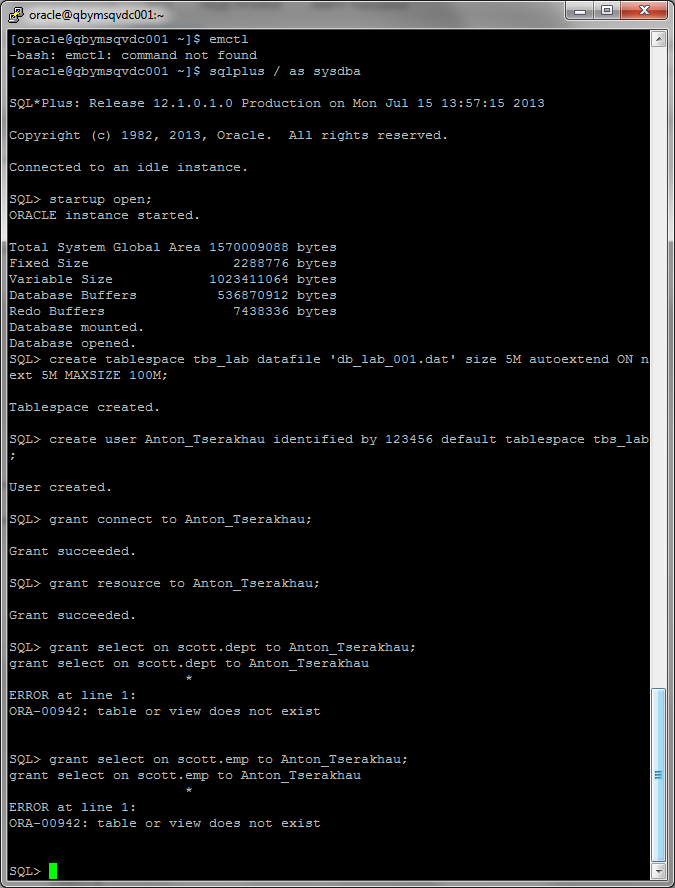
**Report “Lab 3”**

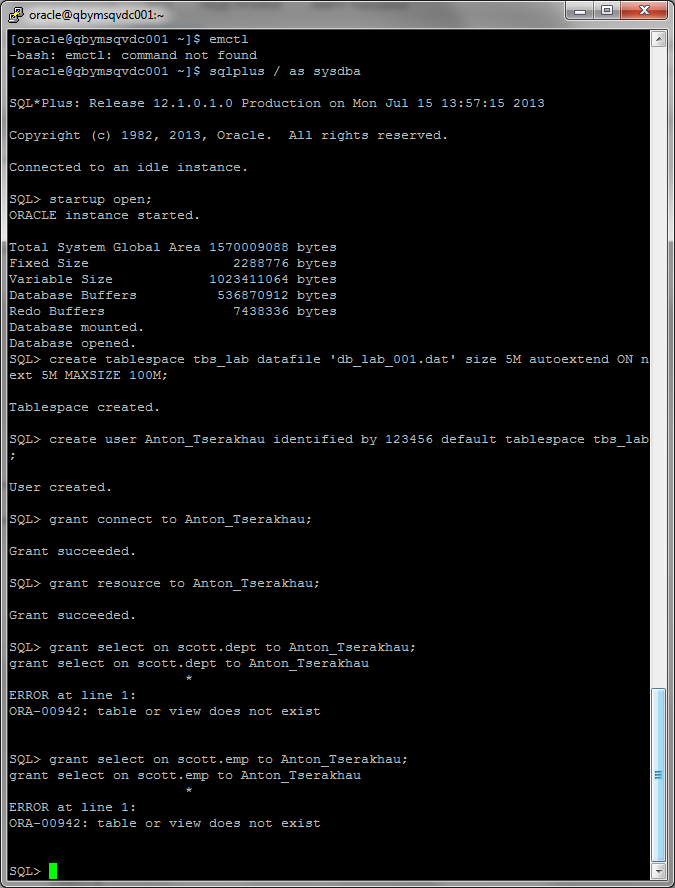
**Anton Tserakhau**

# Prerequisites

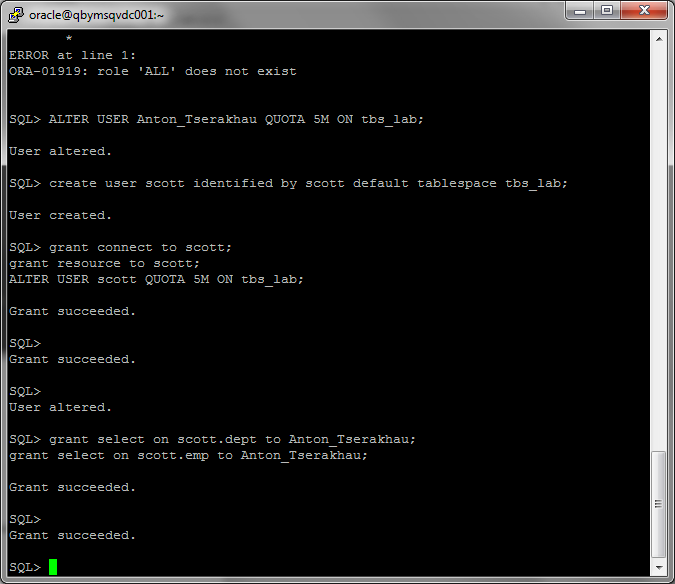
I have connected as system user and created new tablespace *tbs\_lab* with new datafile *db\_lab\_001.dat*,:



I have created new user and granted Connect Role and Resource Role:



I have created user SCOTT and granted Connect Role and Resource Role:



I have run this script to create the necessary tables:

SQL> SQL> PROMPT Building demonstration tables. Please wait.

Building demonstration tables. Please wait.

SQL> drop sequence emp\_table\_seq;

drop sequence dept\_table\_seq;

create sequence emp\_table\_seq start with 1000;

create sequence dept\_table\_Seq start with 41;

DROP TABLE EMP;

DROP TABLE DEPT;

CREATE TABLE DEPT (

DEPTNO NUMBER(2),

Sequence dropped.

SQL> DNAME VARCHAR2(14),

LOC VARCHAR2(13),

CONSTRAINT DEPT\_PK PRIMARY KEY (DEPTNO))

ROWDEPENDENCIES

;

INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');

INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');

INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');

Sequence dropped.

SQL> INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');

CREATE TABLE EMP (

EMPNO NUMBER(4) NOT NULL,

Sequence created.

SQL> ENAME VARCHAR2(10),

JOB VARCHAR2(9),

MGR NUMBER(4),

HIREDATE DATE,

Sequence created.

SQL> SQL> SAL NUMBER(7, 2),

COMM NUMBER(7, 2),

DEPTNO NUMBER(2),

CONSTRAINT EMPLOYEE\_PK PRIMARY KEY (EMPNO),

CONSTRAINT WORKS\_IN\_DEPT FOREIGN KEY (DEPTNO) REFERENCES DEPT ON DELETE SET NULL)

ROWDEPENDENCIES

;

INSERT INTO EMP VALUES

(7369, 'SMITH', 'CLERK', 7902,

TO\_DATE('17-DEC-1980', 'DD-MON-YYYY'), 800, NULL, 20);

INSERT INTO EMP VALUES

(7499, 'ALLEN', 'SALESMAN', 7698,

TO\_DATE('20-FEB-1981', 'DD-MON-YYYY'), 1600, 300, 30);

INSERT INTO EMP VALUES

(7521, 'WARD', 'SALESMAN', 7698,

TO\_DATE('22-FEB-1981', 'DD-MON-YYYY'), 1250, 500, 30);

INSERT INTO EMP VALUES

(7566, 'JONES', 'MANAGER', 7839,

TO\_DATE('2-APR-1981', 'DD-MON-YYYY'), 2975, NULL, 20);

INSERT INTO EMP VALUES

(7654, 'MARTIN', 'SALESMAN', 7698,

TO\_DATE('28-SEP-1981', 'DD-MON-YYYY'), 1250, 1400, 30);

INSERT INTO EMP VALUES

(7698, 'BLAKE', 'MANAGER', 7839,

TO\_DATE('1-MAY-1981', 'DD-MON-YYYY'), 2850, NULL, 30);

INSERT INTO EMP VALUES

(7782, 'CLARK', 'MANAGER', 7839,

TO\_DATE('9-JUN-1981', 'DD-MON-YYYY'), 2450, NULL, 10);

INSERT INTO EMP VALUES

(7788, 'SCOTT', 'ANALYST', 7566,

TO\_DATE('09-DEC-1982', 'DD-MON-YYYY'), 3000, NULL, 20);

INSERT INTO EMP VALUES

(7839, 'KING', 'PRESIDENT', NULL,

TO\_DATE('17-NOV-1981', 'DD-MON-YYYY'), 5000, NULL, 10);

INSERT INTO EMP VALUES

(7844, 'TURNER', 'SALESMAN', 7698,

TO\_DATE('8-SEP-1981', 'DD-MON-YYYY'), 1500, 0, 30);

INSERT INTO EMP VALUES

(7876, 'ADAMS', 'CLERK', 7788,

TO\_DATE('12-JAN-1983', 'DD-MON-YYYY'), 1100, NULL, 20);

INSERT INTO EMP VALUES

(7900, 'JAMES', 'CLERK', 7698,

TO\_DATE('3-DEC-1981', 'DD-MON-YYYY'), 950, NULL, 30);

INSERT INTO EMP VALUES

(7902, 'FORD', 'ANALYST', 7566,

TO\_DATE('3-DEC-1981', 'DD-MON-YYYY'), 3000, NULL, 20);

INSERT INTO EMP VALUES

(7934, 'MILLER', 'CLERK', 7782,

TO\_DATE('23-JAN-1982', 'DD-MON-YYYY'), 1300, NULL, 10);

COMMIT;

REM ALTER TABLE EMP ADD (

REM CONSTRAINT MANAGED\_BY FOREIGN KEY (MGR)

REM REFERENCES EMP);

CREATE INDEX DEPTNO\_ON\_EMP ON EMP(DEPTNO);

create or replace trigger dept\_table\_befins

before insert on dept for each row

begin

if (:new.deptno is null or :new.deptno < 0) then

select dept\_table\_seq.nextval into :new.deptno from dual;

end if;

end;

.

/

create or replace trigger emp\_table\_befins

before insert on emp for each row

begin

if (:new.empno is null or :new.empno < 0) then

select emp\_Table\_seq.nextval into :new.empno from dual;

end if;

Table dropped.

SQL> end;

.

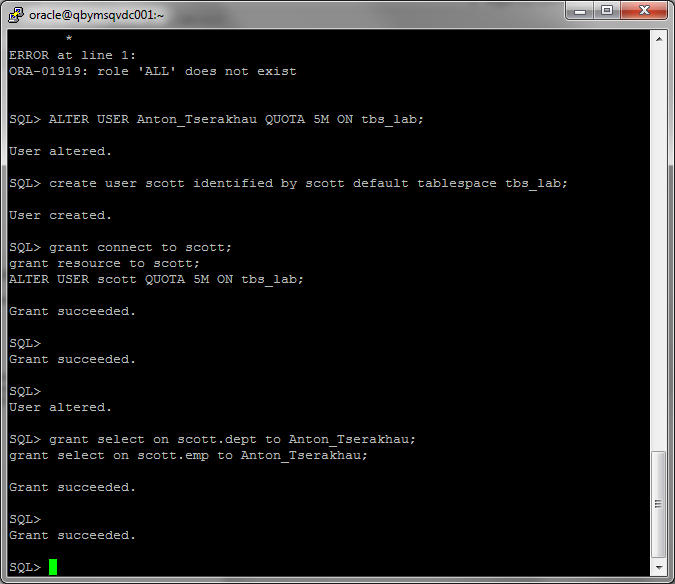
/

PROMPT Demonstration table build is complete.

