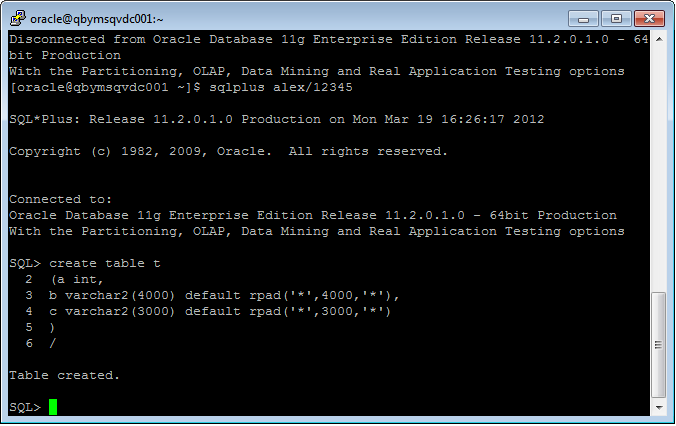
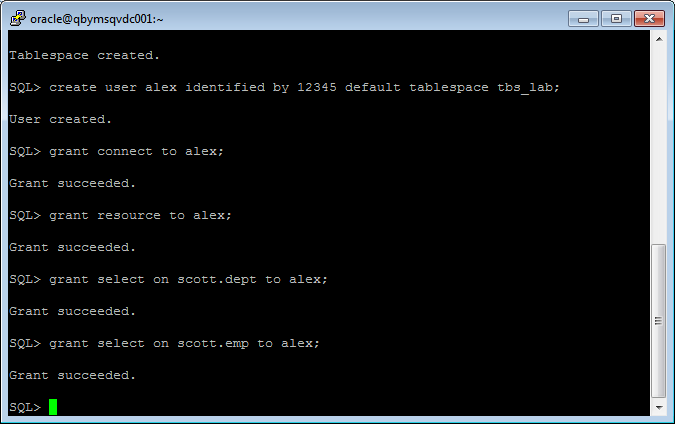
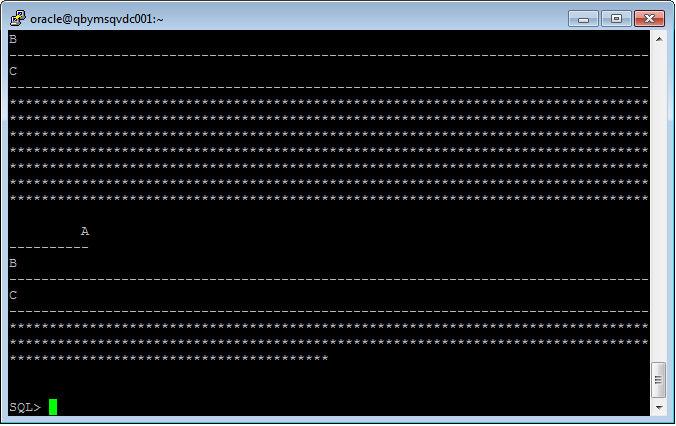
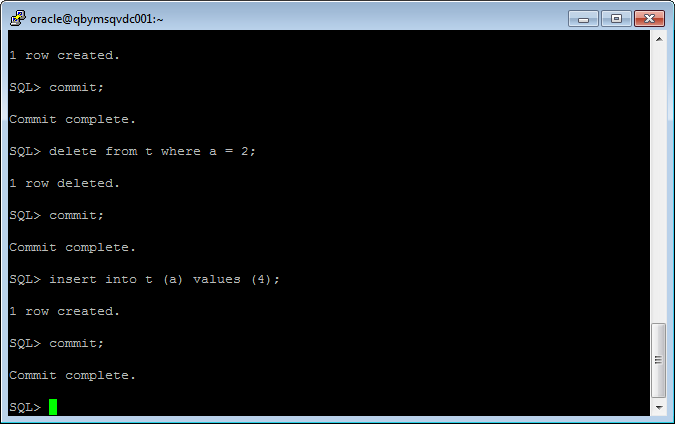
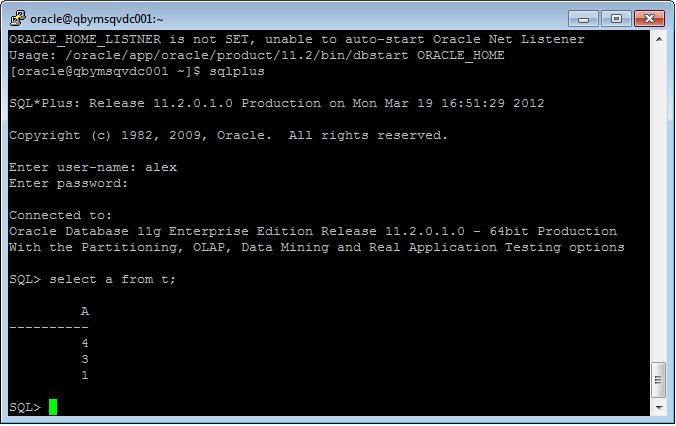
# Task 1:

**Task Results:**



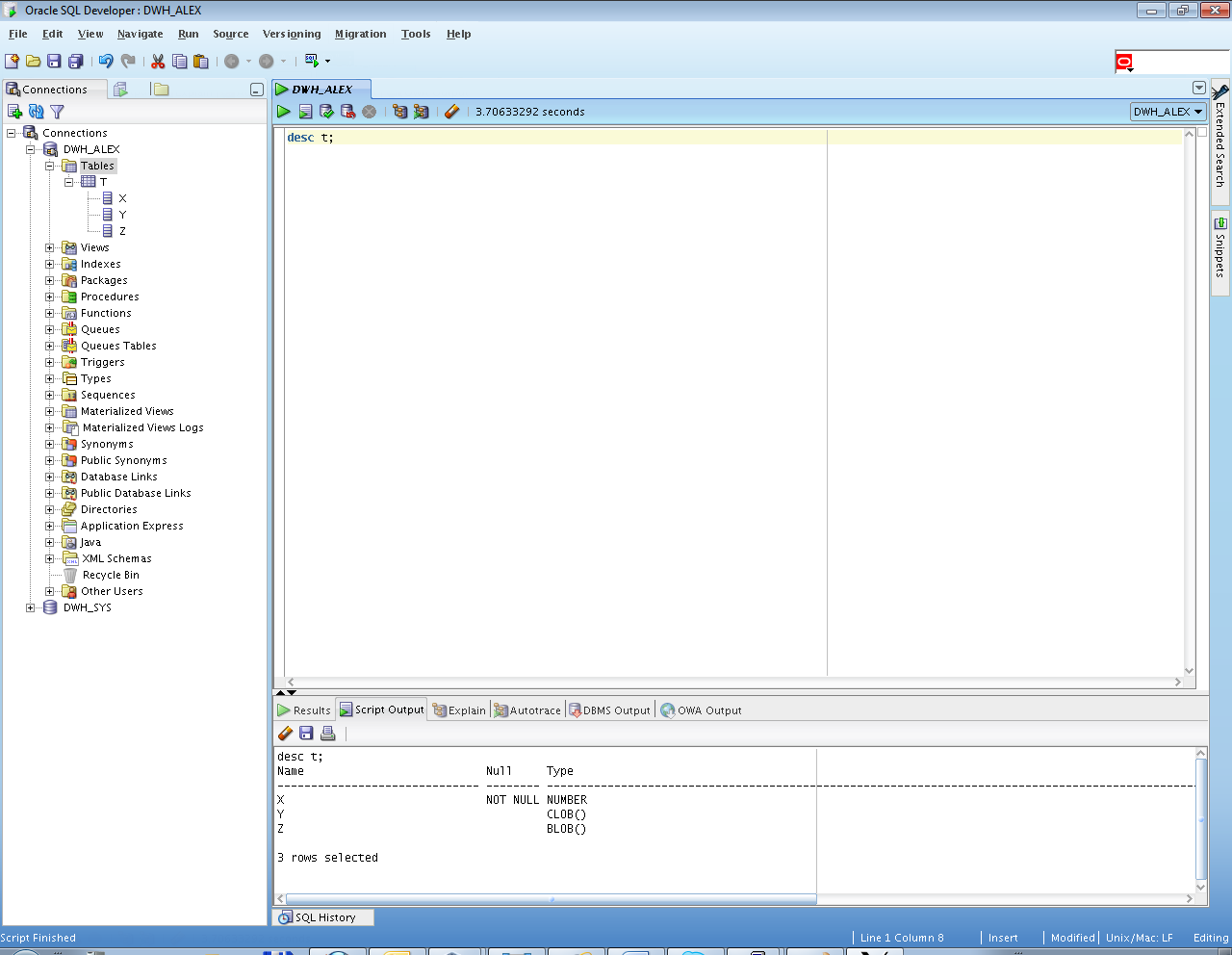




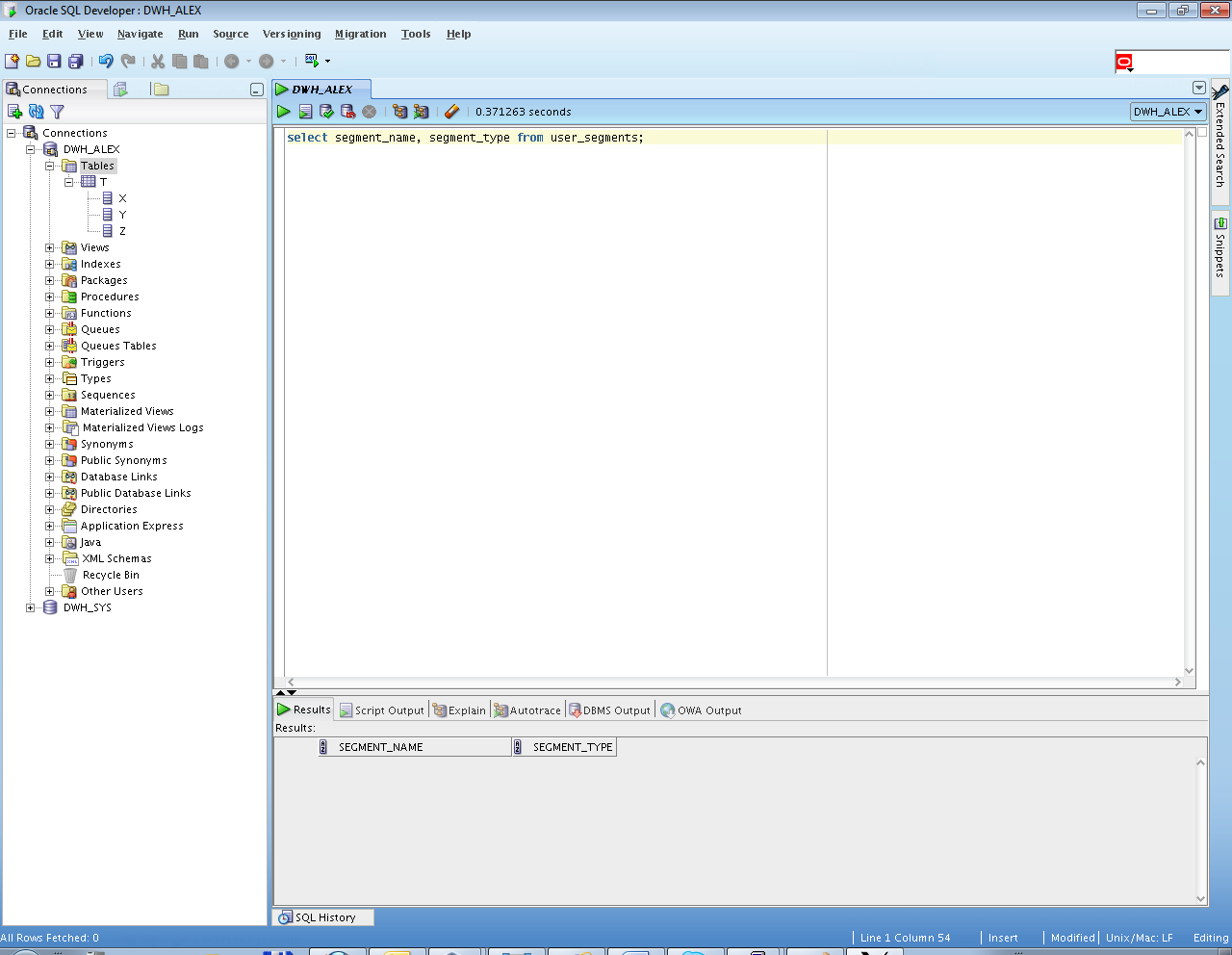
## Task 2 – Understanding Low