U1M3.LW.Database Types of Tables, Indexes

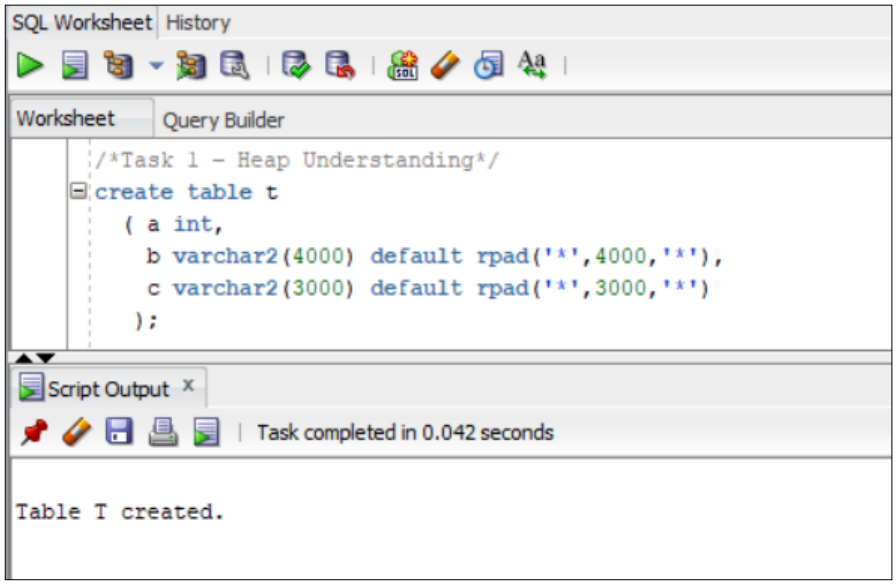
Shkrabatouskaya Vera

https://github.com/VeraShkrabatouskaya/DataMola_Data-Camping-2022

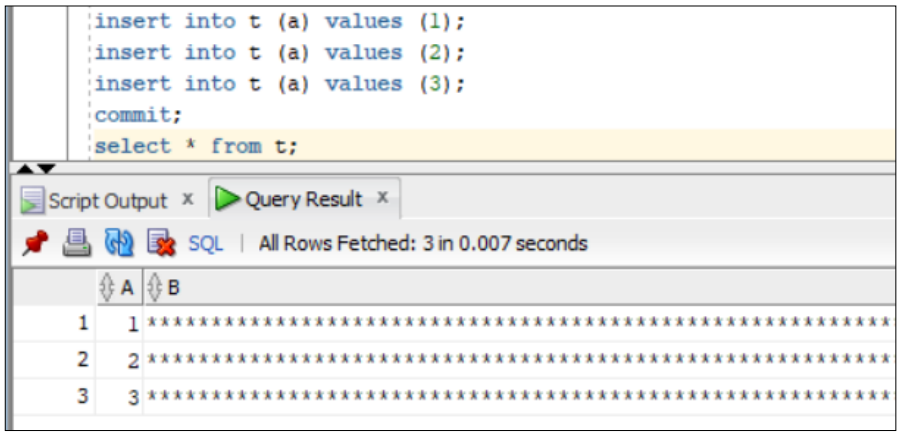
2. Heap Organized Tables

2.1. Task 1 – Heap Understanding

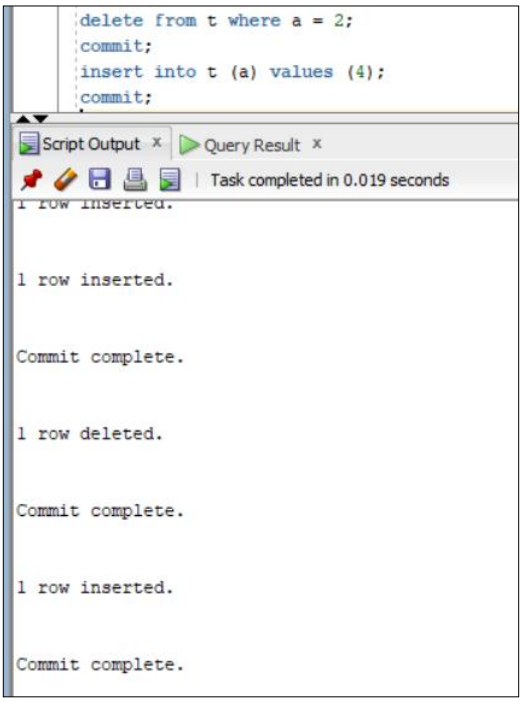
Step 1: Create table t



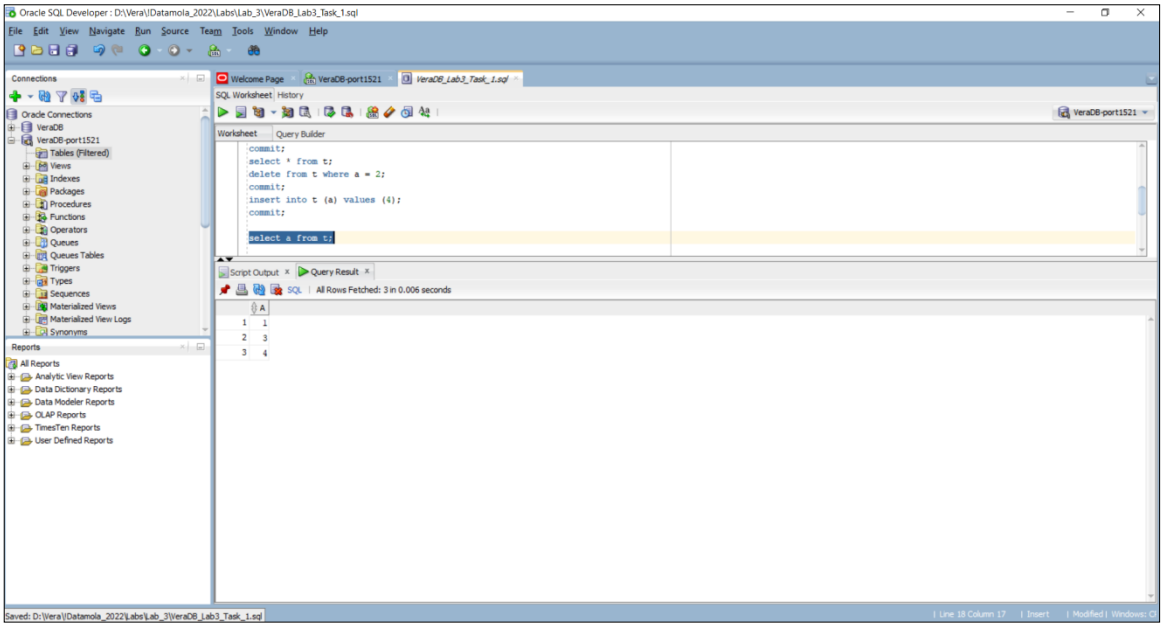
Step 2: Insert values into the table



Change values in the table



<u>Step 3:</u> Data results

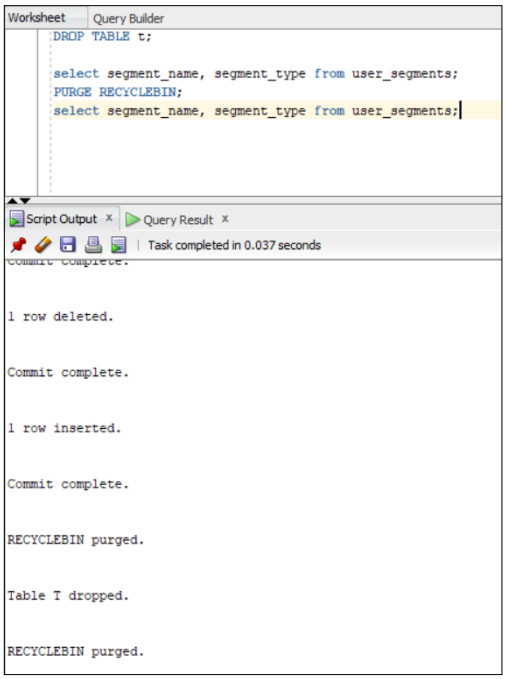


Expected:

```
select a from t;
A
----------
1
4
3
```

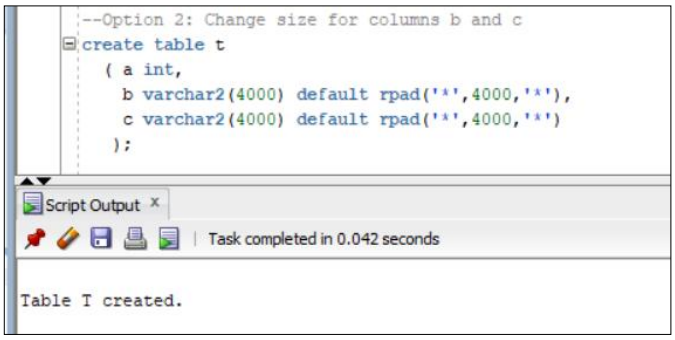
We may notice that the data is unloaded in a different order than expected. Let's try the following option.