PURGE RECYCLEBIN

/

After this, I have granted privileges for user to use tables from schema SCOTT:

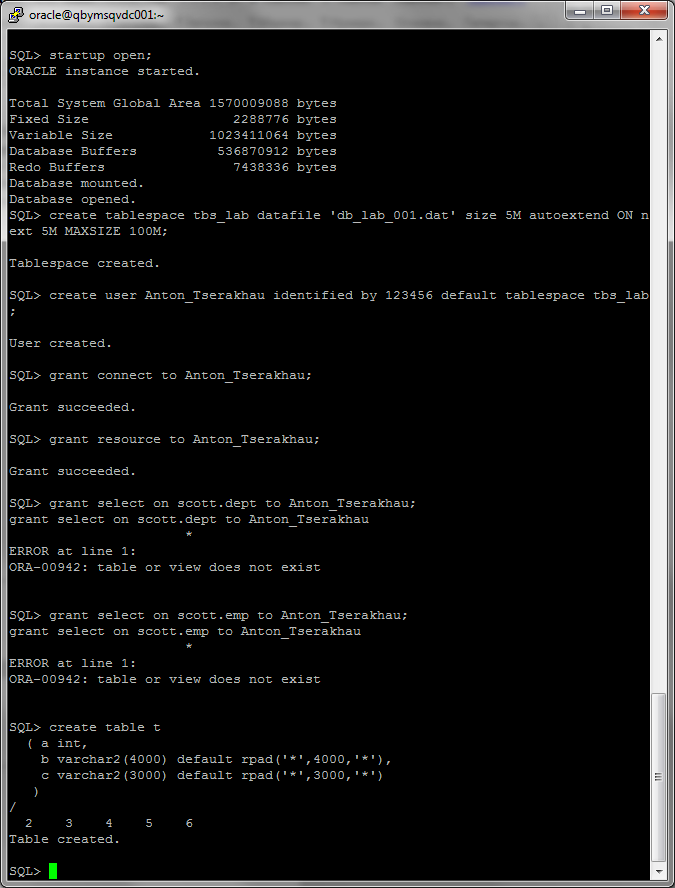


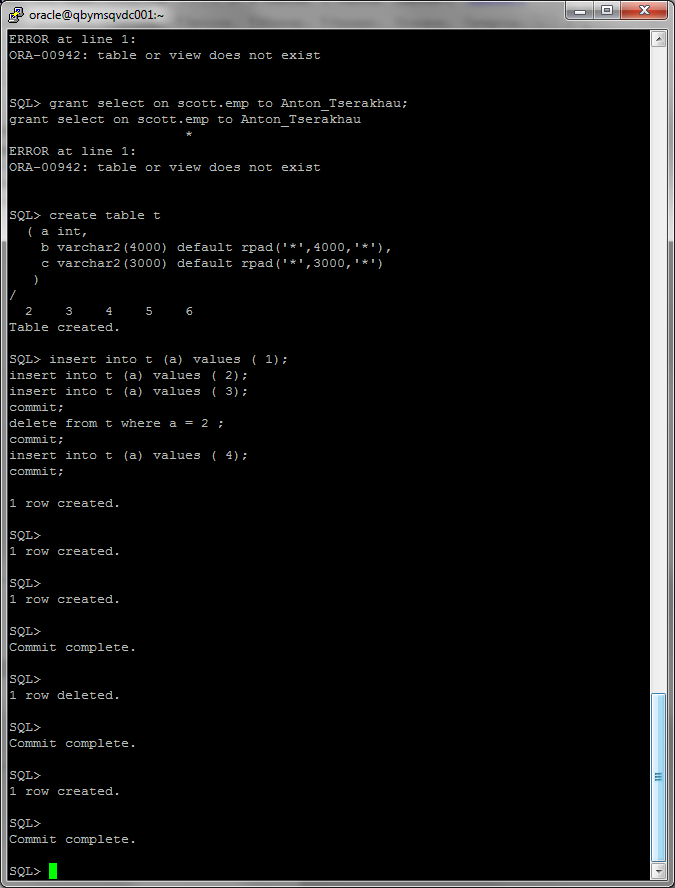
# Heap Organized Tables

## Task 1 – Heap Understanding

A heap is a classic data structure studied in computer science. It is basically a big area of space, disk, or memory (disk in the case of a database table, of course) that is managed in an apparently random fashion. Data will be placed where it fits best, rather than in any specific sort of order.

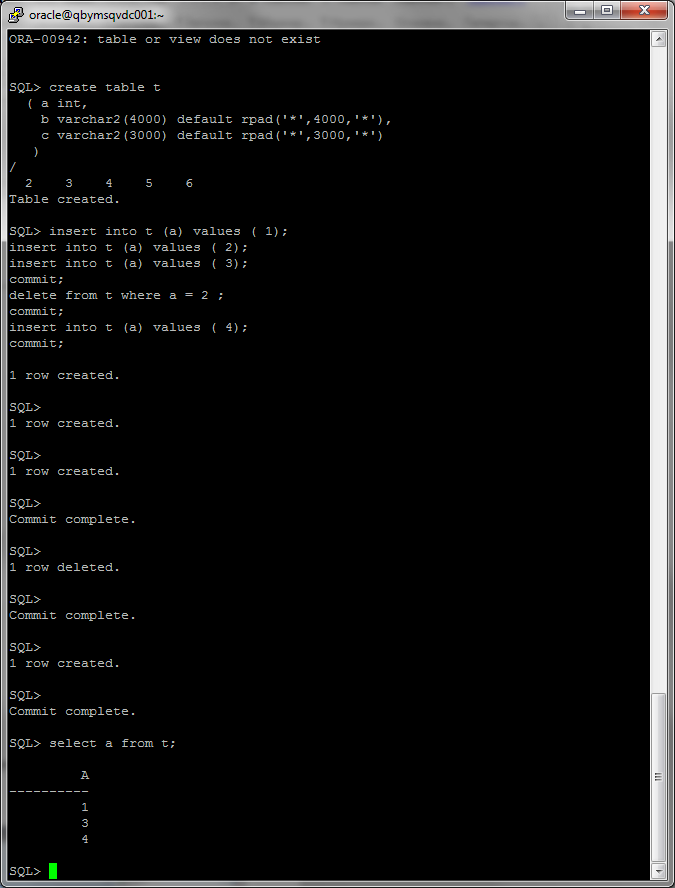
I have created table and inserted data:



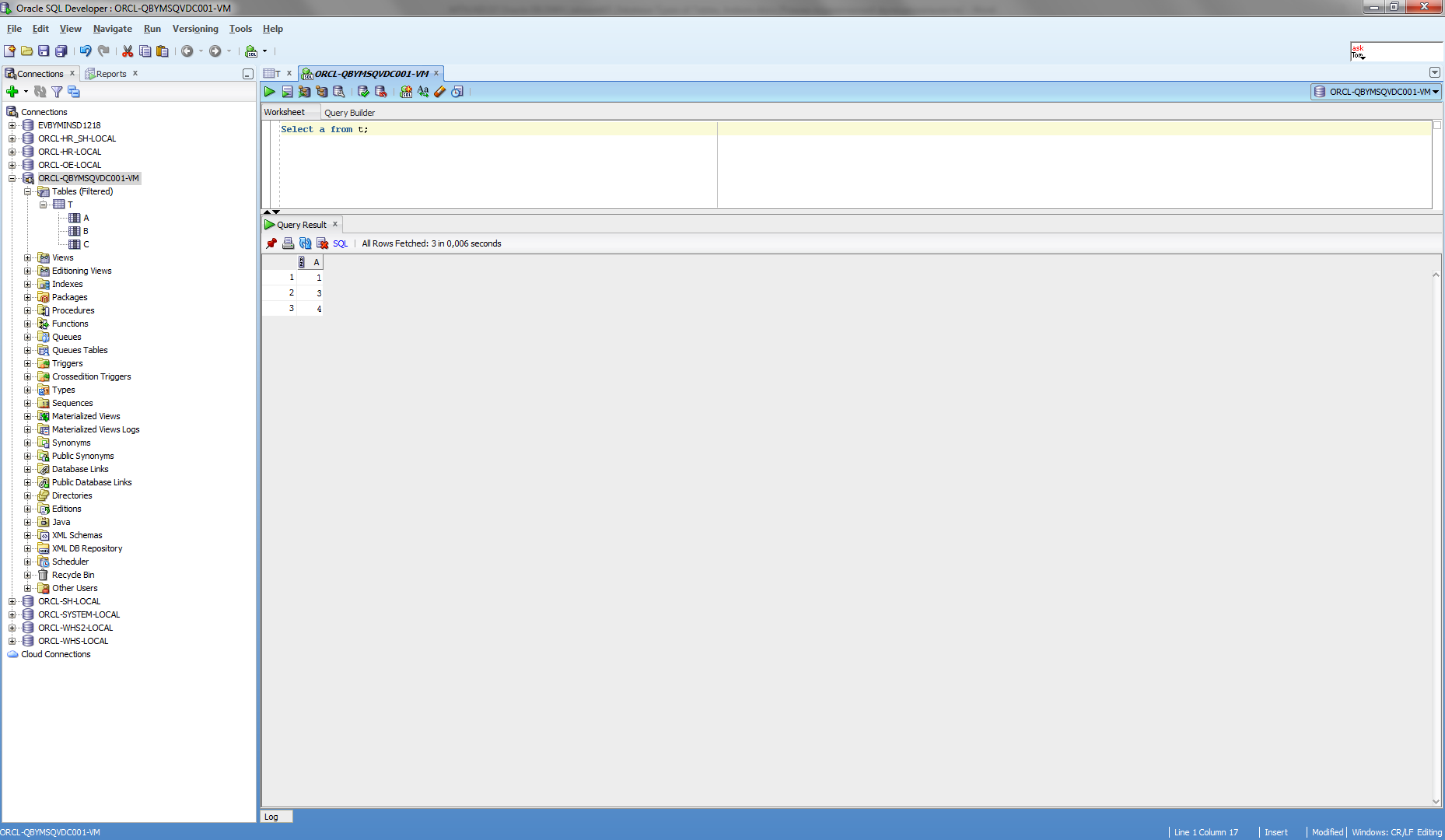


Result of SELECT query:

SQL Plus:



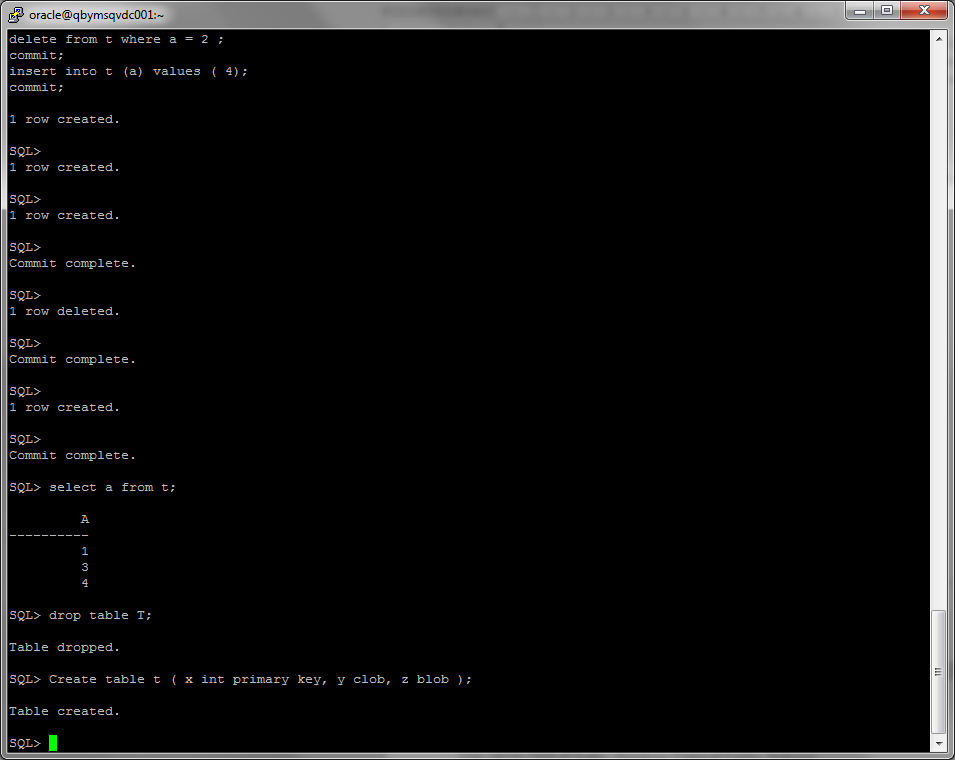
Oracle SQL Developer:

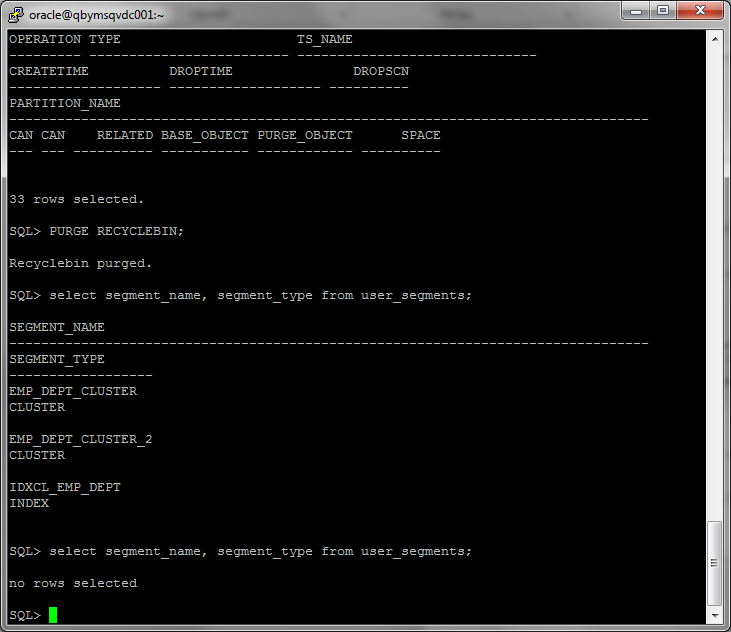


## Task 2 – Understanding Low level of data abstraction: Heap Table Segments

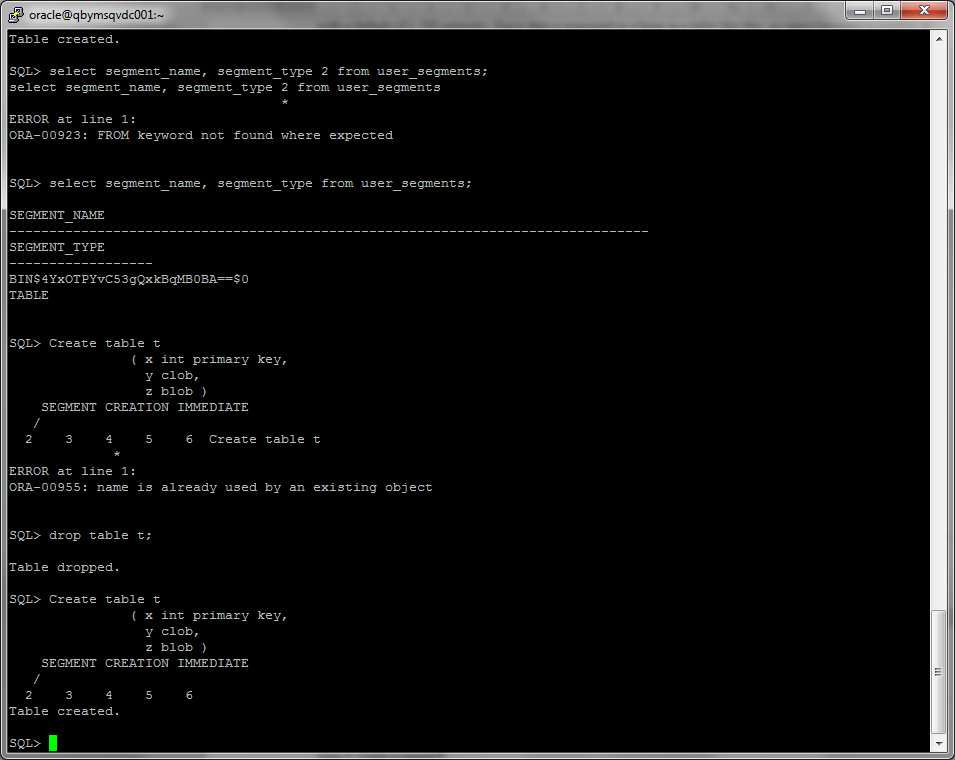
I have created tables without and with SEGMENT CREATION.

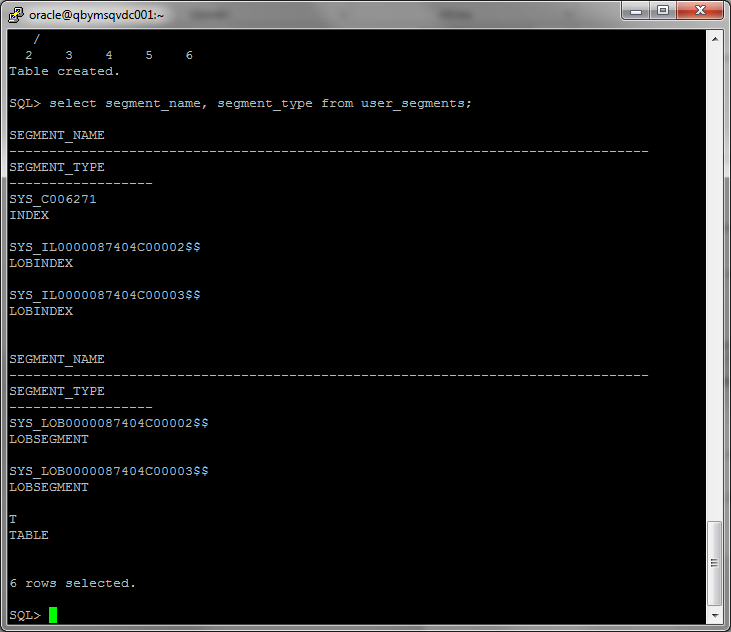
There is not SEGMENT CREATION, so there is not records about segments in the system table *user\_segments*:

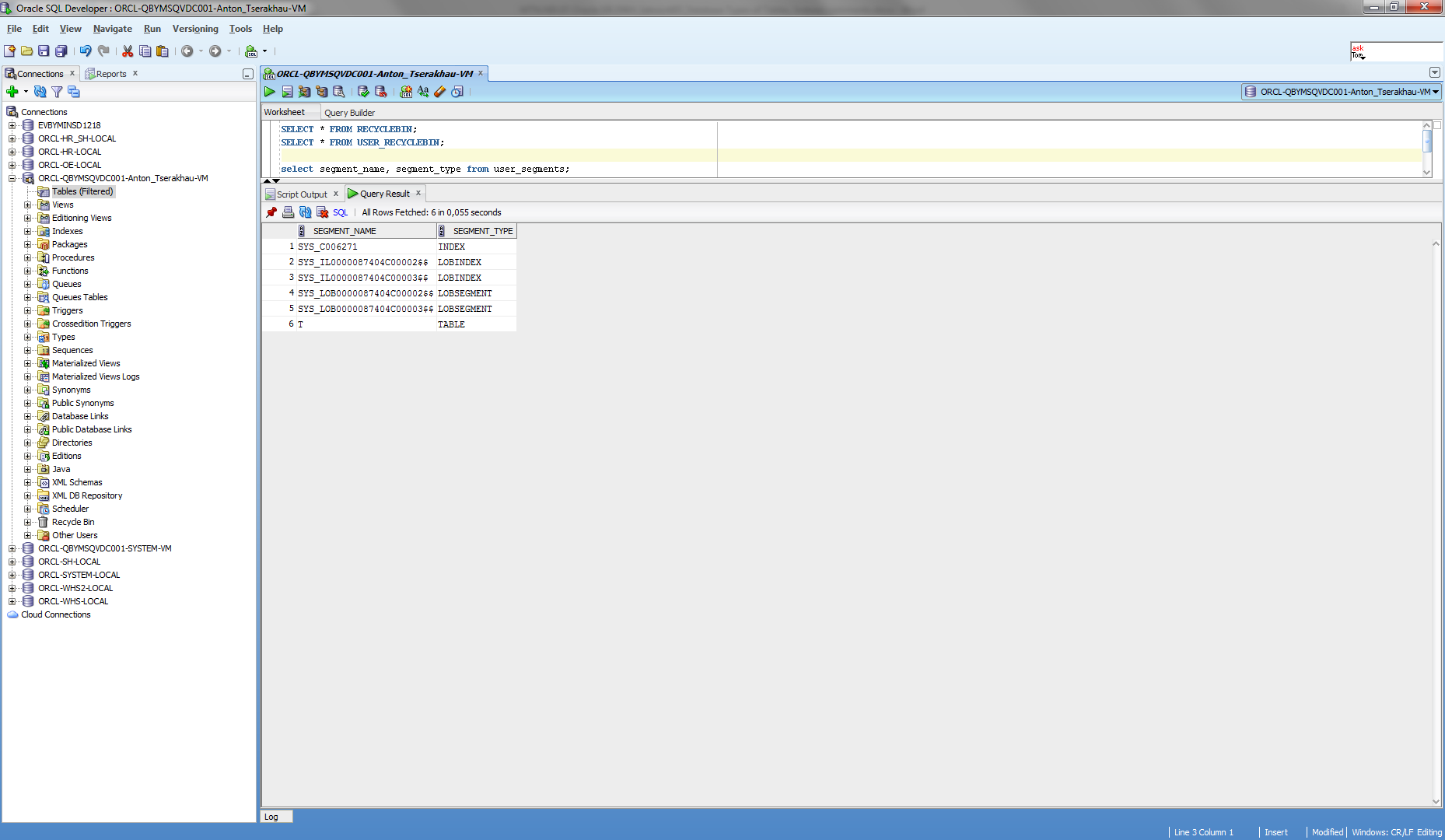




There is SEGMENT CREATION, so there are records about segments in the system table *user\_segments*:



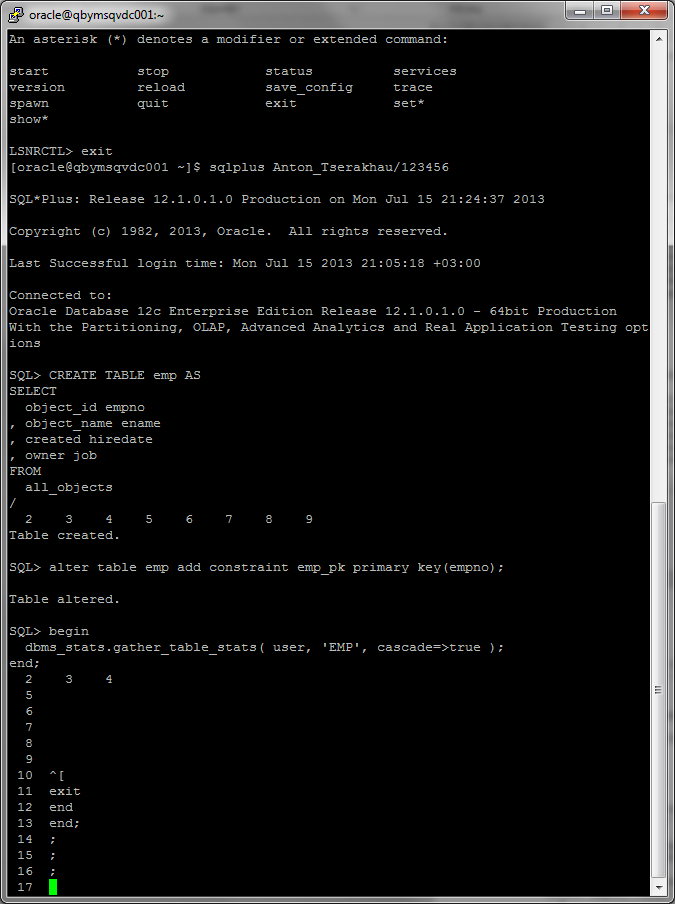


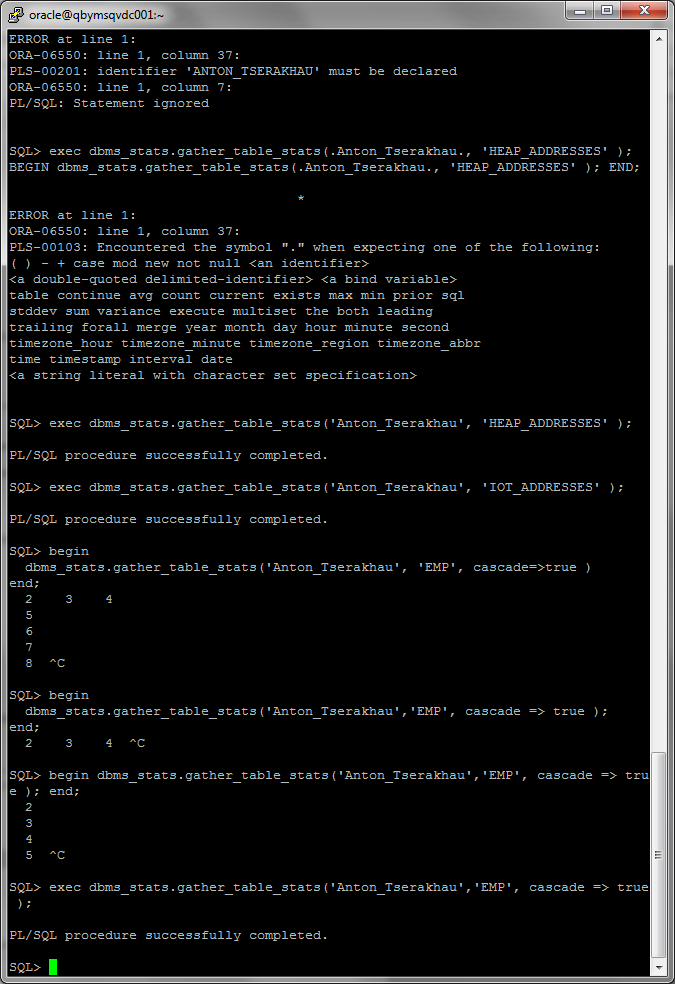


# Index Organized Tables

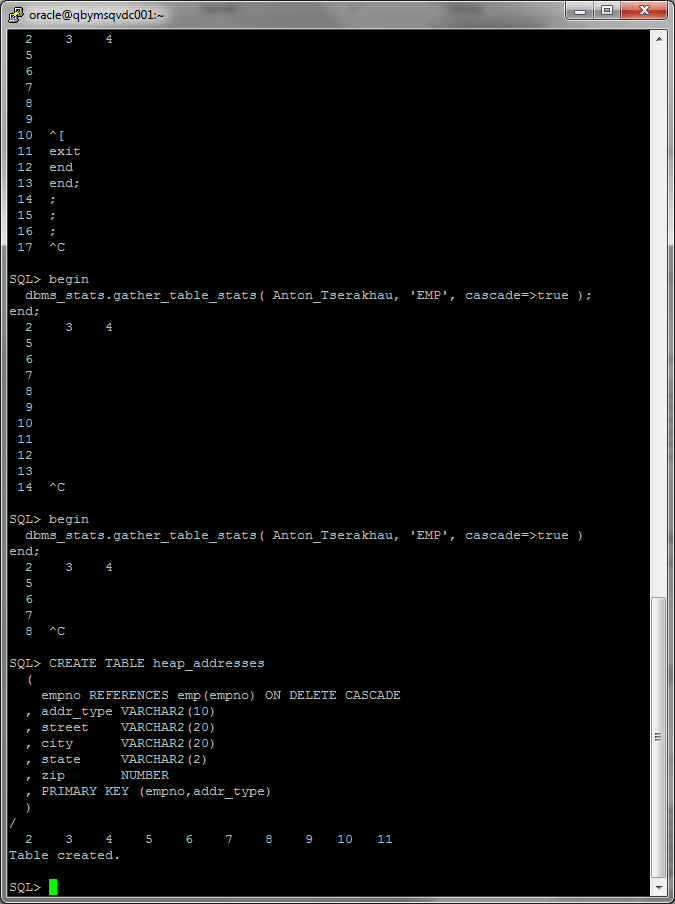
## Task 3: Compare performance of using IOT tables

I have created table and added primary key, calculated statistics:

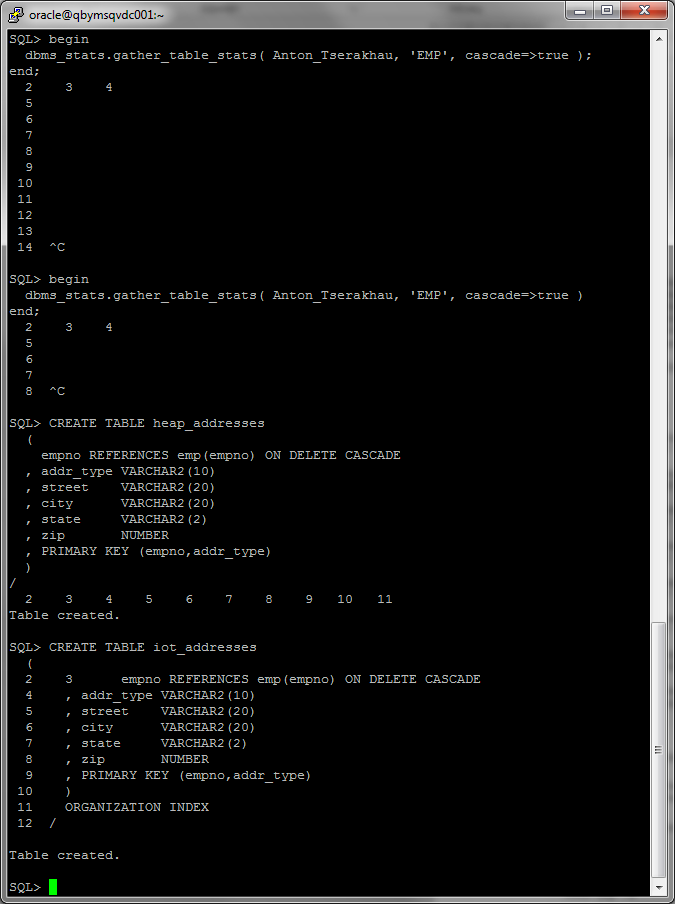




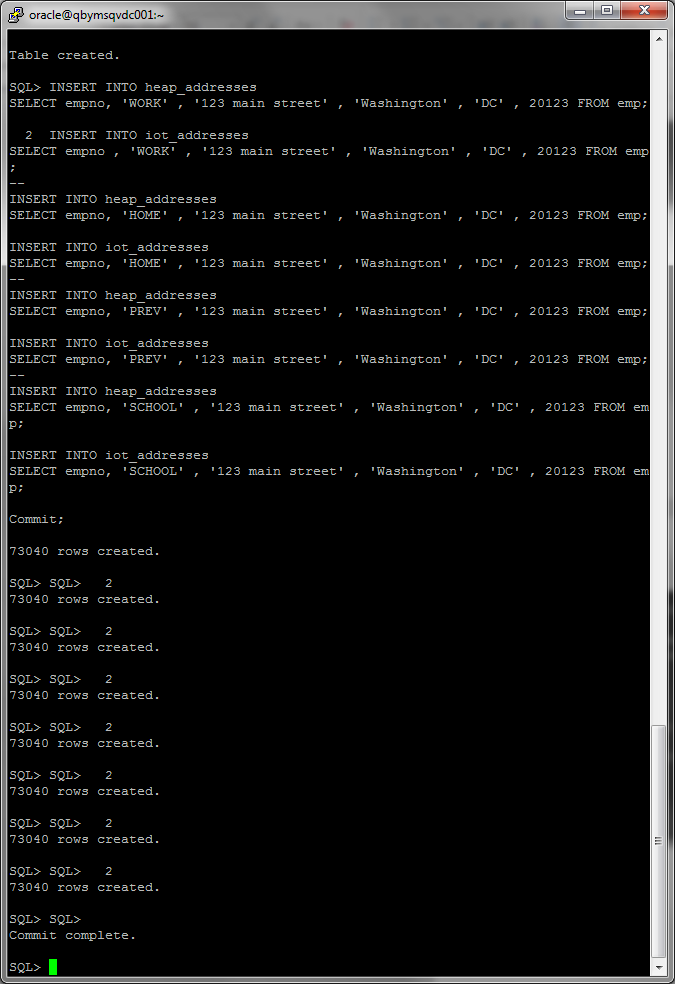
I have created heap organized table:



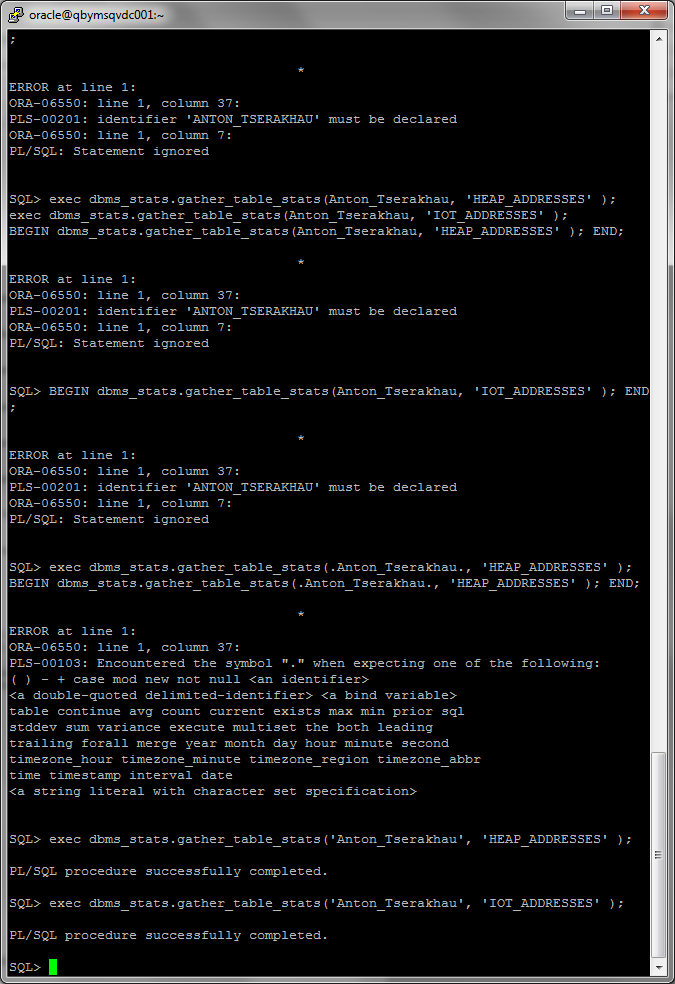
I have created index organized table:



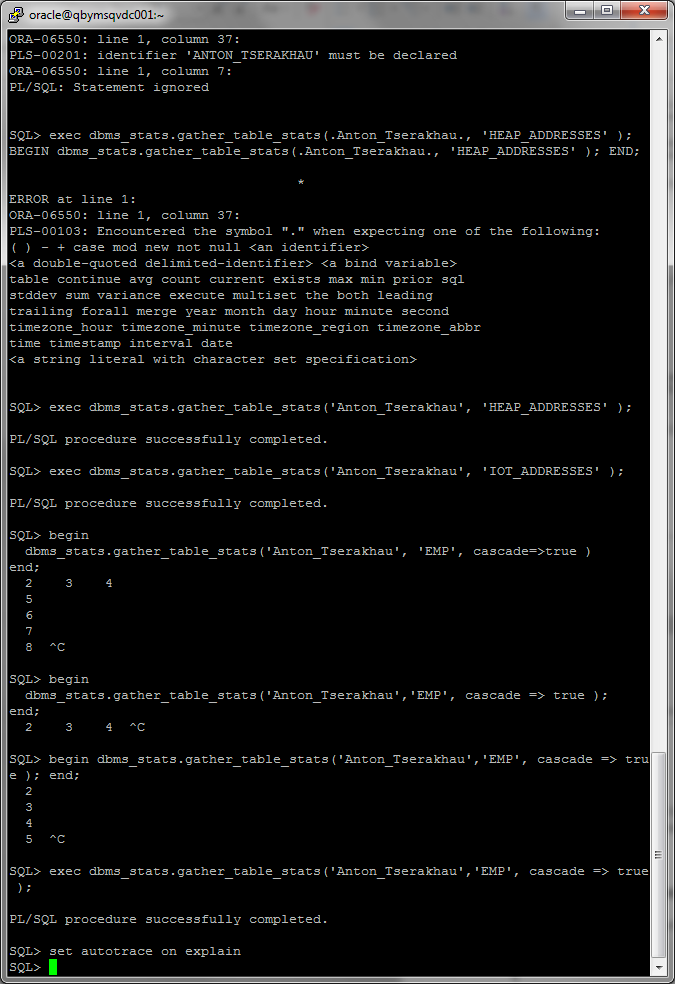
I have inserted data:



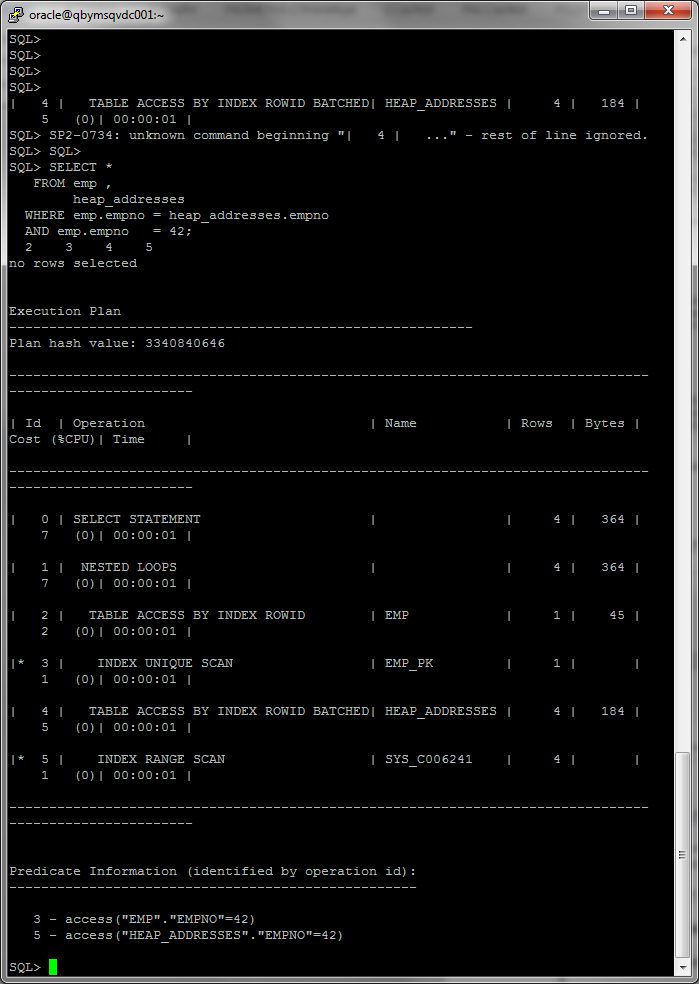
I have calculated statistics

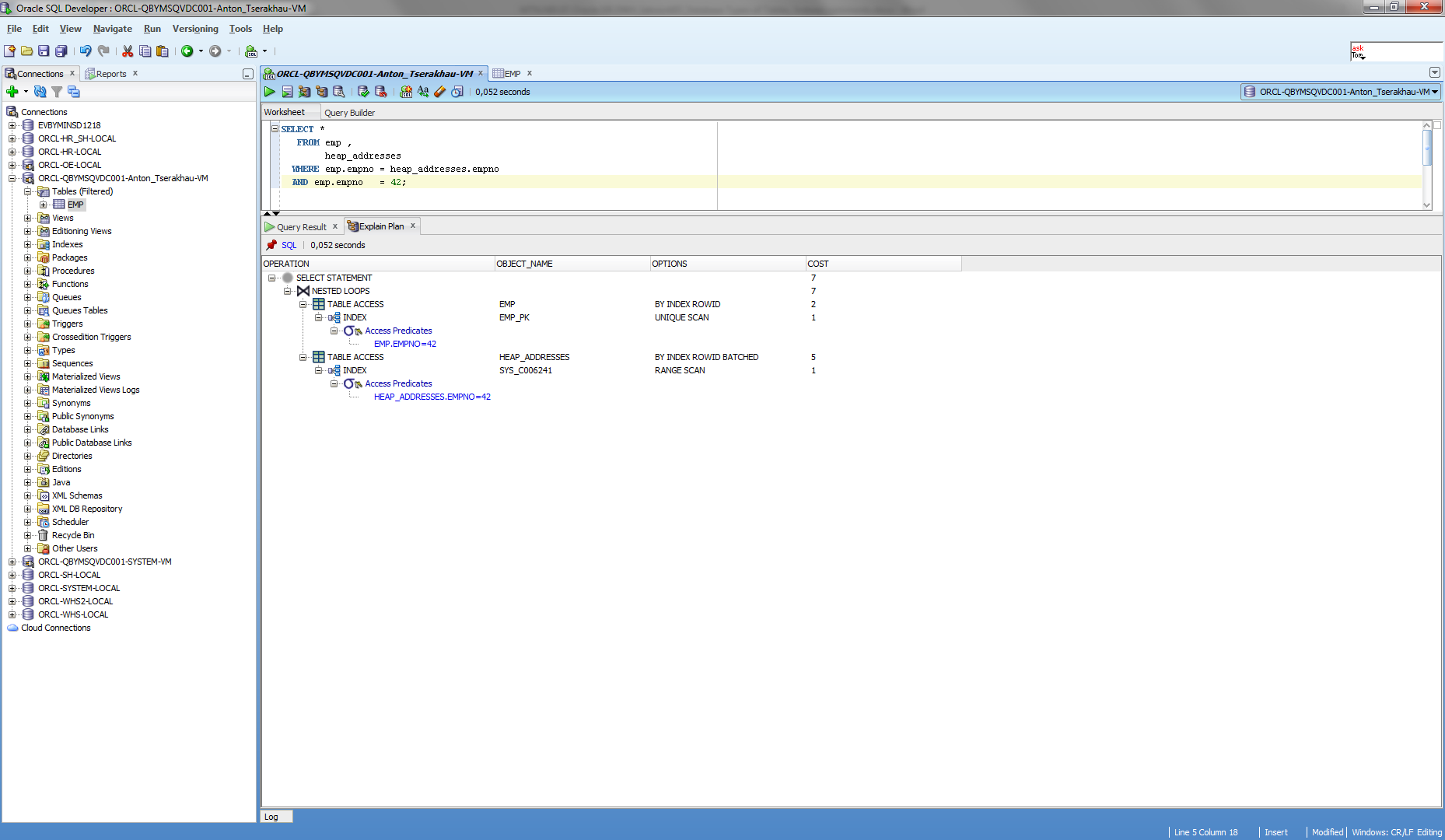


I have turned on autotrace:

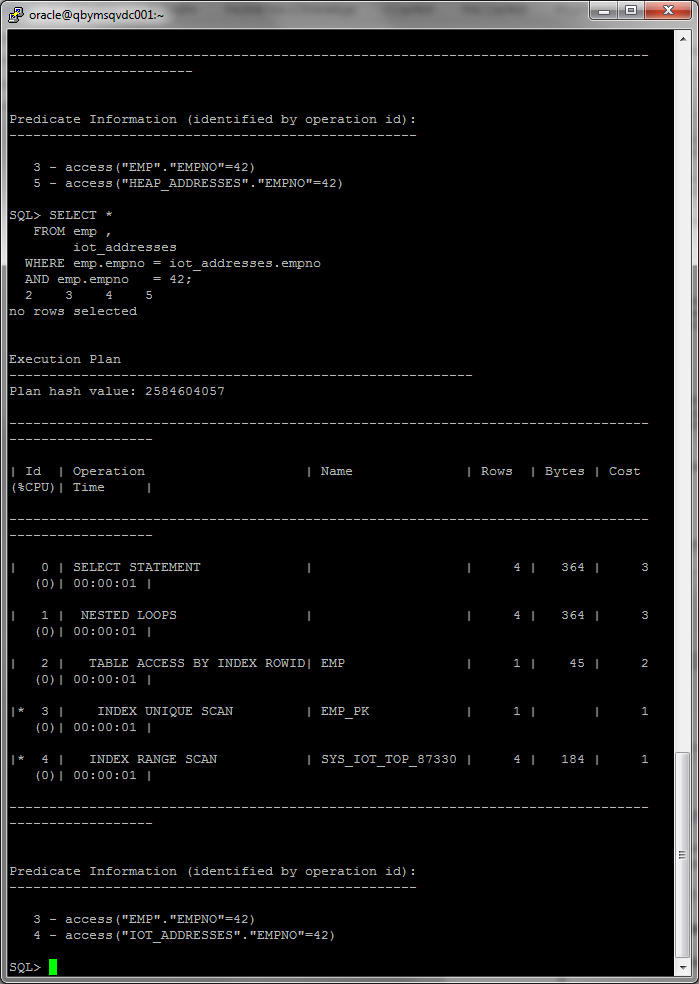


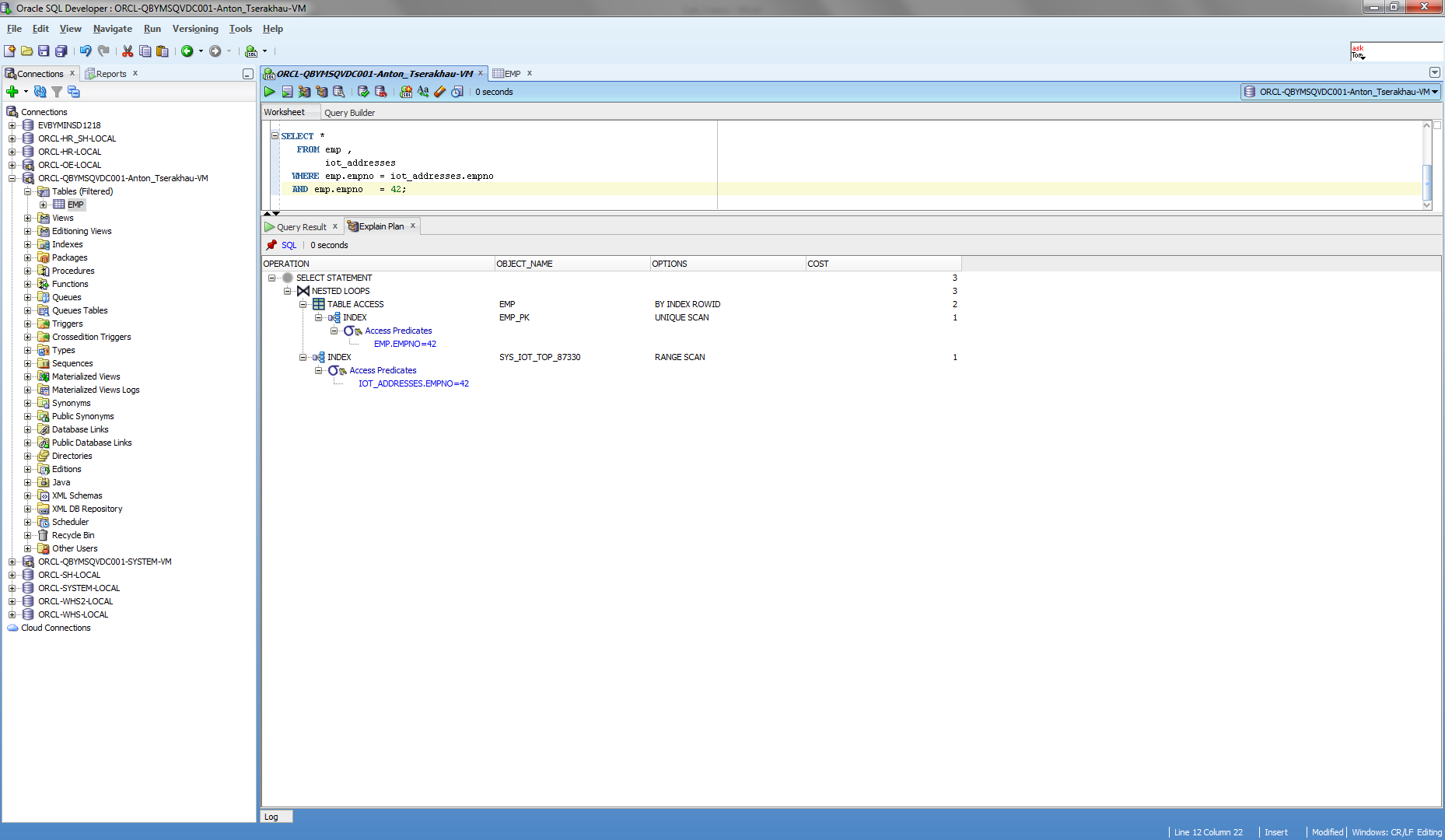
I have received result of query with heap organized table, execution plan and statistics:





I have received result of query with index organized table, execution plan and statistics:

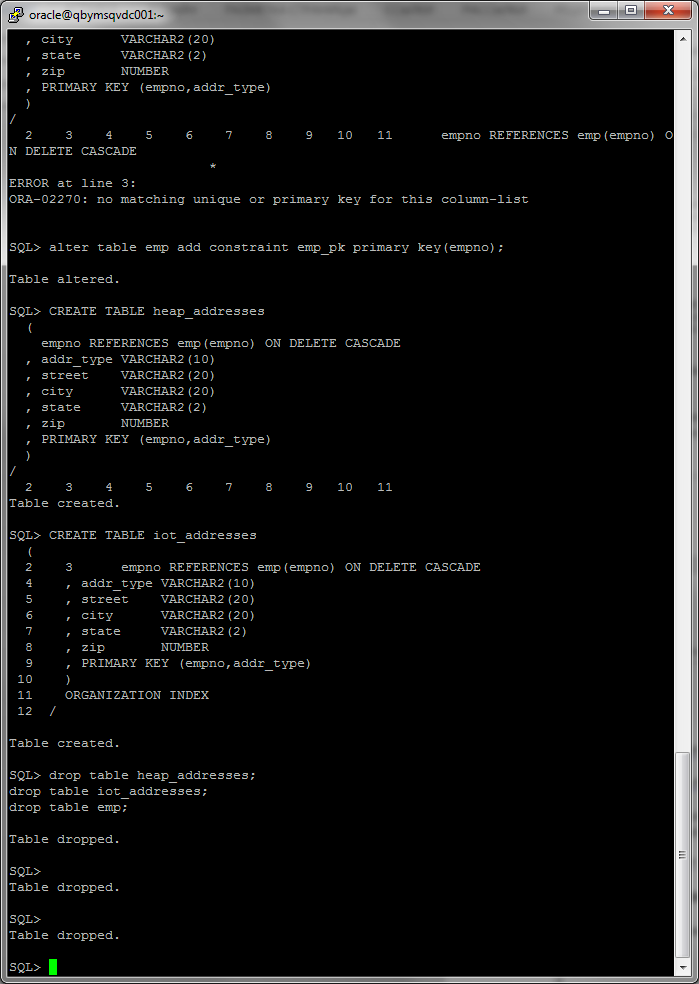




Explanation:

Whereas a table stored in a heap is unorganized (i.e., data goes wherever there is available space), data in an IOT is stored and sorted by primary key. The search is performed by primary key. Therefore it is not needed access to the table HEAP\_ADDRESSES (to receive data), which cost 5% CPU. So heap table cost > IOT table cost.

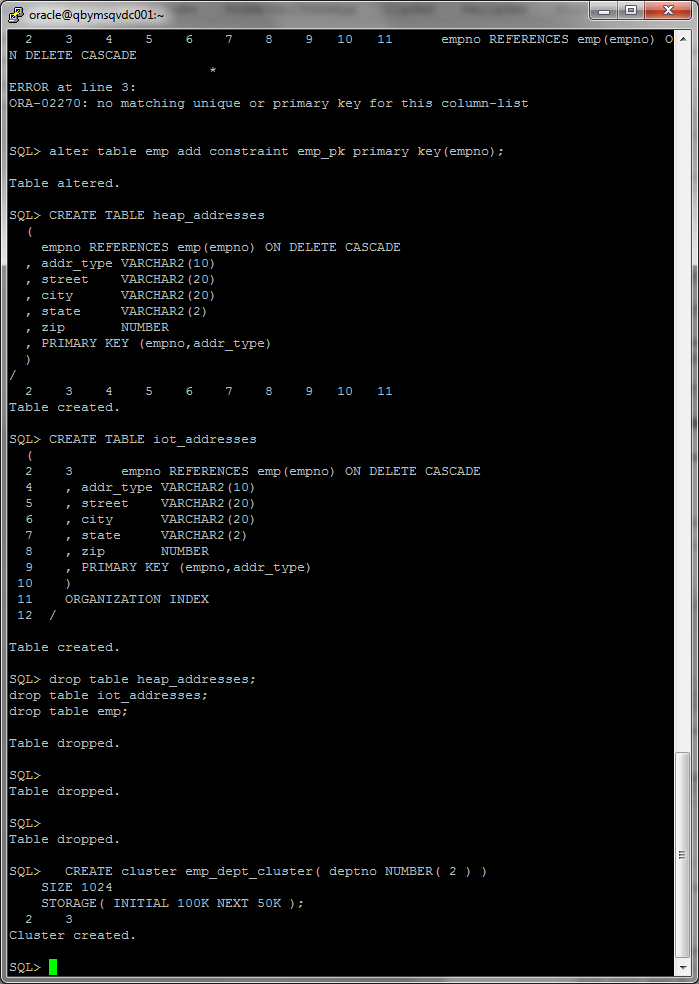
I have cleaned up scheme:

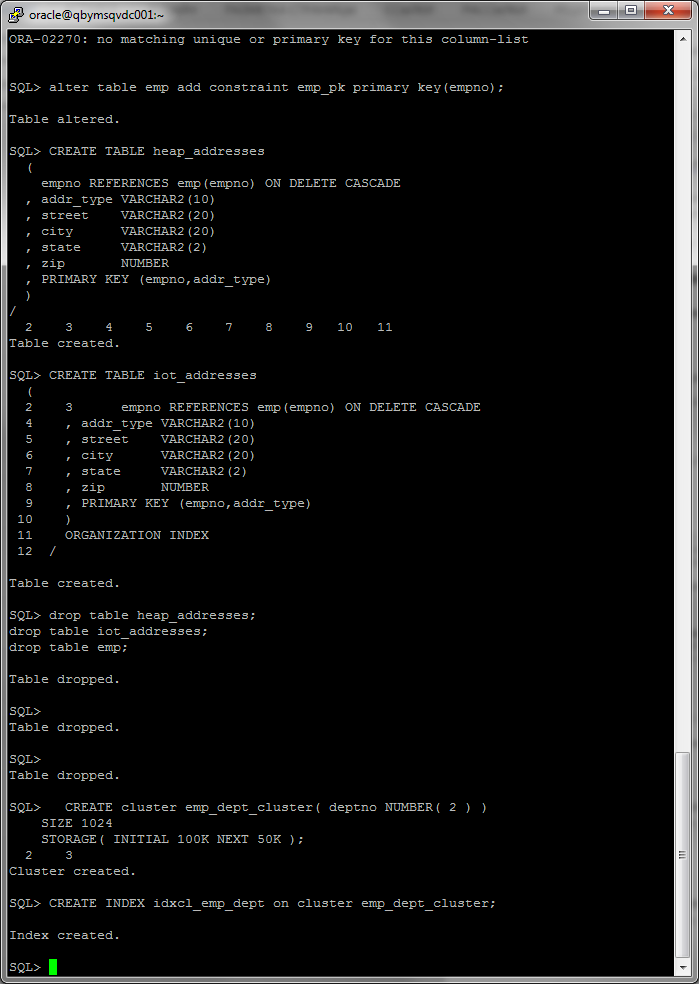


# Index Clustered Tables

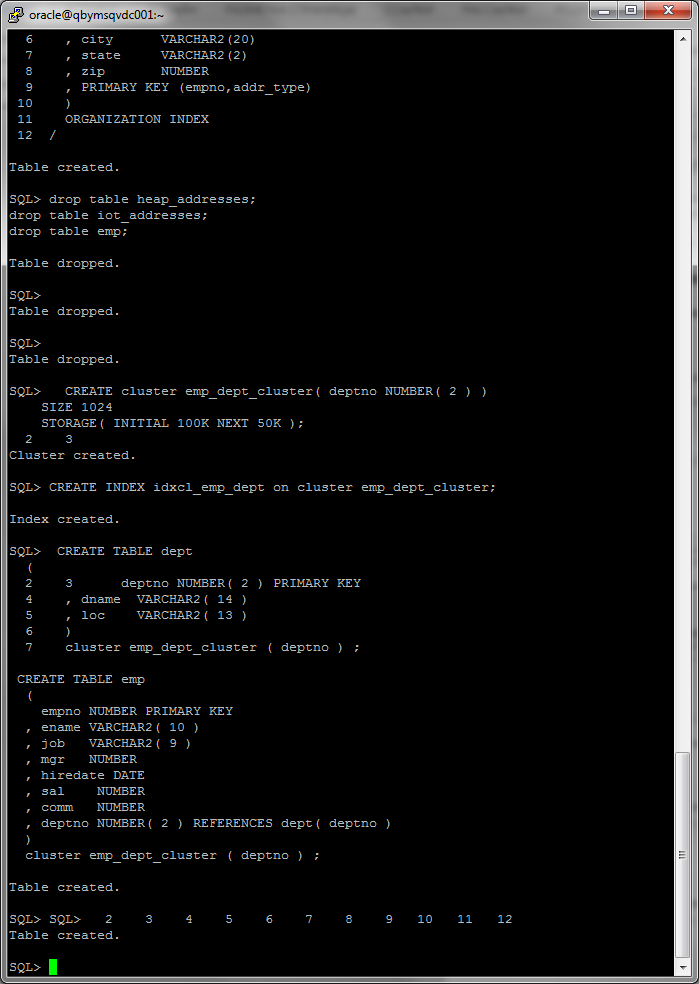
## Task 4: Analyses Cluster Storage by Blocks