level of data abstraction: Heap Table Segments

**Task Results:**

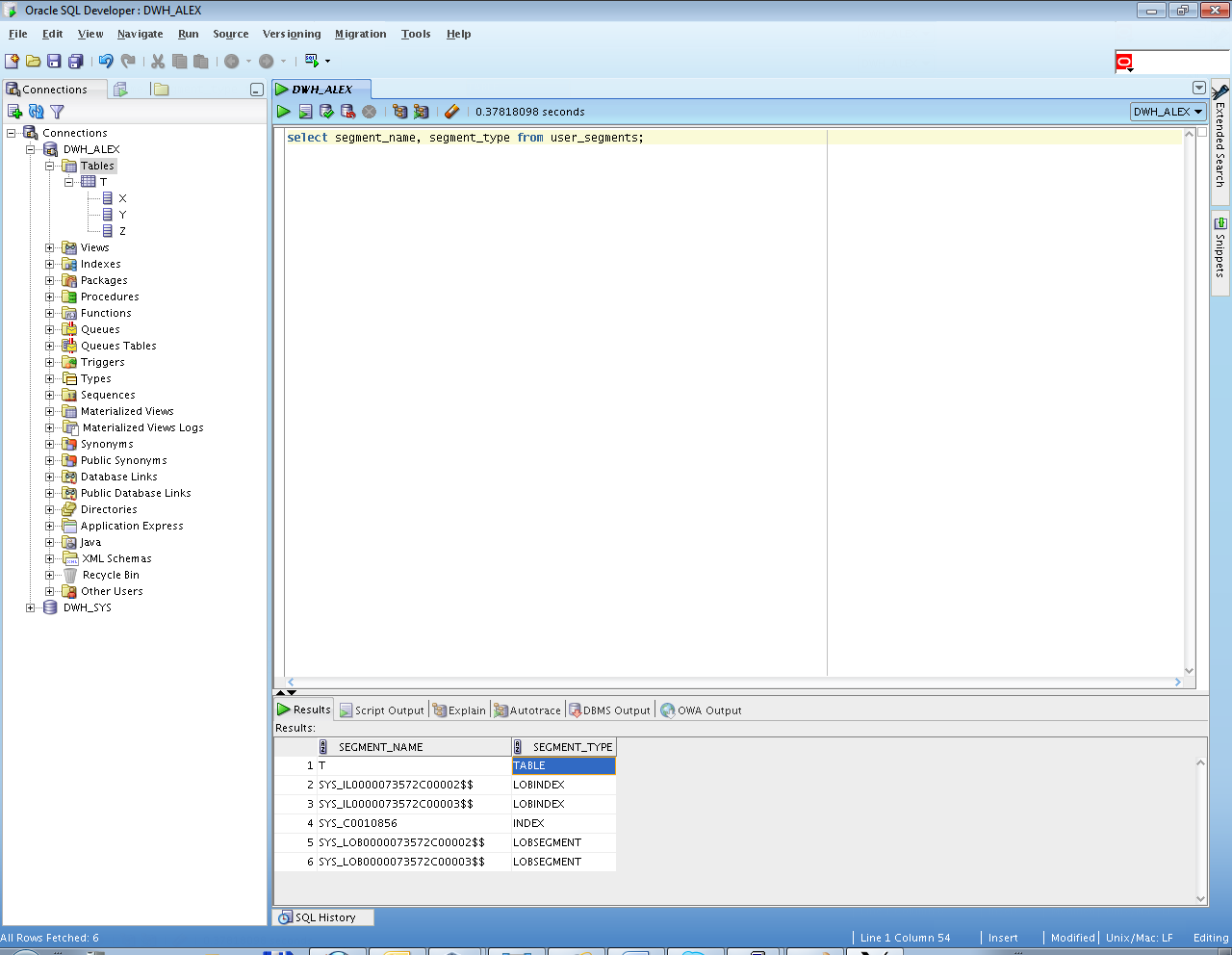
Step 1: Table Created;



Step 2: Empty selection



**Step 3**: Expected Result (After SEGMENT CREATION IMMEDIATE):



## Task 3: Compare performance of using IOT tables

Step 1:

CREATE TABLE emp AS

SELECT

object\_id empno

, object\_name ename

, created hiredate

, owner job

FROM

all\_objects

/

Create Index:

# alter table emp add constraint emp\_pk primary key(empno)

Calculate Statistic:

begin

dbms\_stats.gather\_table\_stats( user, 'EMP', cascade=>true );

end;

Step 2:

CREATE TABLE heap\_addresses

(

empno REFERENCES emp(empno) ON DELETE CASCADE

, addr\_type VARCHAR2(10)

, street VARCHAR2(20)

, city VARCHAR2(20)

, state VARCHAR2(2)

, zip NUMBER

, PRIMARY KEY (empno,addr\_type)

)

/

Step 3:

CREATE TABLE iot\_addresses

(

empno REFERENCES emp(empno) ON DELETE CASCADE

, addr\_type VARCHAR2(10)

, street VARCHAR2(20)

, city VARCHAR2(20)

, state VARCHAR2(2)

, zip NUMBER

, PRIMARY KEY (empno,addr\_type)

)

ORGANIZATION INDEX

/

Step 4: Initial inserts:

INSERT INTO heap\_addresses

SELECT empno, 'WORK' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno , 'WORK' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

--

INSERT INTO heap\_addresses

SELECT empno, 'HOME' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno, 'HOME' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

--

INSERT INTO heap\_addresses

SELECT empno, 'PREV' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno, 'PREV' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

--

INSERT INTO heap\_addresses

SELECT empno, 'SCHOOL' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno, 'SCHOOL' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

Commit;

Step 5: Calculate statistic:

exec dbms\_stats.gather\_table\_stats( $username$, 'HEAP\_ADDRESSES' );

exec dbms\_stats.gather\_table\_stats( $username$, 'IOT\_ADDRESSES' );

**Step 6:** Compare Trace and Performance:

Explain 1:

SELECT \*

FROM emp ,

heap\_addresses

WHERE emp.empno = heap\_addresses.empno

AND emp.empno = 100;

Explain 2:

SELECT \*

FROM emp ,

iot\_addresses

WHERE emp.empno = iot\_addresses.empno

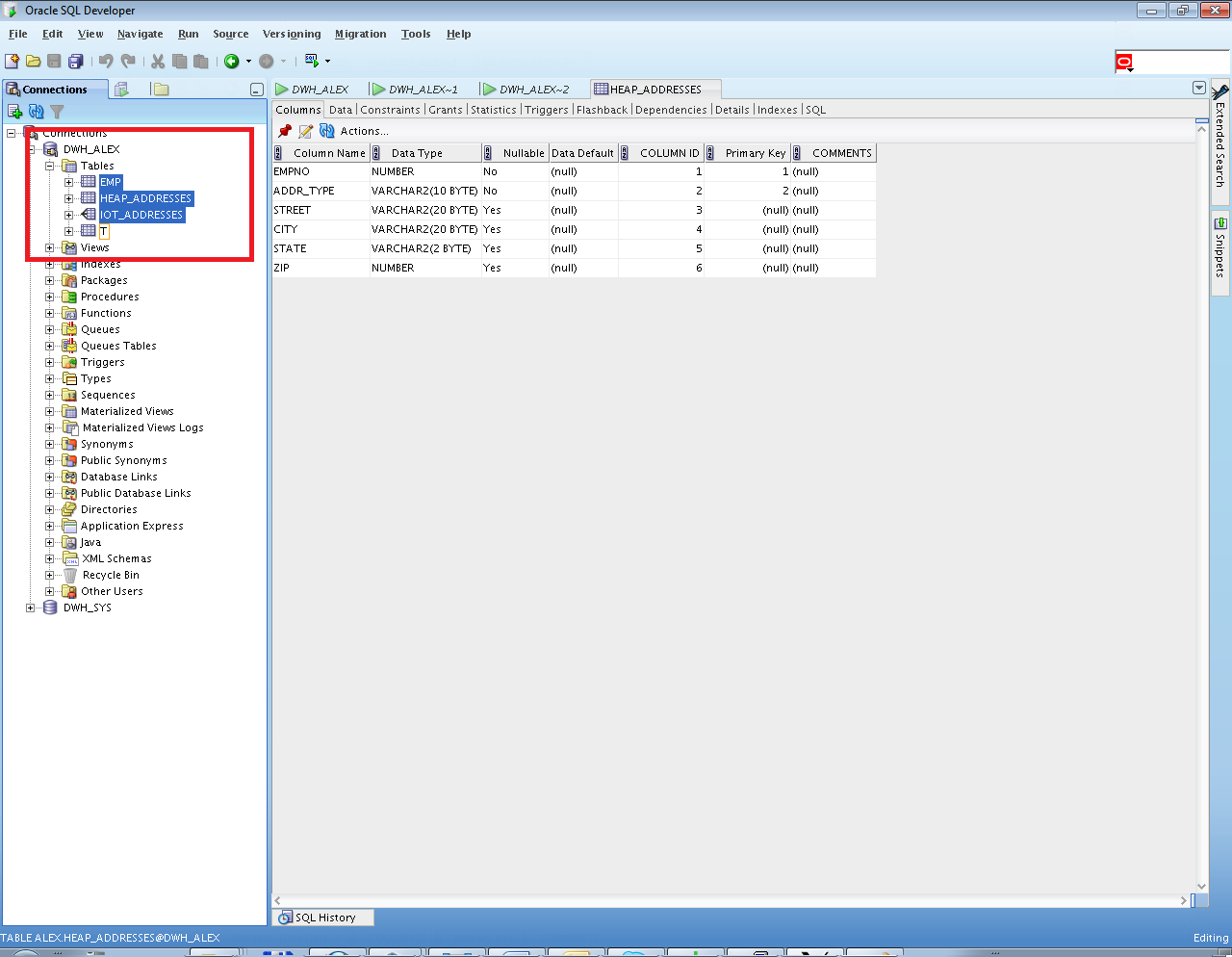
AND emp.empno = 100;

Compare results and explain Cost value calculation and difference on execution plan.

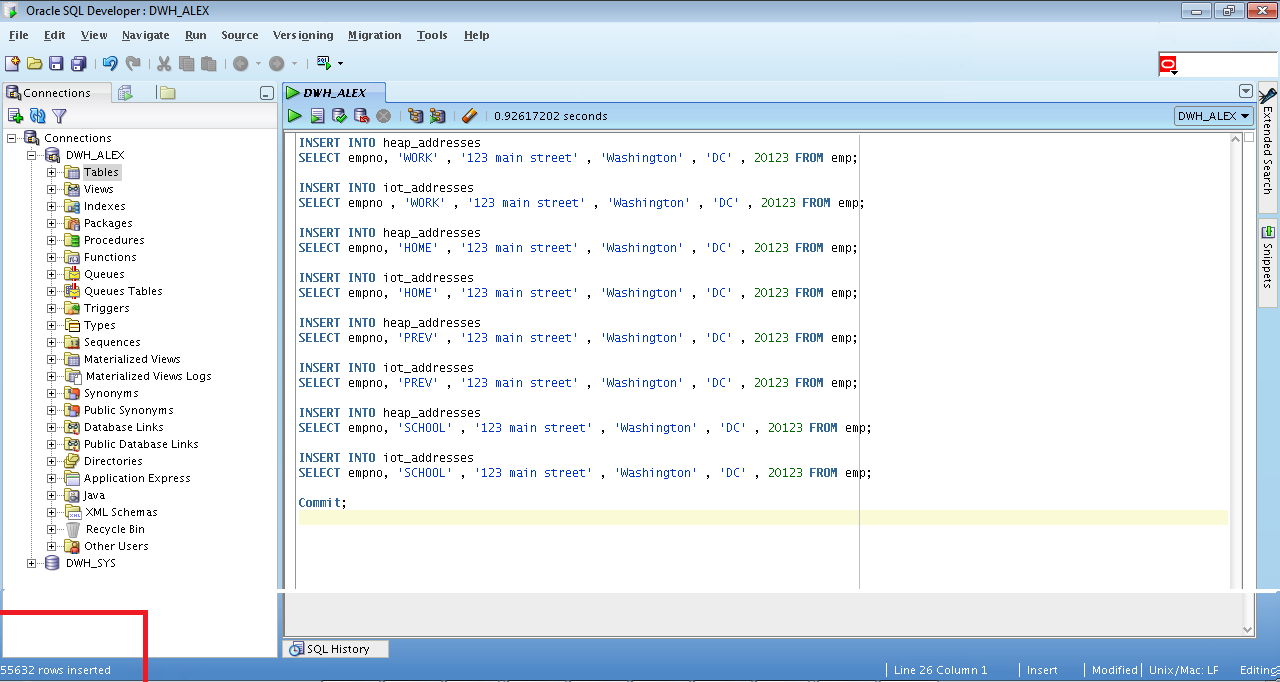
Step 7: Drop all tables;

**Task Results:**

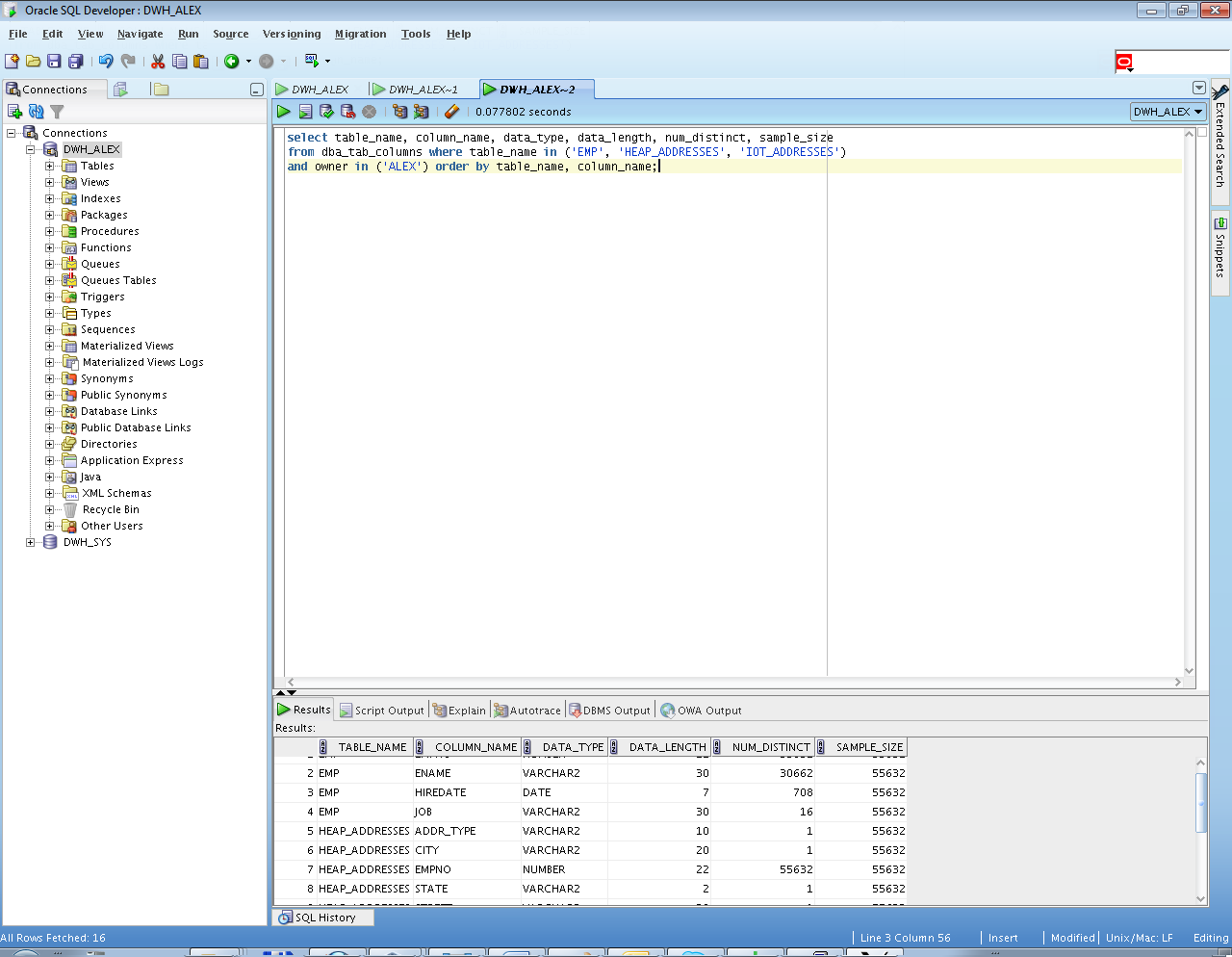
Step 1-3: Tables Created;



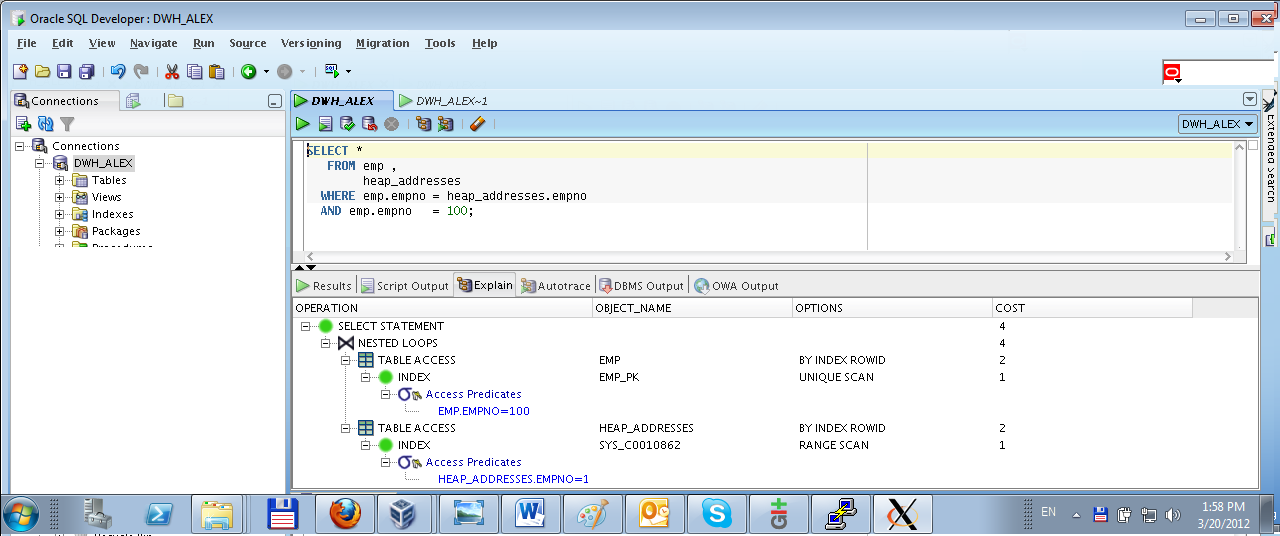
Step 4: Inserting rows;

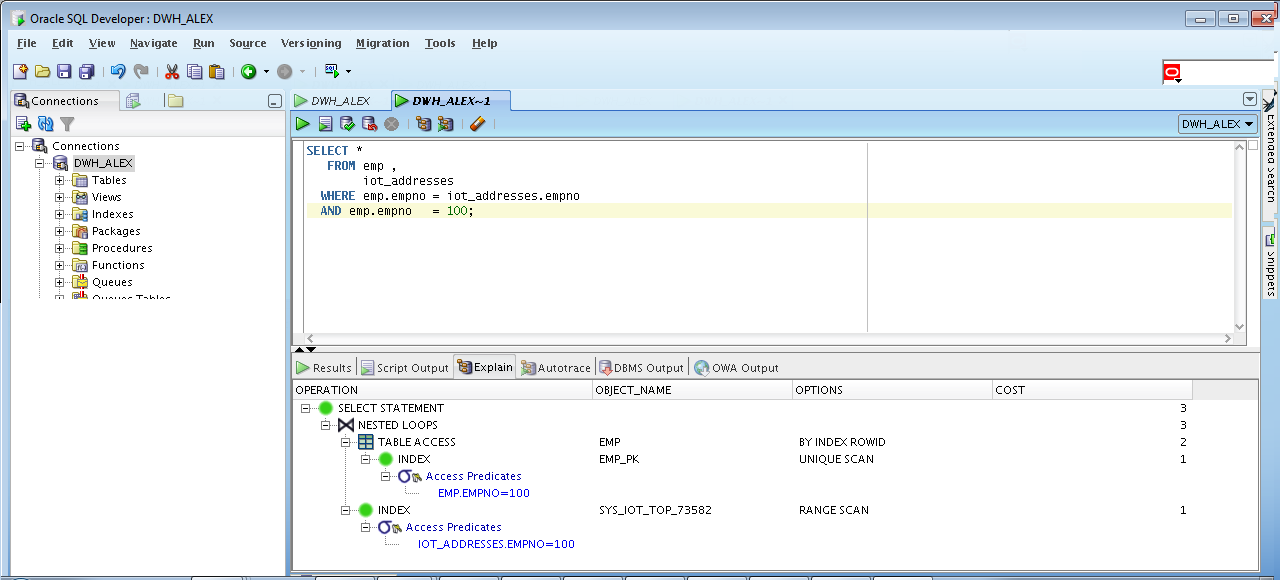


Step 5: Calculate statistic;



**Step 6:**

****

****

Expected Heap table cost > IOT table cost

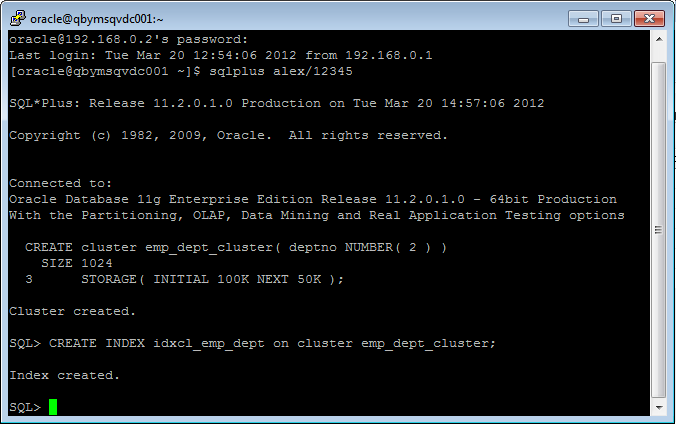
Prepare screenshots and write explanation why cost is different.

В данном случае видим, что cost IOT меньше из-за того, что ищем empno и сразу же получаем необходимые данные. Cost heap больше, т.к. ищем empno, возвращаемся и по empno получаем данные. IOT эффективен в случае поиска по индексу, если же поиск осуществляется не по индексу, то heap будет эффективнее.

