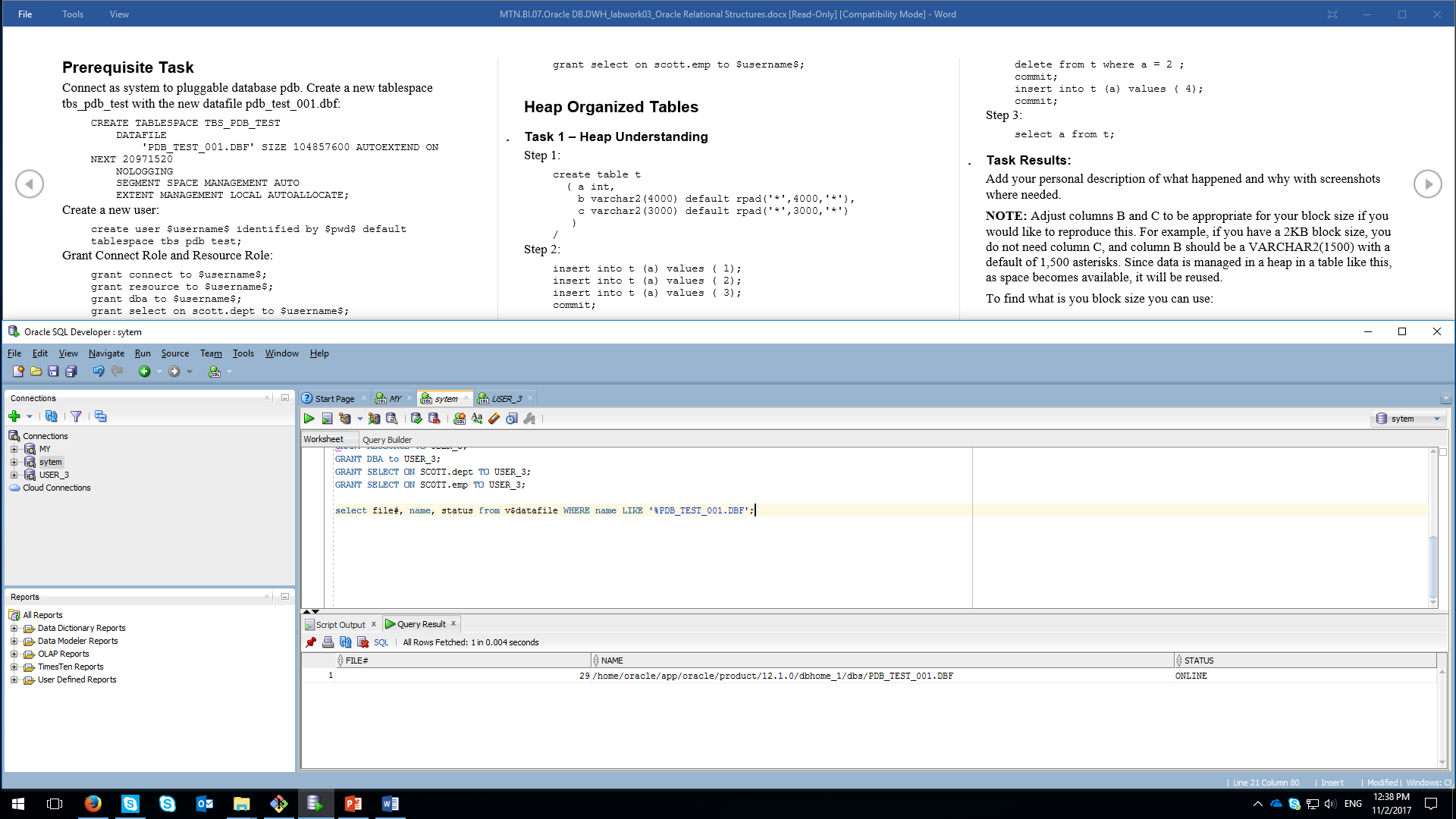