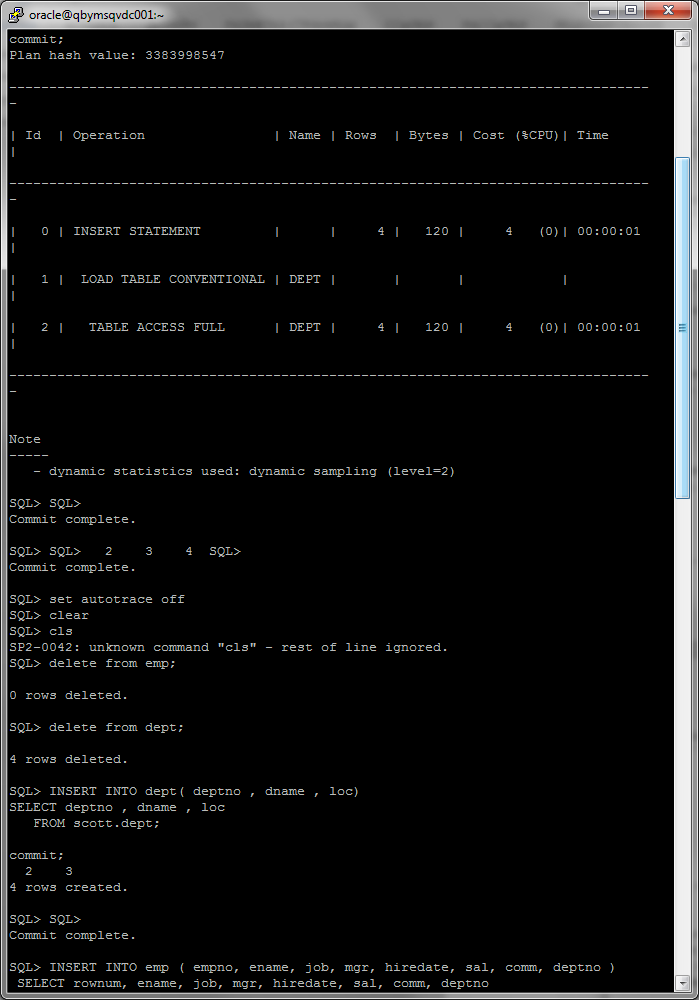
I have created cluster and index on this cluster:

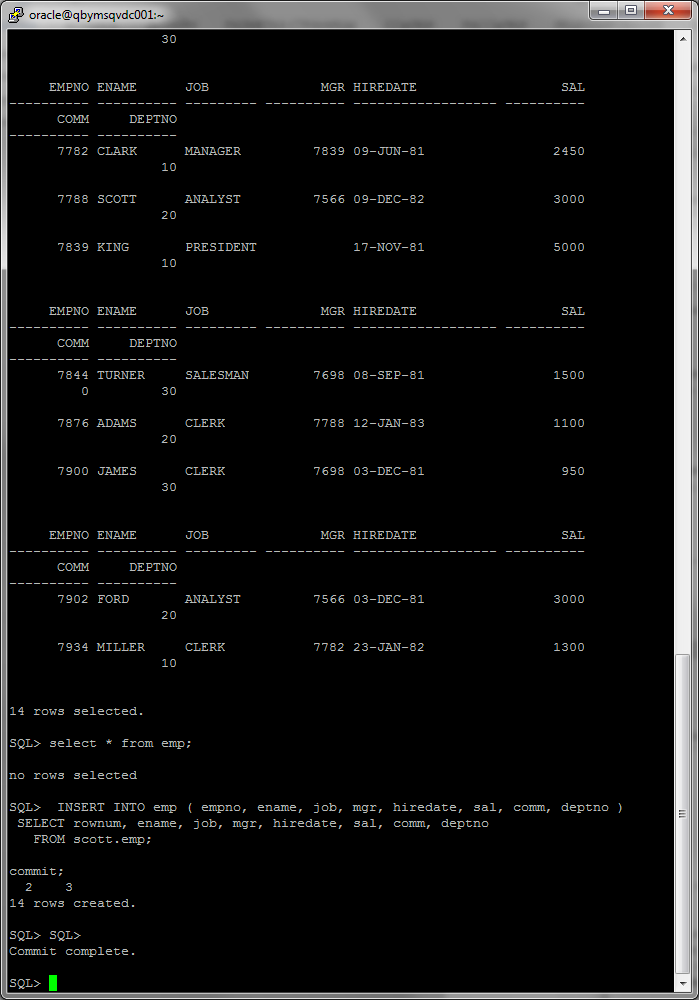




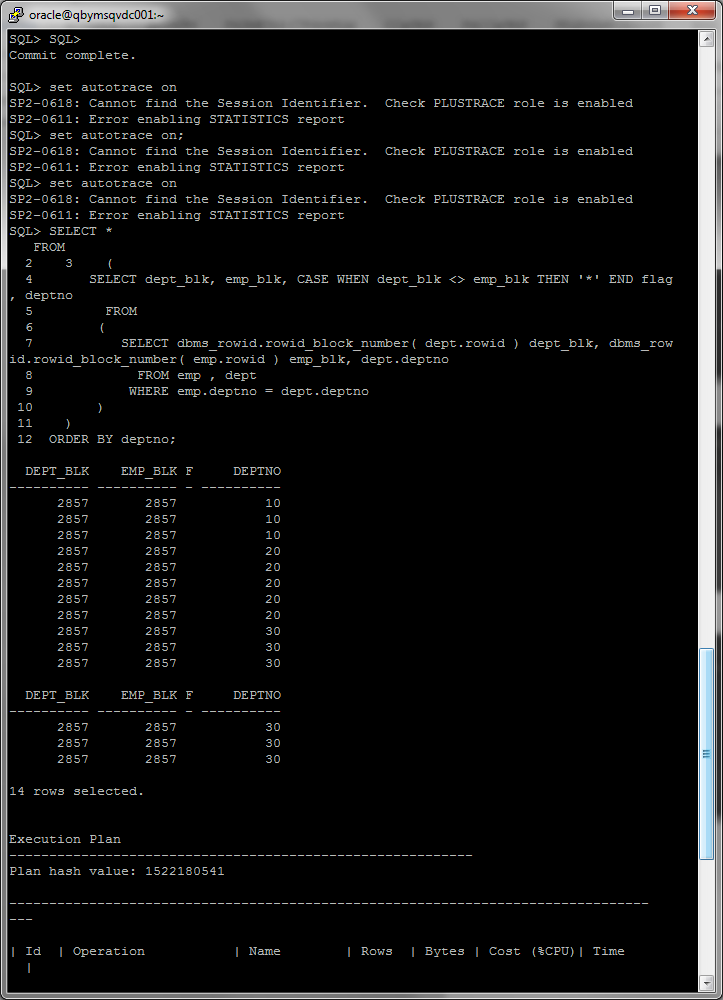
I have created index clustered tables and inserted data:







All data are stored on the same block:



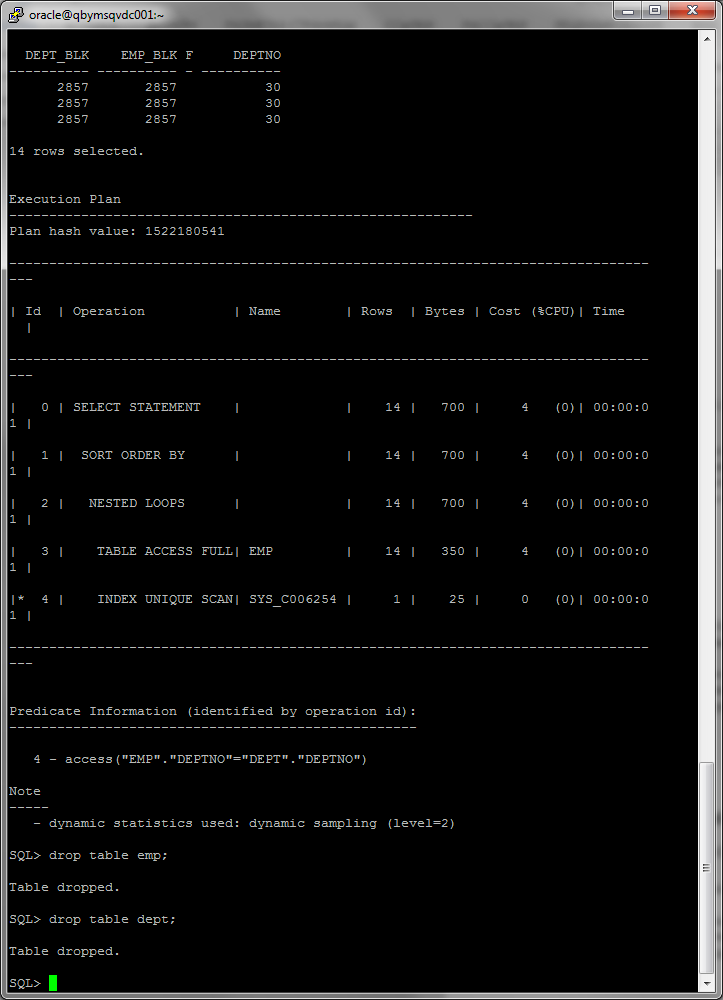
Explanation:

With a cluster, a single block of data may contain data from many tables. Conceptually, you are storing the data “prejoined”. Cluster is storing the data clustered by some key, but in a heap. So all data by one number of department are stored together – in this example, in block 2857 (we have few data in this example, so we have only one block).

Advantages of this type of storage:

Clustered tables give you the ability to physically prejoin data together. You use clusters to store related data from many tables on the same database block. Clusters can help read-intensive operations that always join data together or access related sets of data.