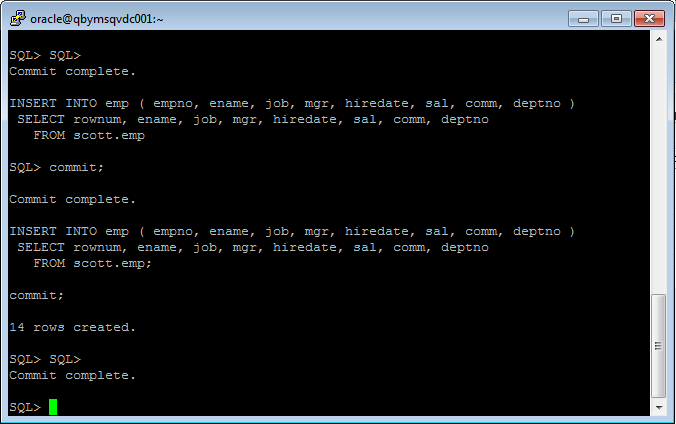
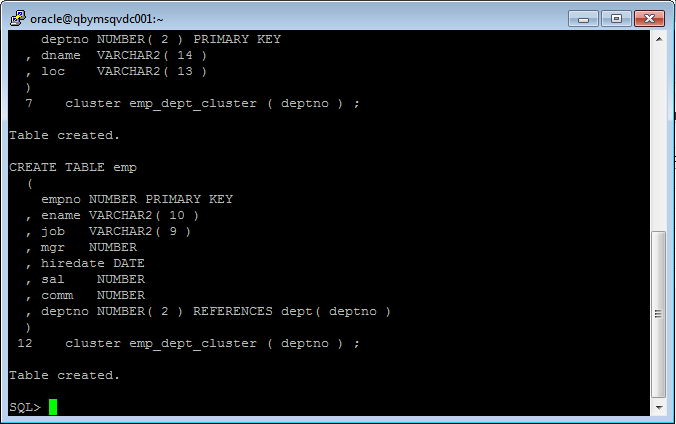
## Task 4: Analyses Cluster Storage by Blocks

**Task Results:**

Step 1-2: Cluster Created and Index Created;



Step 3-4: Table Created and Initialize Inserting rows Finished;

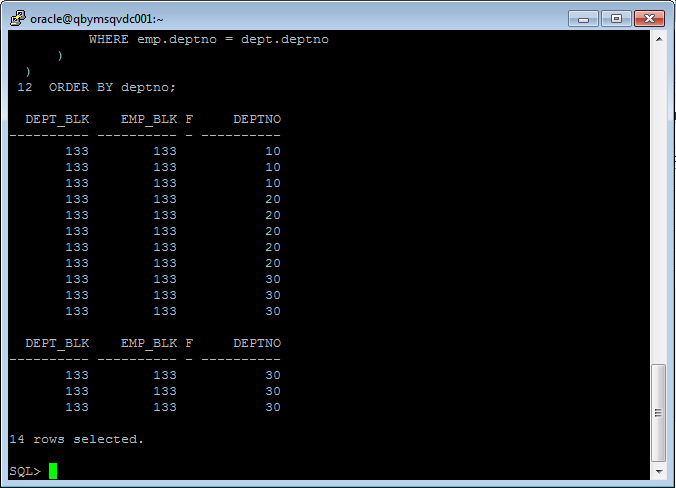


**Step 5:**

Expected All data have to be stored on the same block.

Prepare screenshots and write explanation why data storage look like on select.

Describe advantages of this type of storage.



Кластеры представляют собой группу из одной или нескольких таблиц, которые физически хранятся в одном блоке базы данных. Строки с общим ключом физически хранятся близко друг к другу. Таким образом, данные из нескольких таблиц хранятся в одном блоке.

## Task 5: Analyses Cluster Storage by Blocks

Make the same steps as on task 4.1 with one difference create Hash Cluster.

NOTE: To find more information about hash cluster create statements use: Hash Clustered Tables

1. Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g Programming; Techniques and Solutions, Second Edition; Thomas Kyte ; 2010 (Chapter 10: Database Tables - Hash Clustered Tables).

Step 1:

CREATE cluster emp\_dept\_hash\_cluster( deptno NUMBER( 2 ) )  
 SIZE 1024   
 STORAGE( INITIAL 100K NEXT 50K )

HASH IS deptno HASHKEYS 150;

Step 2:

CREATE TABLE dept

(

deptno NUMBER( 2 ) PRIMARY KEY

, dname VARCHAR2( 14 )

, loc VARCHAR2( 13 )

)

cluster emp\_dept\_hash\_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\_hash\_cluster ( deptno ) ;

Step 3:

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;

**Step 4:**

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

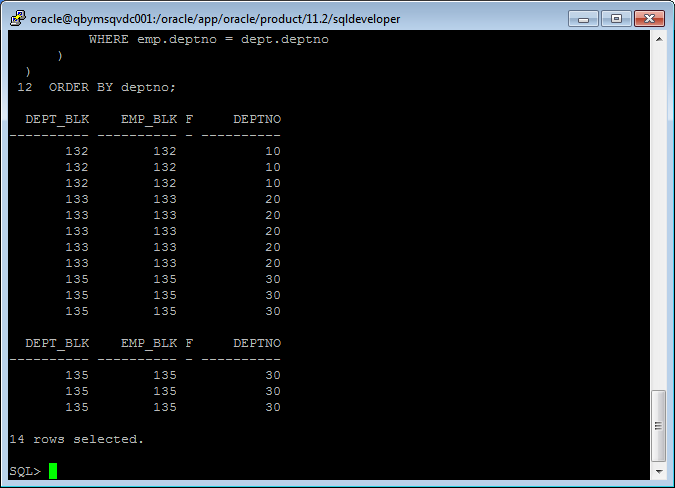
FROM emp , dept

WHERE emp.deptno = dept.deptno

)

)

ORDER BY deptno;



Hash кластерные таблицы похожи на кластерные таблицы. Отличие в том, что кластерный ключ заменяется на hash функцию. Что бы найти, где данные хранятся на диске, используется hashkey. Недостаток – невозможность изменения диапазона сканирования таблицы в кластере без добавления индекса к таблице.