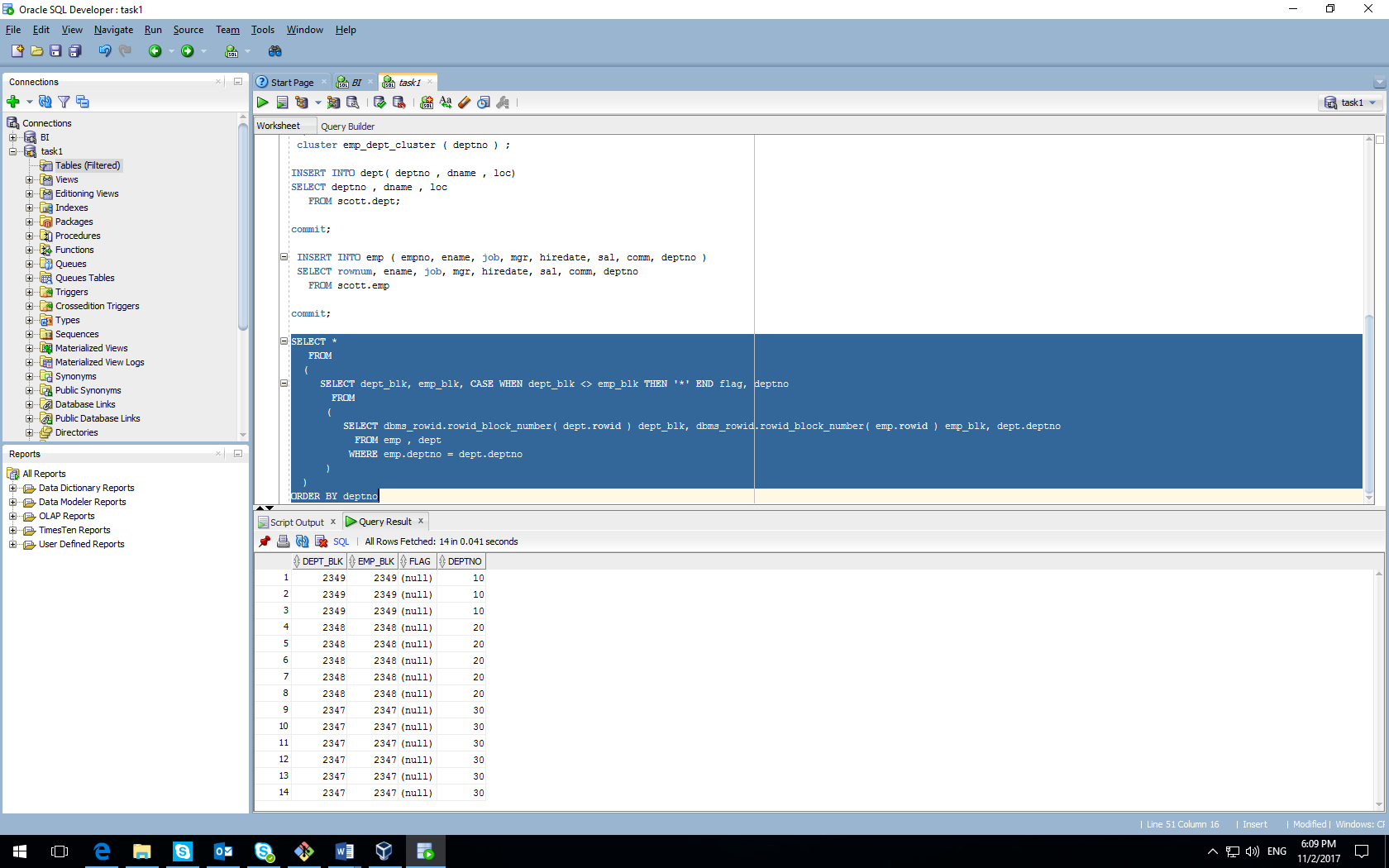
FROM emp , dept

WHERE emp.deptno = dept.deptno

)

)

ORDER BY deptno



With the hash clustered key data with the same cluster key is stored in the one block, because they have the same value of the hashed clustered key while in the first case data based on B-Tree index, and data with different clustered keys may be stored in one block.

Ex6

Chained rows - a chained rows is a LOB row (usually a BLOB, CLOB, RAW or LONG RAW) where the row length is large than the data block size.  Many shops will create a 32k blocksize to store large columns without row chaining.

Migrated rows - Migrated rows occur when an UPDATE DML causes the rows to expand onto another data block.  This can be avoided by setting PCTFREE to a large enough value to accommodate row expansion, and existing migrated rows can be fixed by reorganizing the tables with the dbms\_redefinition utility.

CREATE TABLE row\_mig\_chain\_demo (

x int PRIMARY KEY,

a CHAR(1000),

b CHAR(1000),

c CHAR(1000),

d CHAR(1000),

e CHAR(1000)

);

INSERT INTO row\_mig\_chain\_demo (x) VALUES (1);

INSERT INTO row\_mig\_chain\_demo (x) VALUES (2);

INSERT INTO row\_mig\_chain\_demo (x) VALUES (3);

COMMIT;

UPDATE row\_mig\_chain\_demo SET a = 'z1', b = 'z2', c = 'z3' WHERE x = 3;

COMMIT;

UPDATE row\_mig\_chain\_demo SET a = 'y1', b = 'y2', c = 'y3' WHERE x = 2;

COMMIT;

UPDATE row\_mig\_chain\_demo SET a = 'w1', b = 'w2', c = 'w3' WHERE x = 1;

COMMIT;

SELECT \* FROM row\_mig\_chain\_demo;

SELECT a.name, b.value

FROM v$statname a, v$mystat b

WHERE a.statistic# = b.statistic#

AND lower(a.name) = 'table fetch continued row';

SELECT \* FROM row\_mig\_chain\_demo WHERE x = 3;

SELECT a.name, b.value

FROM v$statname a, v$mystat b

WHERE a.statistic# = b.statistic#

AND lower(a.name) = 'table fetch continued row';

SELECT \* FROM row\_mig\_chain\_demo WHERE x = 1;

SELECT a.name, b.value

FROM v$statname a, v$mystat b

WHERE a.statistic# = b.statistic#

AND lower(a.name) = 'table fetch continued row';