Drop table and purge the recycle bin

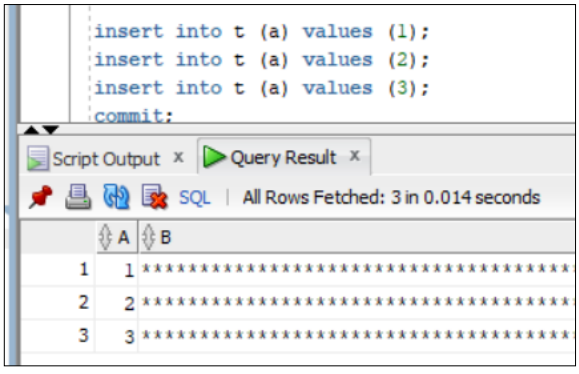


Option 2:

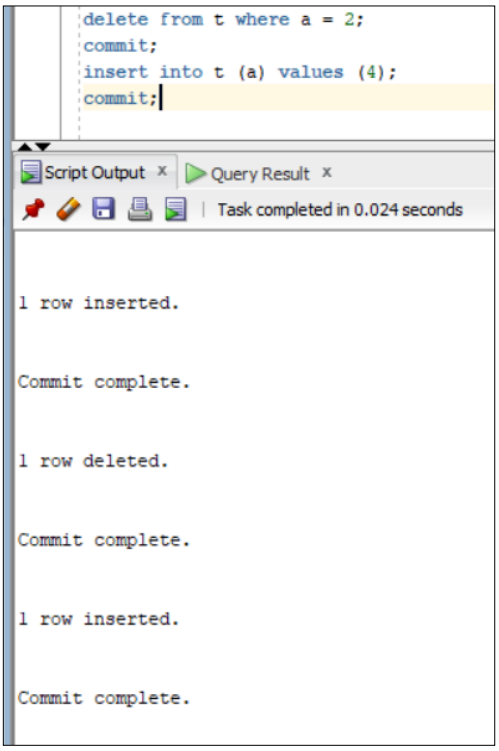
Step 1: Create table t and resize column c



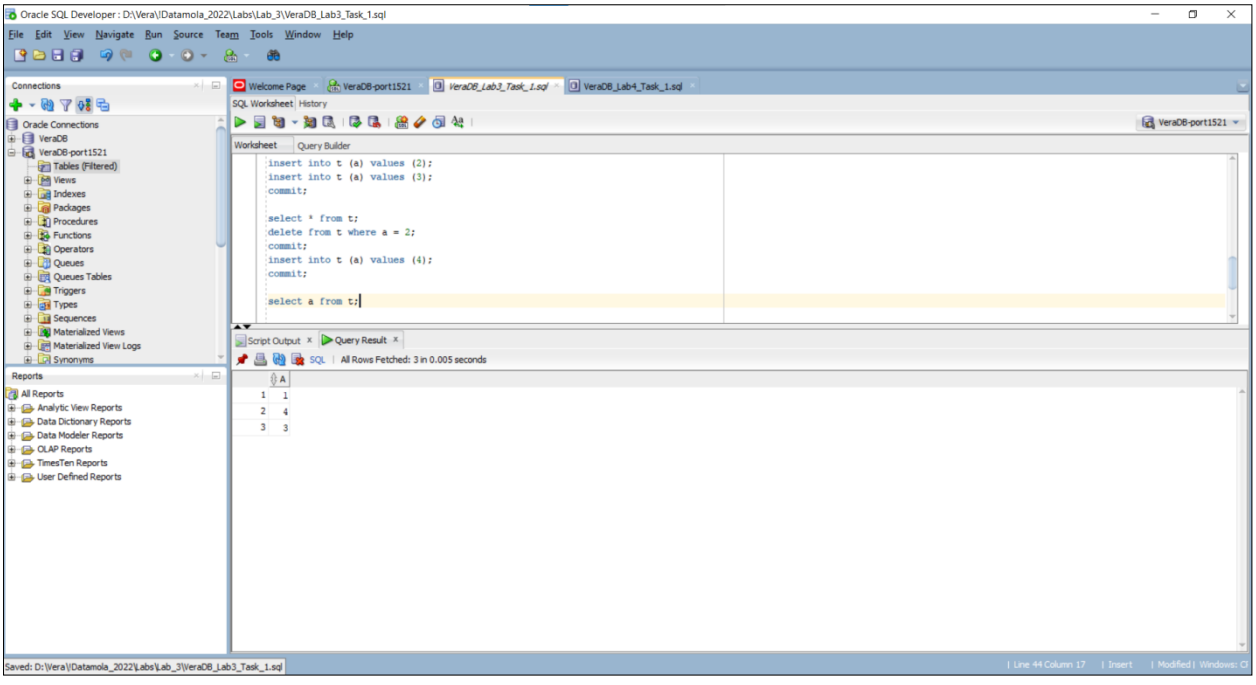
Step 2: Insert values into the table



Change values in the table



<u>Step 3:</u> Data results

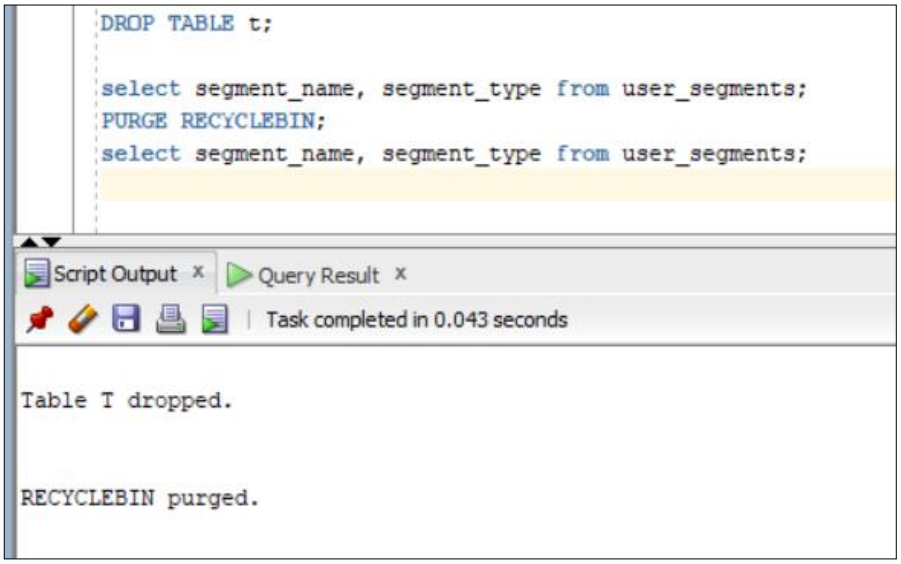


Expected:

```
select a from t;
A
----------
1
4
3
```

Summary: We see that the result is the same as expected. It is noticed that when the amount of data in a block of table increases to be appropriate for block size, since data is managed in a heap in a table like this, as space becomes available, it will be reused.

Drop table and purge the recycle bin

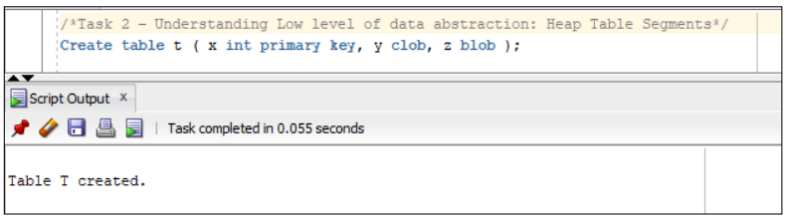


Code: Task 1

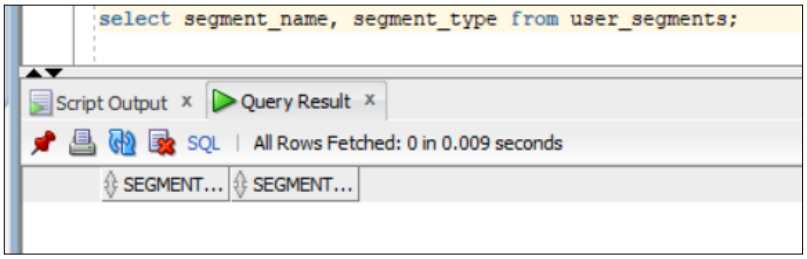
```
/*Task 1 – Heap Understanding*/

create table t
 ( a int,
   b varchar2(4000) default rpad('*',4000,'*'),
   c varchar2(3000) default rpad('*',3000,'*')
  );

insert into t (a) values (1);
insert into t (a) values (2);
insert into t (a) values (3);
commit;

select * from t;

delete from t where a = 2;
commit;
insert into t (a) values (4);
commit;

select a from t;

DROP TABLE t;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;

/*Option 2: Change size for column c*/

create table t
 ( a int,
   b varchar2(4000) default rpad('*',4000,'*'),
   c varchar2(4000) default rpad('*',4000,'*')
  );

insert into t (a) values (1);
insert into t (a) values (2);
insert into t (a) values (3);
commit;

select * from t;

delete from t where a = 2;
commit;
insert into t (a) values (4);
commit;

select a from t;

DROP TABLE t;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;
```

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

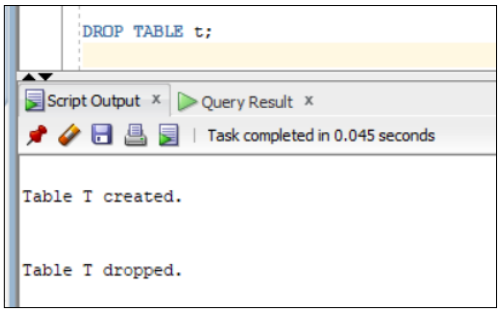
Step 1: Create table



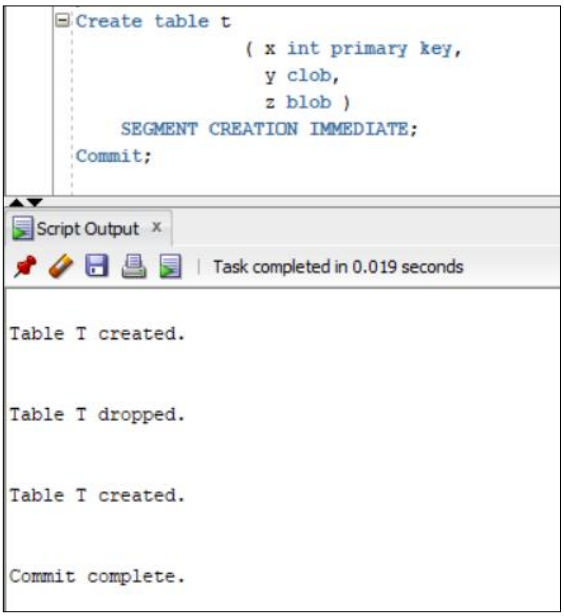
Step 2: View all shared memory segments



Drop table



Step 3: Create table with immediate segment creation



Step 4: View all shared memory segments

```
select segment_name, segment_type from user_segments;
```

Script Output | Query Result | SQL | All Rows Fetched: 6 in 0.013 seconds

	SEGMENT_NAME	SEGMENT_TYPE
1	SYS_C007074	INDEX
2	SYS_IL0000066815C00002$$	LOBINDEX
3	SYS_IL0000066815C00003$$	LOBINDEX
4	SYS_LOB0000066815C00002$$	LOBSEGMENT
5	SYS_LOB0000066815C00003$$	LOBSEGMENT
6	T	TABLE

Step 5: Show the DDL for all tables in the current user's schema. DBMS_METADATA is the PL/SQL package that implements Metadata API. It allows callers to retrieve metadata from the database Dictionary.

```
SELECT DBMS_METADATA.GET_DDL('TABLE','T') FROM dual;
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.188 seconds

	DBMS_METADATA.GET_DDL('TABLE','T')
1	CREATE TABLE "VSHKRABATOVSKAYA"."T" ("X" NUMBER(*,0), "Y" CLOB, "Z" BLOB, PRIMARY KEY ("X") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 16...

Drop table and purge the recycle bin

```
DROP TABLE t;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;
```

Script Output | Query Result | Task completed in 0.075 seconds

```
Table T created.

Commit complete.

Table T dropped.

RECYCLEBIN purged.
```

Summary: Segment creation on demand, or deferred segment creation is a space saving feature of Oracle Database. The functionality can be controlled by the DEFERRED_SEGMENT_CREATION initialization parameter, which is set to TRUE by default. The default behavior is altered by using the IMMEDIATE clause. When using the IMMEDIATE CREATE SEGMENT, the place for the segment is determined immediately.

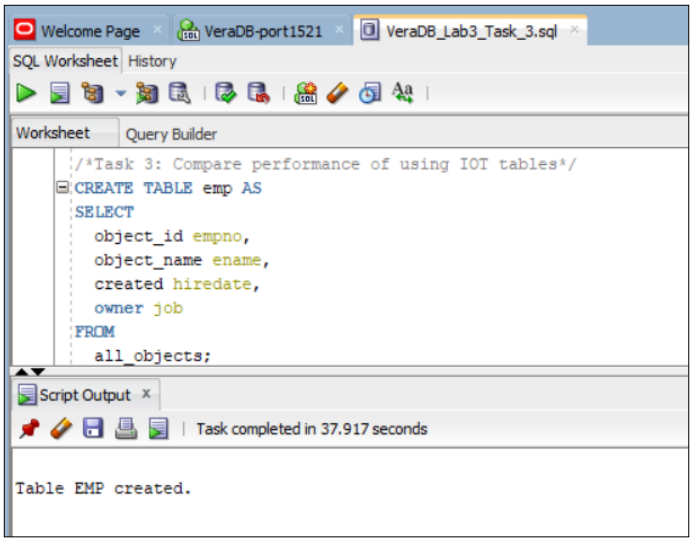
Code: Task 2

```
/*Task 2 – Understanding Low level of data abstraction: Heap Table Segments*/

Create table t ( x int primary key, y clob, z blob );

select segment_name, segment_type from user_segments;

DROP TABLE t;

select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;

Create table t
            ( x int primary key,
              y clob,
              z blob )
   SEGMENT CREATION IMMEDIATE;
Commit;

select segment_name, segment_type from user_segments;

SELECT DBMS_METADATA.GET_DDL('TABLE','T') FROM dual;

DROP TABLE t;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;
```

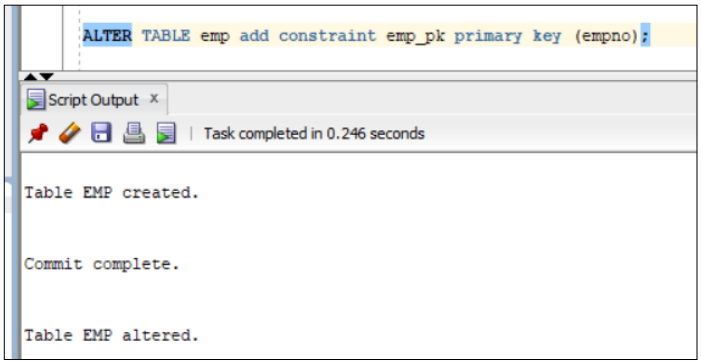
3. Index Organized Tables

Task 3: Compare performance of using IOT tables

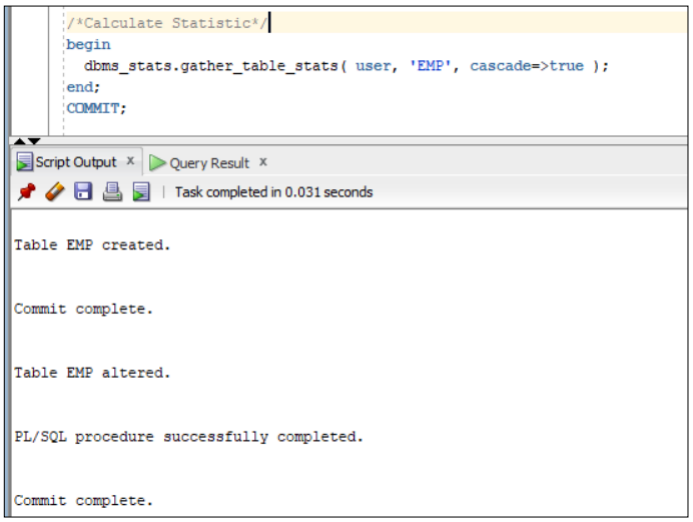
Step 1: Create table emp



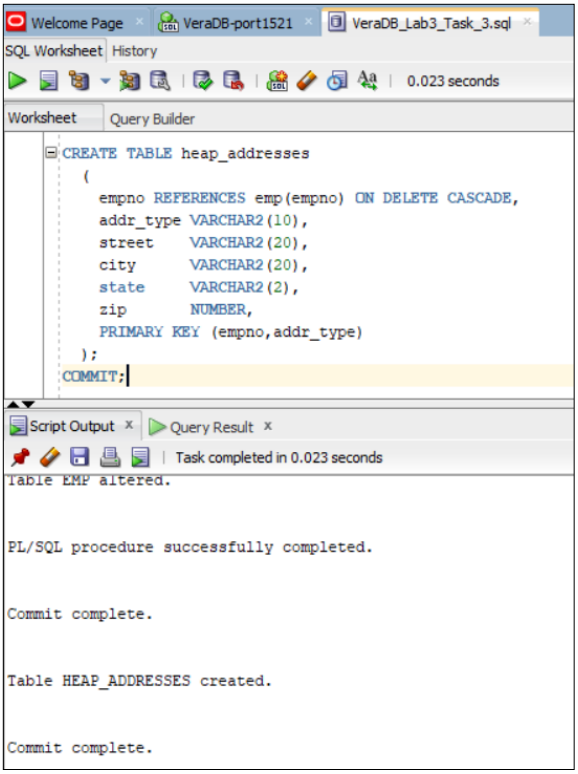
Create Index. A primary key was created for the "empno" field.



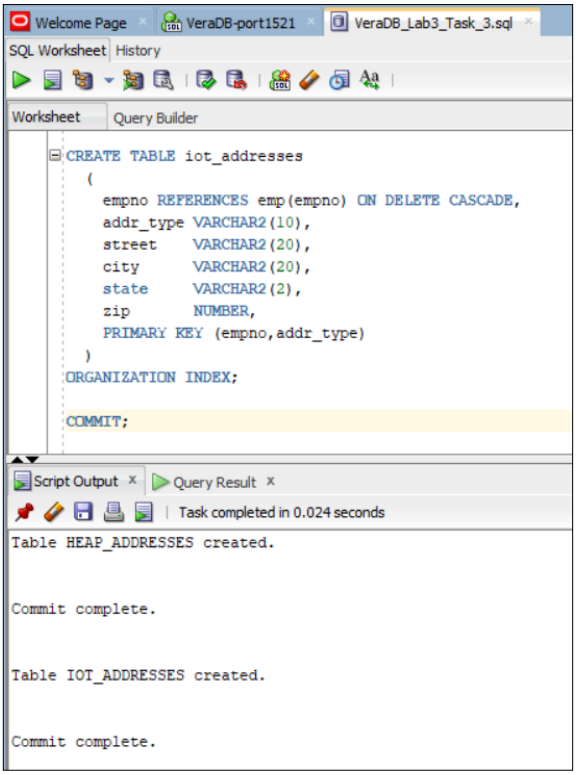
Calculate statistics using the DBMS_STATS package



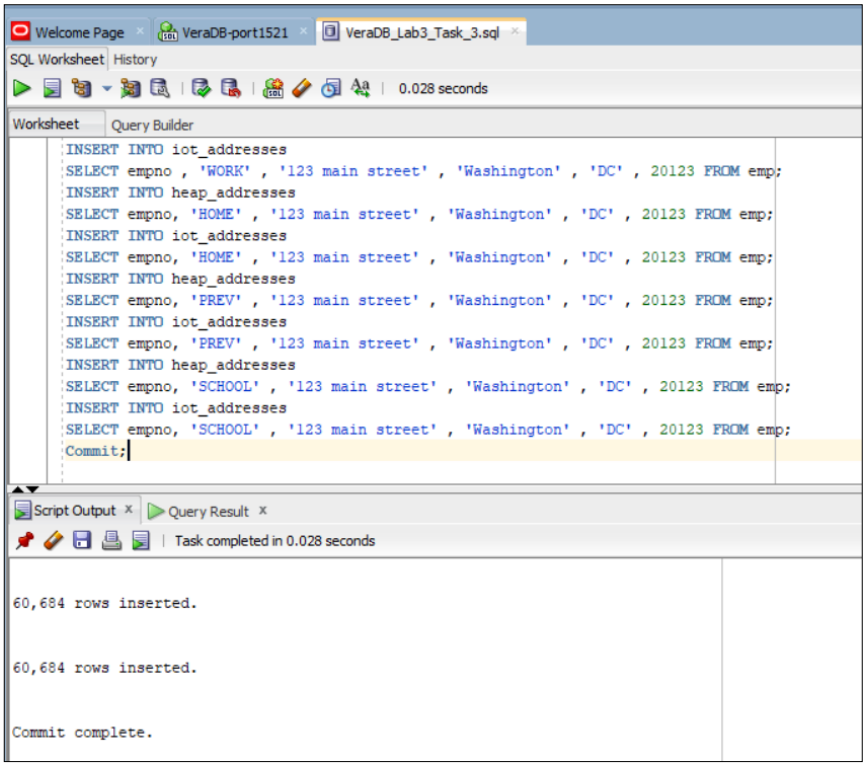
Step 2: Create table heap_addresses



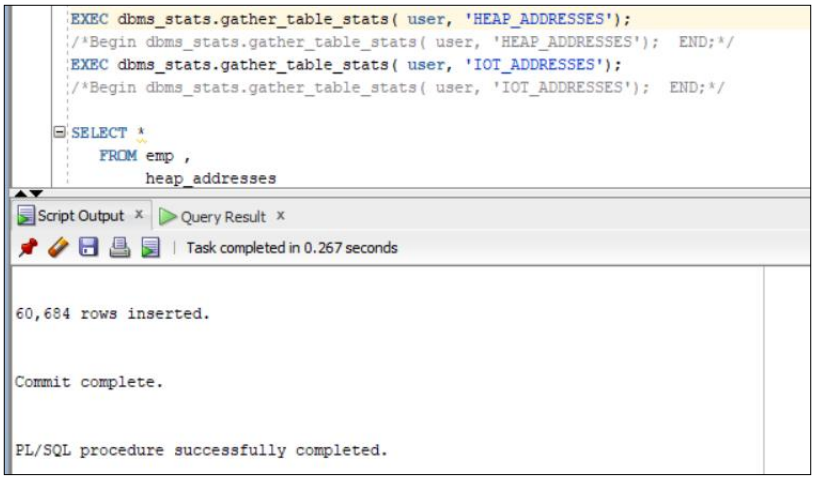
Step 3: Create table iot_addresses. The key phrase ORGANIZATION INDEX indicates that this table is an IOT, not a traditional table. Index Organized Tables (IOT) have their primary key data and non-key column data stored within the same B*Tree structure.

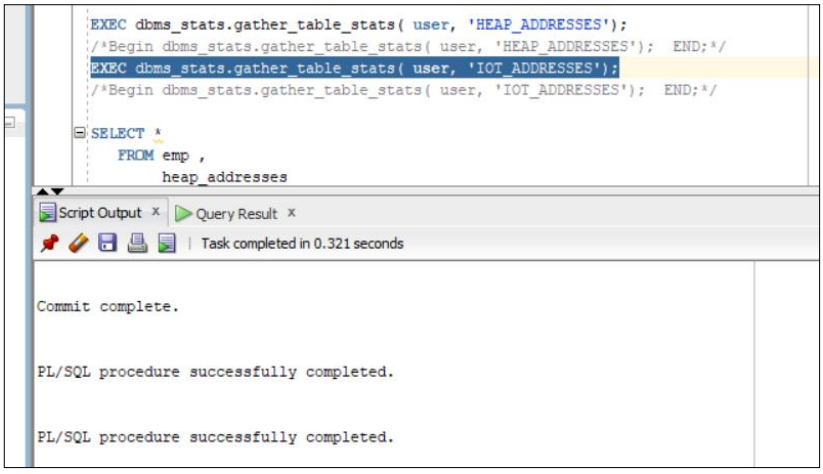


Step 4: Initial inserts:



Step 5: Calculate statistics using the DBMS_STATS package

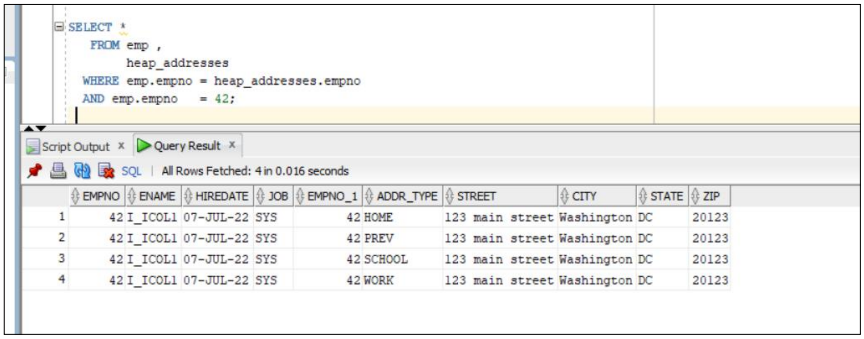




Step 6: Compare Trace and Performance:

Heap Table

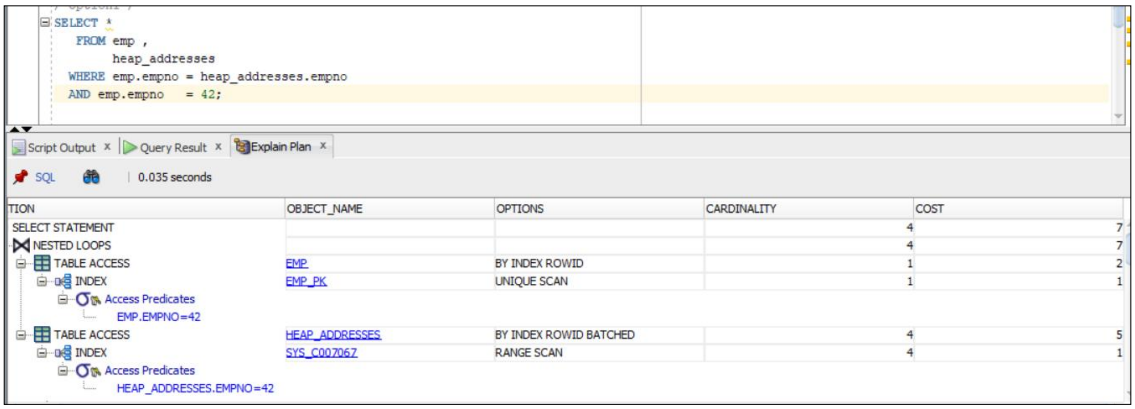
Using Explain Plan



```
SELECT *
  FROM emp ,
       heap_addresses
 WHERE emp.empno = heap_addresses.empno
 AND emp.empno   = 42;
```

Script Output | Query Result | SQL | All Rows Fetched: 4 in 0.016 seconds

	EMPNO	ENAME	HIREDATE	JOB	EMPNO_1	ADDR_TYPE	STREET	CITY	STATE	ZIP
1	42	I_ICOL1	07-JUL-22	SYS	42	HOME	123 main street	Washington	DC	20123
2	42	I_ICOL1	07-JUL-22	SYS	42	PREV	123 main street	Washington	DC	20123
3	42	I_ICOL1	07-JUL-22	SYS	42	SCHOOL	123 main street	Washington	DC	20123
4	42	I_ICOL1	07-JUL-22	SYS	42	WORK	123 main street	Washington	DC	20123

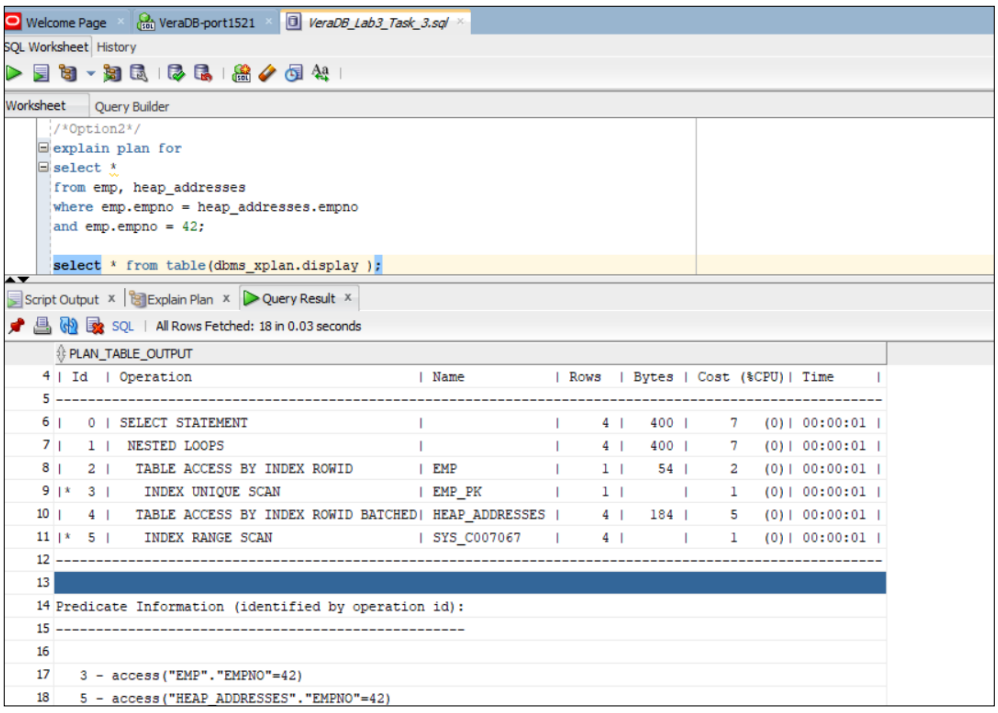


```
SELECT *
  FROM emp ,
       heap_addresses
 WHERE emp.empno = heap_addresses.empno
 AND emp.empno   = 42;
```

Script Output | Query Result | Explain Plan | SQL | 0.035 seconds

TION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4	7
NESTED LOOPS			4	7
TABLE ACCESS	EMP	BY INDEX ROWID	1	2
INDEX	EMP_PK	UNIQUE SCAN	1	1
Access Predicates				
EMP.EMPNO=42				
TABLE ACCESS	HEAP_ADDRESSES	BY INDEX ROWID BATCHED	4	5
INDEX	SYS_C007067	RANGE SCAN	4	1
Access Predicates				
HEAP_ADDRESSES.EMPNO=42				

Option 2



Welcome Page | VeraDB-port1521 | VeraDB_Lab3_Task_3.sql

SQL Worksheet | History

Worksheet | Query Builder

```
/*Option2*/
explain plan for
select *
from emp, heap_addresses
where emp.empno = heap_addresses.empno
and emp.empno = 42;

select * from table(dbms_xplan.display );
```

Script Output | Explain Plan | Query Result | SQL | All Rows Fetched: 18 in 0.03 seconds

```
PLAN_TABLE_OUTPUT
| Id  | Operation                           | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |               |     4 |   400 |     7   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                       |               |     4 |   400 |     7   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID       | EMP           |     1 |    54 |     2   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN                | EMP_PK        |     1 |       |     1   (0)| 00:00:01 |
|   4 |   TABLE ACCESS BY INDEX ROWID BATCHED| HEAP_ADDRESSES |     4 |   184 |     5   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN                 | SYS_C007067   |     4 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("EMP"."EMPNO"=42)
   5 - access("HEAP_ADDRESSES"."EMPNO"=42)
```

IOT

Using Explain Plan

```
SELECT *
  FROM emp ,
       iot_addresses
 WHERE emp.empno = iot_addresses.empno
 AND emp.empno   = 42;
```

Script Output | Query Result | SQL | All Rows Fetched: 4 in 0.01 seconds

	EMPNO	ENAME	HIREDATE	JOB	EMPNO_1	ADDR_TYPE	STREET	CITY	STATE	ZIP
1	42	I_ICOL1	07-JUL-22	SYS	42	HOME	123 main street	Washington	DC	20123
2	42	I_ICOL1	07-JUL-22	SYS	42	PREV	123 main street	Washington	DC	20123
3	42	I_ICOL1	07-JUL-22	SYS	42	SCHOOL	123 main street	Washington	DC	20123
4	42	I_ICOL1	07-JUL-22	SYS	42	WORK	123 main street	Washington	DC	20123

```
SELECT *
  FROM emp ,
       iot_addresses
 WHERE emp.empno = iot_addresses.empno
 AND emp.empno   = 42;
```

Script Output | Query Result | Explain Plan | SQL | 0.016 seconds

TION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			4	3
NESTED LOOPS			4	3
TABLE ACCESS	EMP	BY INDEX ROWID	1	2
INDEX	EMP_PK	UNIQUE SCAN	1	1
Access Predicates				
EMP.EMPNO=42				
INDEX	SYS_IOT_TOP_66702	RANGE SCAN	4	1
Access Predicates				
IOT_ADDRESSES.EMPNO=42				
Other XML				

Option 2

Welcome Page | VeraDB-port1521 | VeraDB_Lab3_Task_3.sql

SQL Worksheet | History

Worksheet | Query Builder

```
explain plan FOR
SELECT *
  FROM emp ,
       iot_addresses
 WHERE emp.empno = iot_addresses.empno
 AND emp.empno   = 42;

select * from table(dbms_xplan.display );
```

Script Output | Explain Plan | Query Result | SQL | All Rows Fetched: 17 in 0.034 seconds

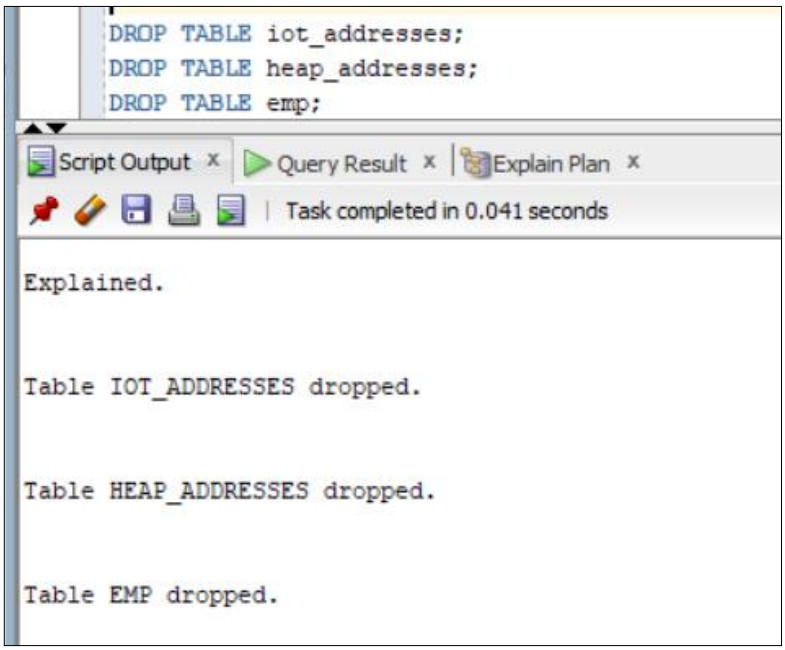
```
PLAN_TABLE_OUTPUT
3  ----------------------------------------------------------------------------------------------------
4  | Id  | Operation                    | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
5  ----------------------------------------------------------------------------------------------------
6  |   0 | SELECT STATEMENT             |                  |     4 |   400 |     3   (0)| 00:00:01 |
7  |   1 |  NESTED LOOPS                |                  |     4 |   400 |     3   (0)| 00:00:01 |
8  |   2 |   TABLE ACCESS BY INDEX ROWID| EMP              |     1 |    54 |     2   (0)| 00:00:01 |
9  |*  3 |    INDEX UNIQUE SCAN         | EMP_PK           |     1 |       |     1   (0)| 00:00:01 |
10 |*  4 |   INDEX RANGE SCAN           | SYS_IOT_TOP_66702 |     4 |   184 |     1   (0)| 00:00:01 |
11 ----------------------------------------------------------------------------------------------------
12
13 Predicate Information (identified by operation id):
14 ---------------------------------------------------
15
16    3 - access("EMP"."EMPNO"=42)
17    4 - access("IOT_ADDRESSES"."EMPNO"=42)
```

Summary: Cardinality is the estimated number of rows the step will return. Cost is the estimated amount of work the plan will do. As we can see, IOT has an advantage over Heap table in terms of cost.

Advantage of IOT

- The IOT combines a table and index in one, saving you the effort of creating an index Queries using the fields of this index result in fewer block accesses
- Results in significant performance gains, since additional access to table blocks is no longer necessary
- Because rows are stored in primary key order, a significant amount of additional storage space savings can be obtained through the use of key compression.

Step 7: Drop tables



Code: Task 3

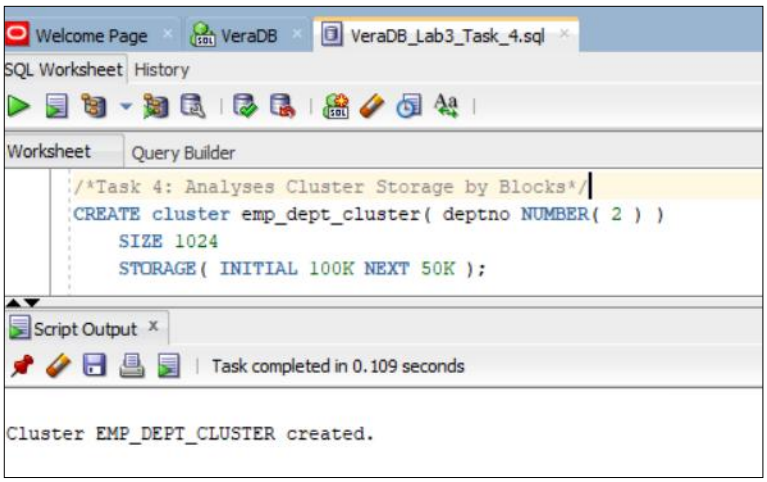
```
/*Task 3: Compare performance of using IOT tables*/

CREATE TABLE emp AS
SELECT
  object_id empno,
  object_name ename,
  created hiredate,
  owner job
FROM
  all_objects;
COMMIT;

ALTER TABLE emp add constraint emp_pk primary key (empno);

SELECT * FROM emp;

/*Calculate Statistic*/

begin
  dbms_stats.gather_table_stats( user, 'EMP', cascade=>true );
end;
COMMIT;

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)
  );
COMMIT;

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;

COMMIT;

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;

EXEC dbms_stats.gather_table_stats( user, 'HEAP_ADDRESSES');
/*Begin dbms_stats.gather_table_stats( user, 'HEAP_ADDRESSES');  END;*/
EXEC dbms_stats.gather_table_stats( user, 'IOT_ADDRESSES');
/*Begin dbms_stats.gather_table_stats( user, 'IOT_ADDRESSES');  END;*/

/*Option1*/
SELECT *
  FROM emp ,
      heap_addresses
 WHERE emp.empno = heap_addresses.empno
 AND emp.empno   = 42;

 SELECT *
  FROM emp ,
      iot_addresses
 WHERE emp.empno = iot_addresses.empno
 AND emp.empno   = 42;

/*Option2*/
explain plan for
select *
from emp, heap_addresses
where emp.empno = heap_addresses.empno
and emp.empno = 42;

select * from table(dbms_xplan.display );


 explain plan FOR
 SELECT *
  FROM emp ,
      iot_addresses
 WHERE emp.empno = iot_addresses.empno
 AND emp.empno   = 42;

 select * from table(dbms_xplan.display );

DROP TABLE iot_addresses;
DROP TABLE heap_addresses;
DROP TABLE emp;
select segment_name, segment_type from user_segments;
PURGE RECYCLEBIN;
select segment_name, segment_type from user_segments;
```

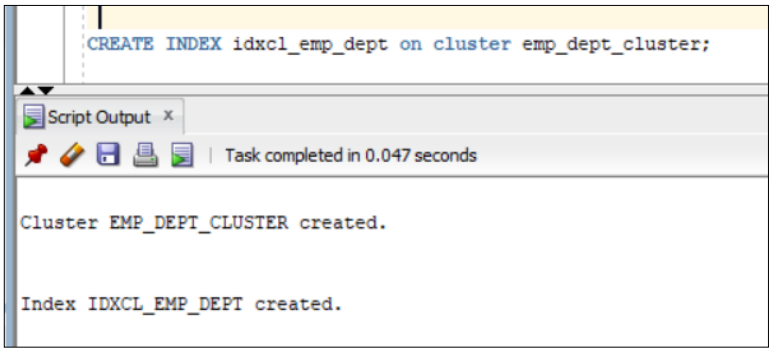
4. Index Clustered Tables

Task 4: Analyses Cluster Storage by Blocks

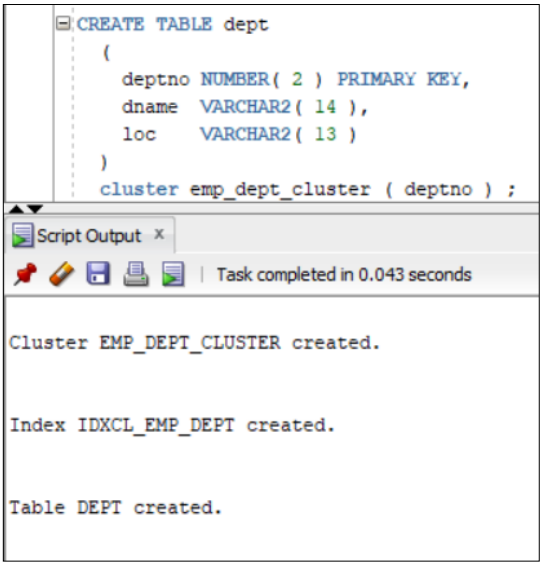
Step 1: Create cluster



Step 2: Create index



Step 3: Create tables



```
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 ) ;
```

Script Output | Task completed in 0.05 seconds

```
Cluster EMP_DEPT_CLUSTER created.


Index IDXCL_EMP_DEPT created.


Table DEPT created.


Table EMP created.
```

<u>Step 4:</u> Insert values into tables

```
INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

COMMIT;
```

Script Output | Task completed in 0.042 seconds

```
Table EMP created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Commit complete.
```

```
INSERT INTO EMP VALUES
(7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO EMP VALUES
(7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO EMP VALUES
(7521,'WARD','SALESMAN',7698,to_date('22-2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO EMP VALUES
(7566,'JONES','MANAGER',7839,to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO EMP VALUES
(7654,'MARTIN','SALESMAN',7698,to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO EMP VALUES
(7698,'BLAKE','MANAGER',7839,to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO EMP VALUES
(7782,'CLARK','MANAGER',7839,to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO EMP VALUES
(7788,'SCOTT','ANALYST',7566,to_date('13-JUL-87','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO EMP VALUES
(7839,'KING','PRESIDENT',NULL,to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO EMP VALUES
(7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO EMP VALUES
(7876,'ADAMS','CLERK',7788,to_date('13-JUL-87', 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO EMP VALUES
(7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);
INSERT INTO EMP VALUES
(7902,'FORD','ANALYST',7566,to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO EMP VALUES
(7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);

COMMIT;
```

Script Output | Task completed in 0.028 seconds

```
Commit complete.
```

Step 5: Data results

```
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;
```

Script Output | Query Result | SQL | All Rows Fetched: 14 in 0.634 seconds

	DEPT_BLK	EMP_BLK	FLAG	DEPTNO
1	37	37	(null)	10
2	37	37	(null)	10
3	37	37	(null)	10
4	37	37	(null)	20
5	37	37	(null)	20
6	37	37	(null)	20
7	37	37	(null)	20
8	37	37	(null)	20
9	37	37	(null)	30
10	37	37	(null)	30
11	37	37	(null)	30
12	37	37	(null)	30
13	37	37	(null)	30
14	37	37	(null)	30

Summary: As we see, all data from the EMP and DEPT tables, selected using the cluster index, fall into a single block. Since the clustered tables are stored in a single database block, the time to perform I/O operations is noticeably reduced. Clustered tables reduce the number of blocks that Oracle must cache. On the downside, unless you can calculate your SIZE parameter setting correctly, clusters may be inefficient with their space utilization and can tend to slow down DML-heavy operations.

Step 6: Drop tables

Code: Task 4

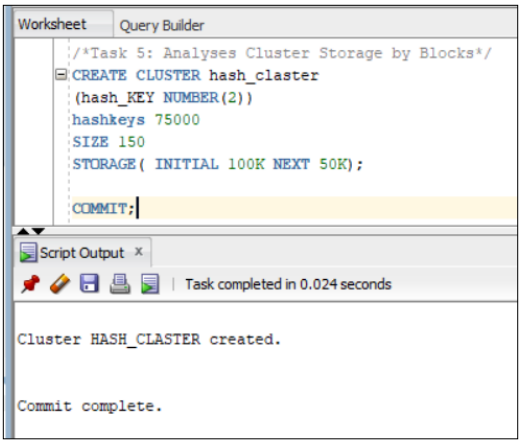
```
/*Task 4: Analyses Cluster Storage by Blocks*/
CREATE cluster emp_dept_cluster( deptno NUMBER( 2 ) )
   SIZE 1024
   STORAGE( INITIAL 100K NEXT 50K );

CREATE INDEX idxcl_emp_dept on cluster emp_dept_cluster;

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 VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

COMMIT;
```

```
INSERT INTO EMP VALUES
(7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO EMP VALUES
(7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO EMP VALUES
(7521,'WARD','SALESMAN',7698,to_date('22-2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO EMP VALUES
(7566,'JONES','MANAGER',7839,to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO EMP VALUES
(7654,'MARTIN','SALESMAN',7698,to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO EMP VALUES
(7698,'BLAKE','MANAGER',7839,to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO EMP VALUES
(7782,'CLARK','MANAGER',7839,to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO EMP VALUES
(7788,'SCOTT','ANALYST',7566,to_date('13-JUL-87','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO EMP VALUES
(7839,'KING','PRESIDENT',NULL,to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO EMP VALUES
(7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO EMP VALUES
(7876,'ADAMS','CLERK',7788,to_date('13-JUL-87', 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO EMP VALUES
(7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);
INSERT INTO EMP VALUES
(7902,'FORD','ANALYST',7566,to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO EMP VALUES
(7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);

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;

DROP TABLE emp;
DROP TABLE dept;
DROP cluster emp_dept_cluster;
```

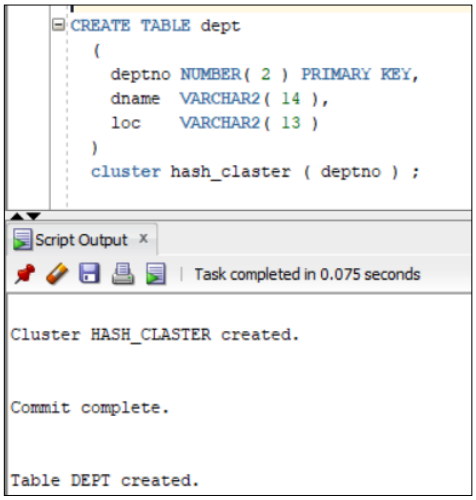
5. Hash Clustered Tables

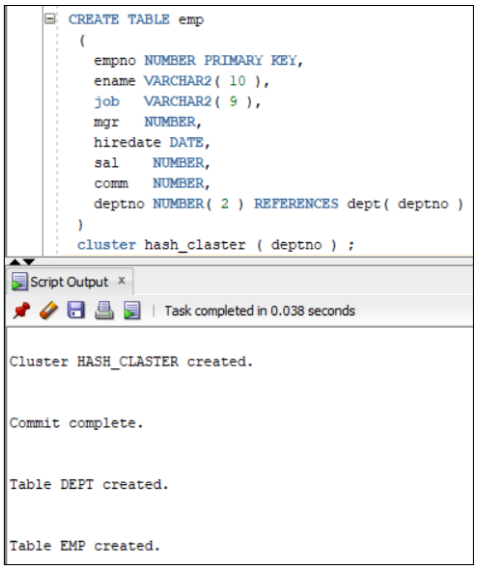
5.1. Task 5: Analyses Cluster Storage by Blocks

Step 1: Create Hush Cluster



Step 2: Create tables





Step 3: Insert values into tables

```
INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

COMMIT;
```

Script Output | Task completed in 0.028 seconds

```
Table DEPT created.


Table EMP created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Commit complete.
```

Worksheet | Query Builder

```
INSERT INTO EMP VALUES
(7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO EMP VALUES
(7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO EMP VALUES
(7521,'WARD','SALESMAN',7698,to_date('22-2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO EMP VALUES
(7566,'JONES','MANAGER',7839,to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO EMP VALUES
(7654,'MARTIN','SALESMAN',7698,to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO EMP VALUES
(7698,'BLAKE','MANAGER',7839,to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO EMP VALUES
(7782,'CLARK','MANAGER',7839,to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO EMP VALUES
(7788,'SCOTT','ANALYST',7566,to_date('13-JUL-87','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO EMP VALUES
(7839,'KING','PRESIDENT',NULL,to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO EMP VALUES
(7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO EMP VALUES
(7876,'ADAMS','CLERK',7788,to_date('13-JUL-87', 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO EMP VALUES
(7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);
INSERT INTO EMP VALUES
(7902,'FORD','ANALYST',7566,to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO EMP VALUES
(7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);

COMMIT;
```

Script Output | Task completed in 0.038 seconds

```
Commit complete.
```

Step 4: Data results

```
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;
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.071 seconds

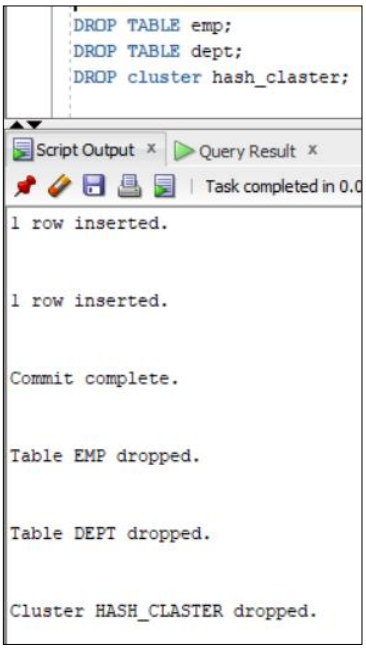
	DEPT_BLK	EMP_BLK	FLAG	DEPTNO
1	73	73	(null)	10
2	73	73	(null)	10
3	73	73	(null)	10
4	157	157	(null)	20
5	157	157	(null)	20
6	157	157	(null)	20
7	157	157	(null)	20
8	157	157	(null)	20
9	301	301	(null)	30
10	301	301	(null)	30
11	301	301	(null)	30
12	301	301	(null)	30
13	301	301	(null)	30
14	301	301	(null)	30

Summary: As a result of creating Hash Clustered Table, we noticed that the data from the EMP and DEPT tables are broken into blocks using a Hash function. Hash clustered tables are very similar in concept to the index clustered tables described earlier 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.

The important things to understand about hash clusters are as follows:

- The hash cluster is allocated right from the beginning
- The number of HASHKEYs in a hash cluster is a fixed size
- Range scanning on the cluster key is not available.

Step 5: Drop tables



Code: Task 5

```
/*Task 5: Analyses Cluster Storage by Blocks*/
CREATE CLUSTER hash_claster
(hash_KEY NUMBER(2))
hashkeys 75000
SIZE 150
STORAGE( INITIAL 100K NEXT 50K);

COMMIT;

CREATE TABLE dept
 (
   deptno NUMBER( 2 ) PRIMARY KEY,
   dname  VARCHAR2( 14 ),
   loc    VARCHAR2( 13 )
 )
 cluster hash_claster ( 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 hash_claster ( deptno ) ;

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');
INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');
INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');
```

```
COMMIT;

INSERT INTO EMP VALUES
(7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO EMP VALUES
(7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO EMP VALUES
(7521,'WARD','SALESMAN',7698,to_date('22-2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO EMP VALUES
(7566,'JONES','MANAGER',7839,to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO EMP VALUES
(7654,'MARTIN','SALESMAN',7698,to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO EMP VALUES
(7698,'BLAKE','MANAGER',7839,to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO EMP VALUES
(7782,'CLARK','MANAGER',7839,to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO EMP VALUES
(7788,'SCOTT','ANALYST',7566,to_date('13-JUL-87','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO EMP VALUES
(7839,'KING','PRESIDENT',NULL,to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO EMP VALUES
(7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO EMP VALUES
(7876,'ADAMS','CLERK',7788,to_date('13-JUL-87', 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO EMP VALUES
(7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);
INSERT INTO EMP VALUES
(7902,'FORD','ANALYST',7566,to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO EMP VALUES
(7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);

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;

DROP TABLE emp;
DROP TABLE dept;
DROP cluster hash_claster;
```