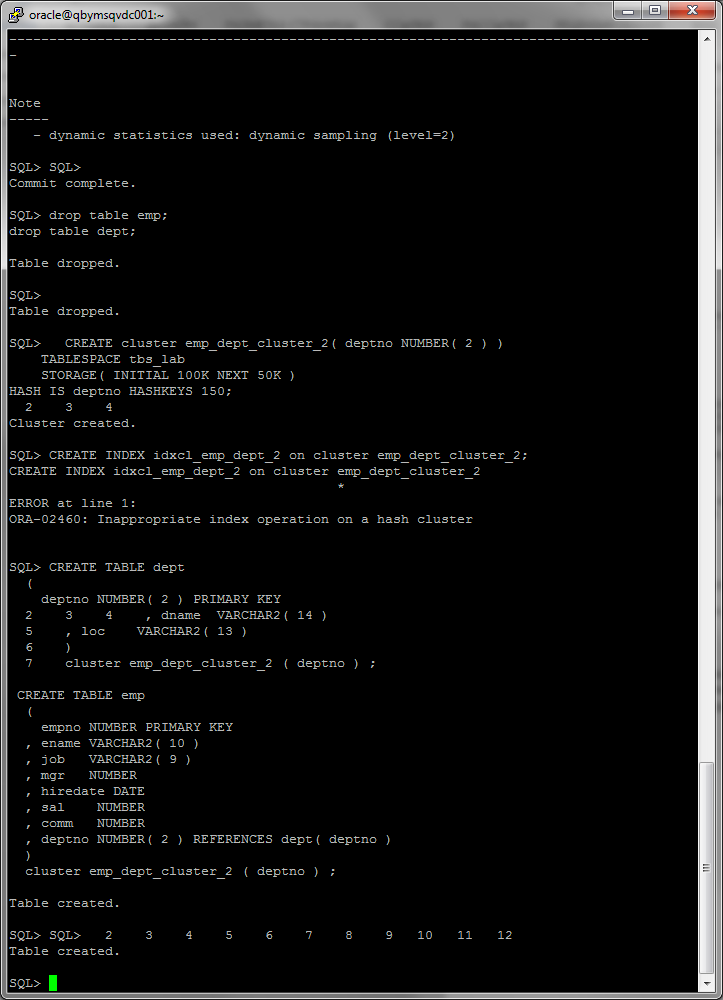
I have cleaned up scheme:



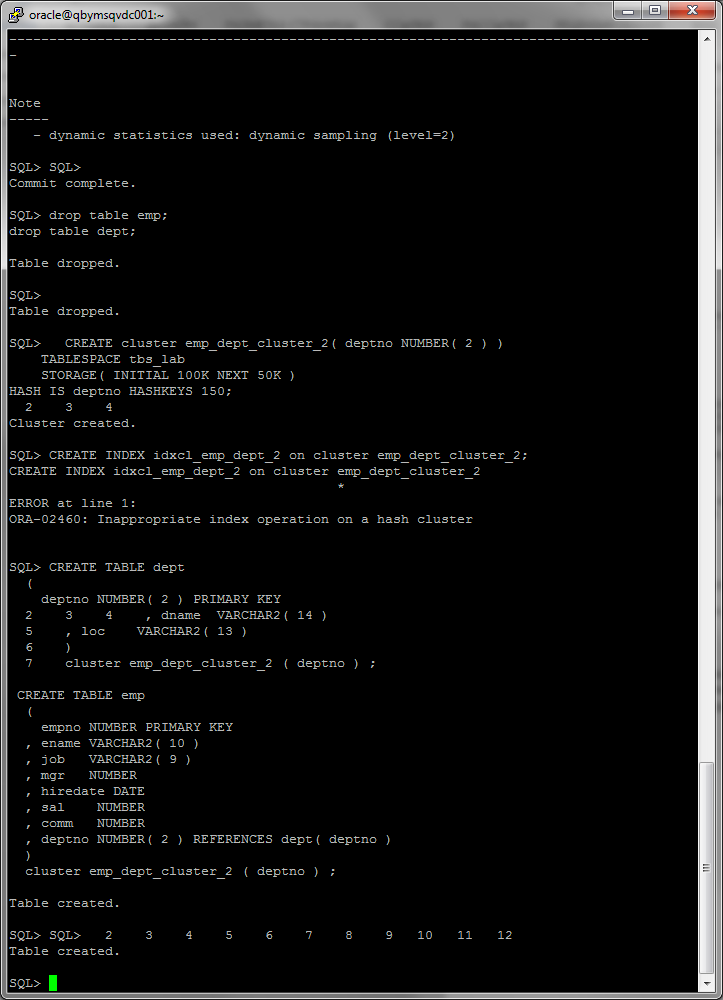
# Hash Clustered Tables

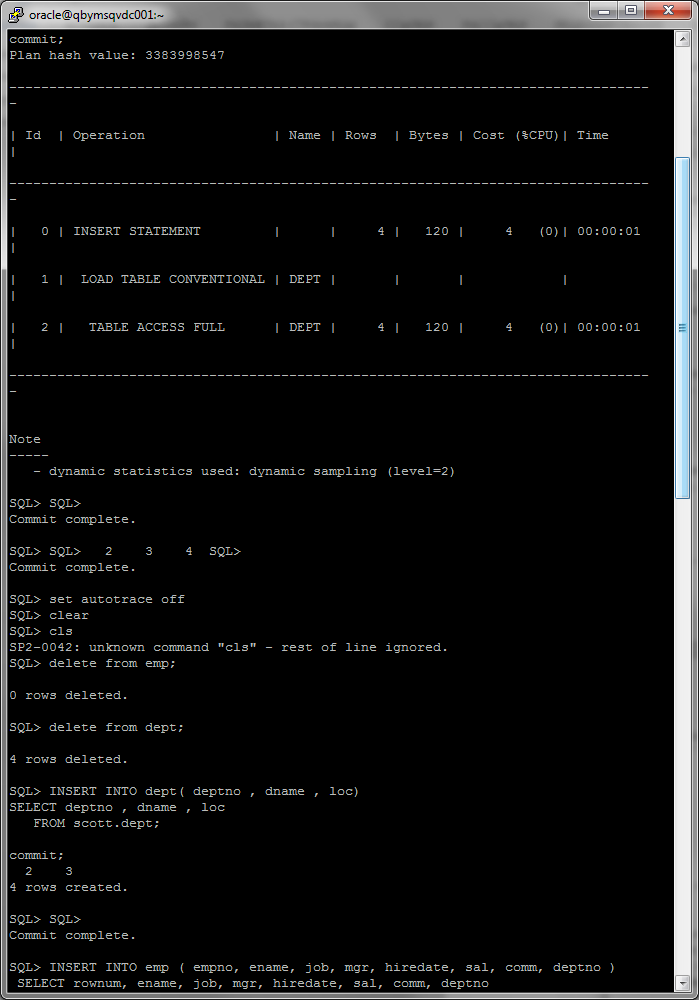
## Task 5: Analyses Cluster Storage by Blocks

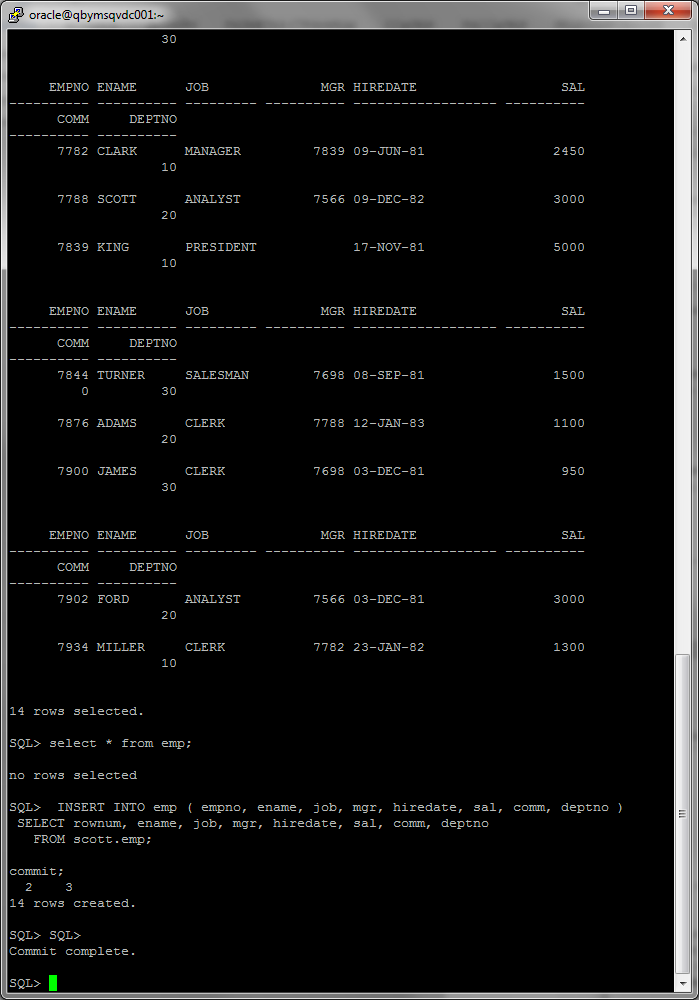
I have created hash cluster:



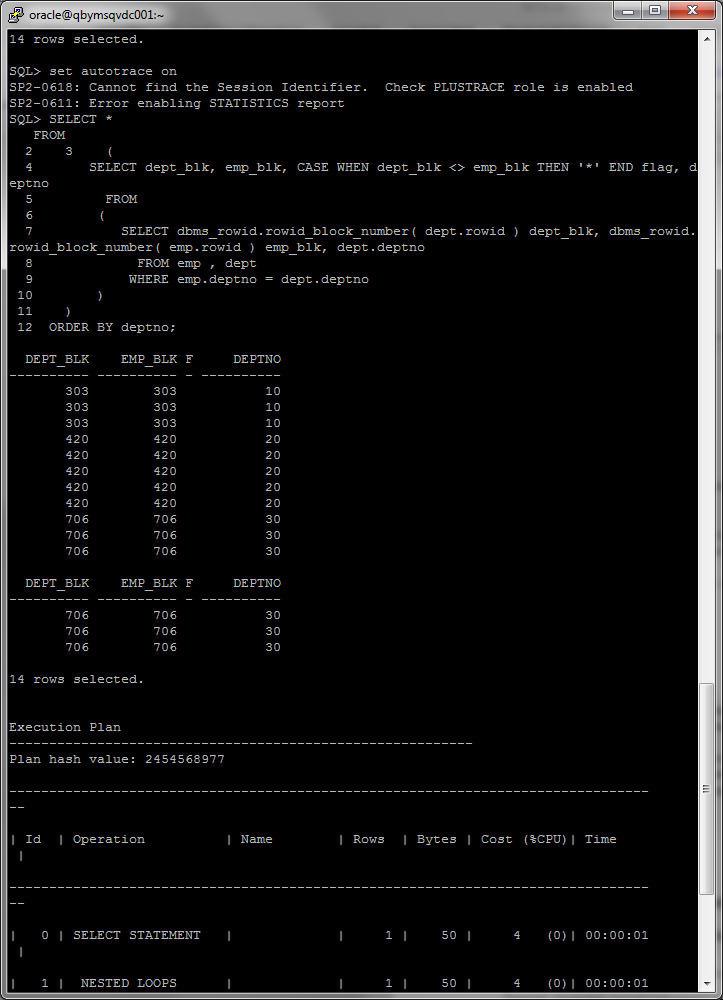
I have created hash clustered tables and inserted data:







All data from one department are stored on the same block:



Explanation:

Hash clustered tables are very similar in concept to the index clustered tables with one main exception: the cluster key index is replaced with a hash function. The data in the table is the index; there is no physical index. Oracle will take the key value for a row, hash it using either an internal function or one you supply, and use that to figure out where the data should be on disk. Therefore, with this options, all data by one number of department are stored together – 10 department in 303 block, 20 – 420 etc.

Advantages of this type of storage:

The hash cluster is allocated right from the beginning. The number of HASHKEYs in a hash cluster is a fixed size. It simply limits the number of unique hash keys that can be generated for this cluster.

I have cleaned up scheme:

