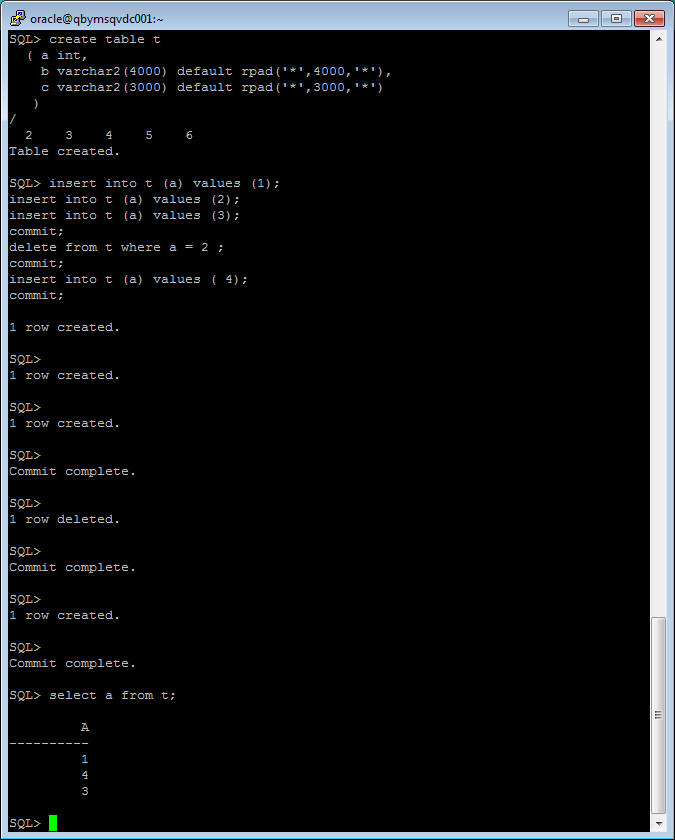
**Task 3 – Aliaksandr\_Kudzelka**

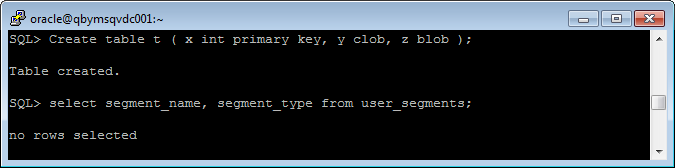
**Ex1 – Heap**

Steps 1-3

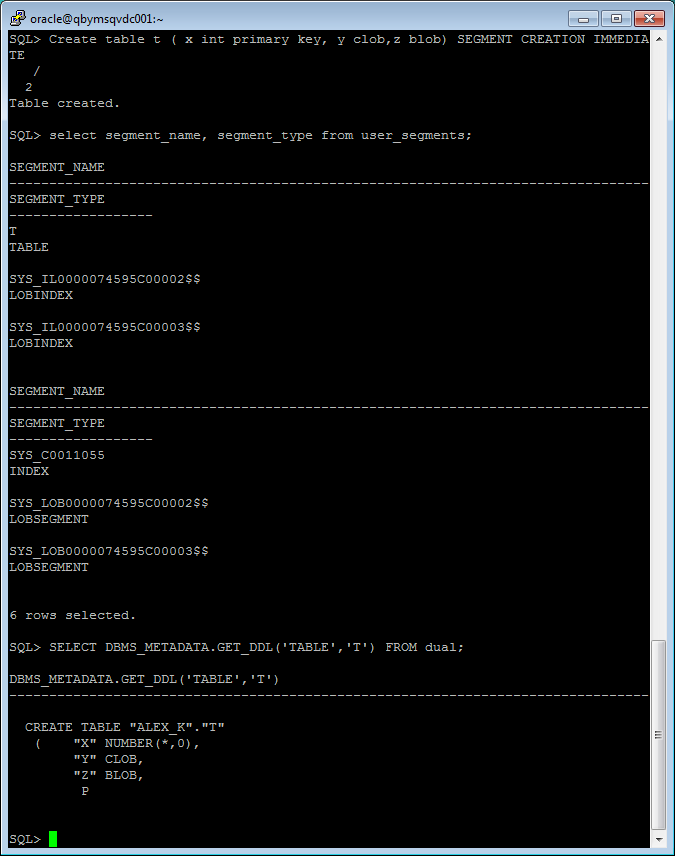


**Ex2 – Low level of Data Abstraction**

Steps 1-2

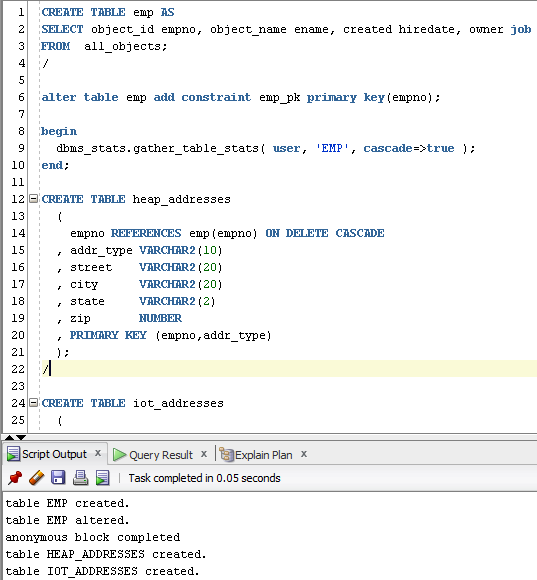


Step 3-5

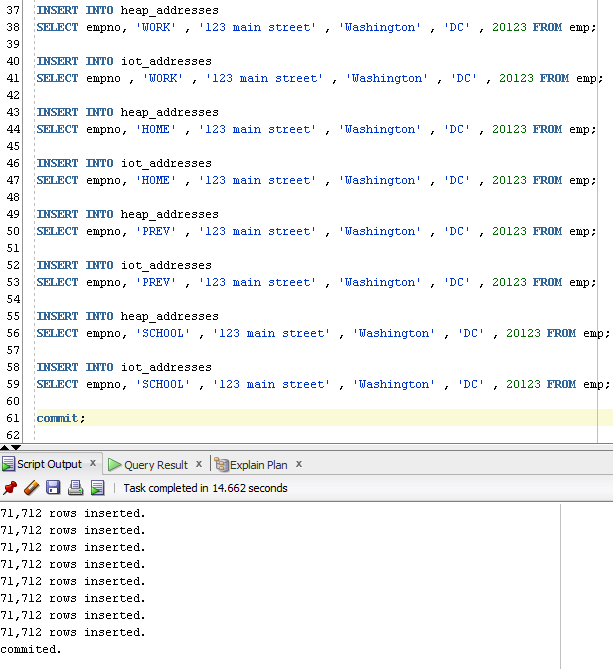


**Ex3 – Comparing Performance of using IOT Tables**

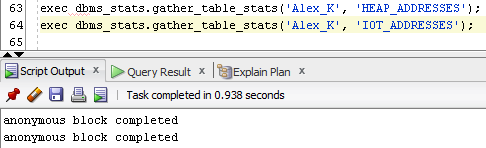
Steps 1-3



Step 4 - Inserting

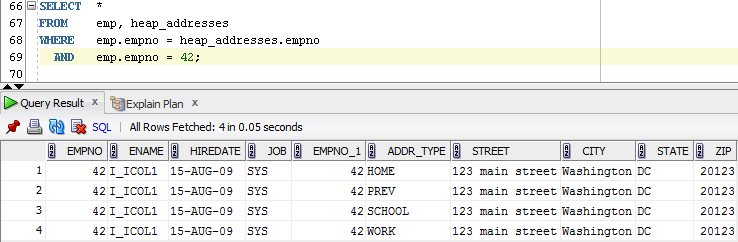


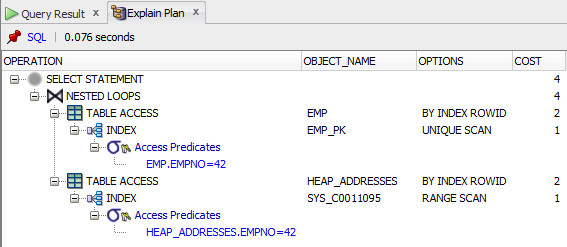
Step 5 – Calculating statistics



Step 6 – Inserting

(1)

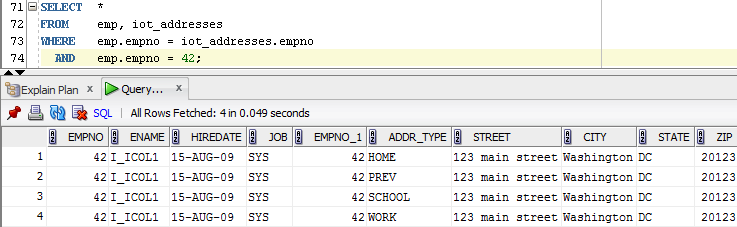


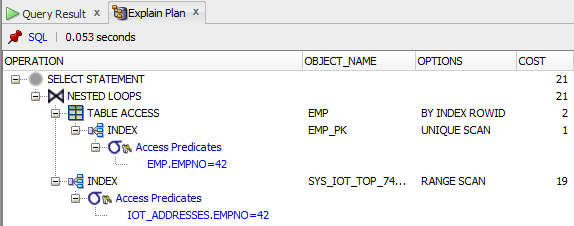


IO table stores rows in a B-tree index structure that is logically sorted in primary key (empno) order.

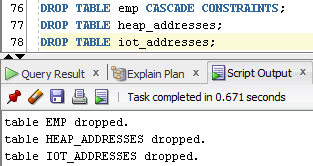
Unlike normal primary key indexes (heap organized table), which store only the columns included in it definition, IOT indexes store all the columns of the table.

(2)





Step 7 – Dropping tables



**Ex4-5 – Analysing Cluster Storage by Blocks**

**Index Clustered Tables** VS **HASH Clustered Tables**

Steps 1-2

|  |  |
| --- | --- |
|  |  |

**Index Clustered Tables** deliver 2 advantages:

1. The value for the cluster key is only stored once, and this reduces the amount of space required for storage.
2. A query can retrieve all the related data with usually no more than 2 logical reads: one *to get the cluster key* and another *to retrieve a data block that contains only the relevant data*.

|  |  |
| --- | --- |
|  |  |

Step 3-4 – Creating Tables & Inserting

|  |  |
| --- | --- |
|  |  |

Step 5 - Select

As we can see, data from **Index Clustered Table** are stored on the same block, and the cluster key value is the value of the cluster key columns for a particular row.

To use hashing, I created the hash cluster and loaded tables into it. The database physically stores the rows of a table in a hash cluster and retrieves them according to the results of a **hash function**.