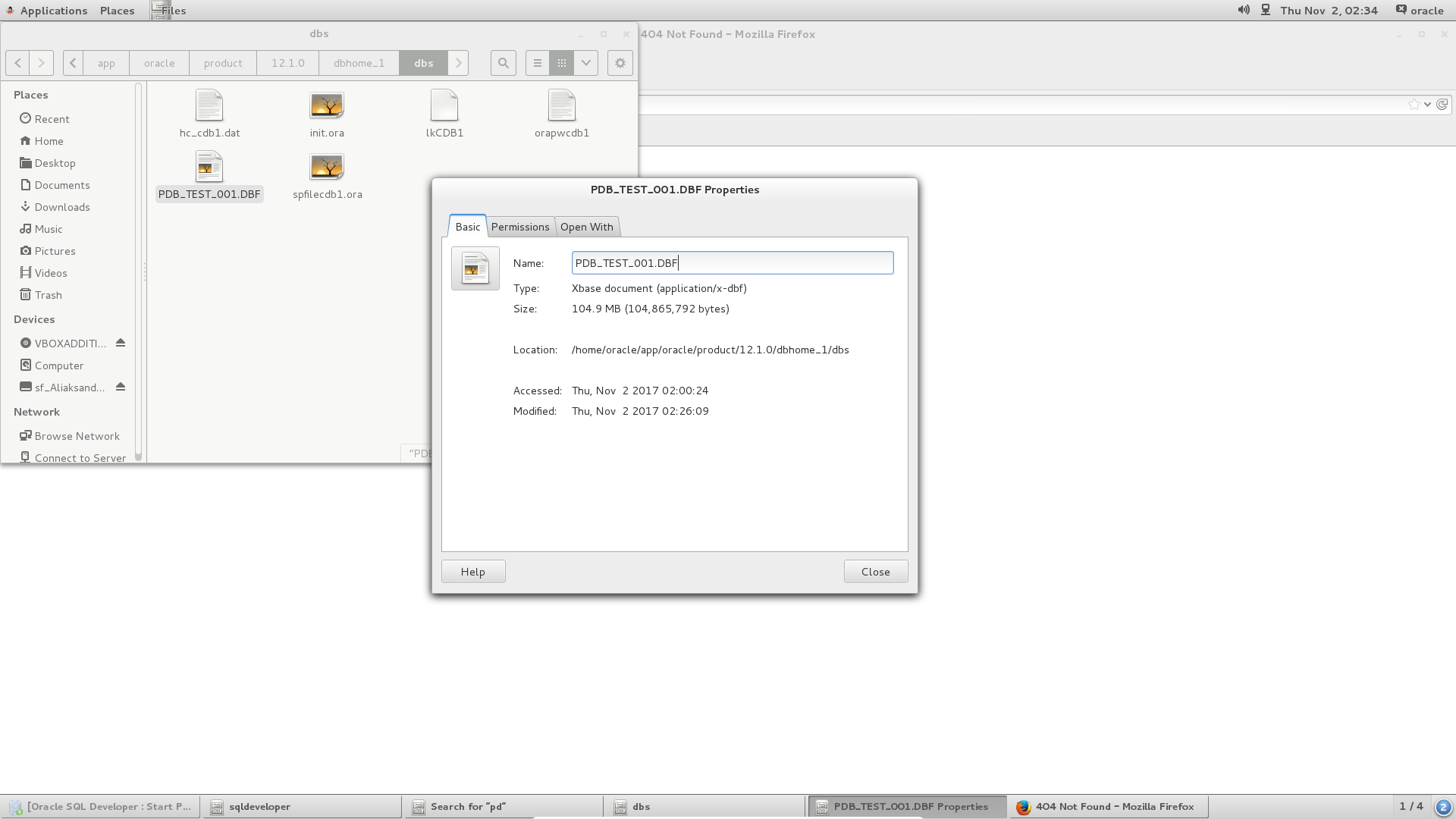
Task 1

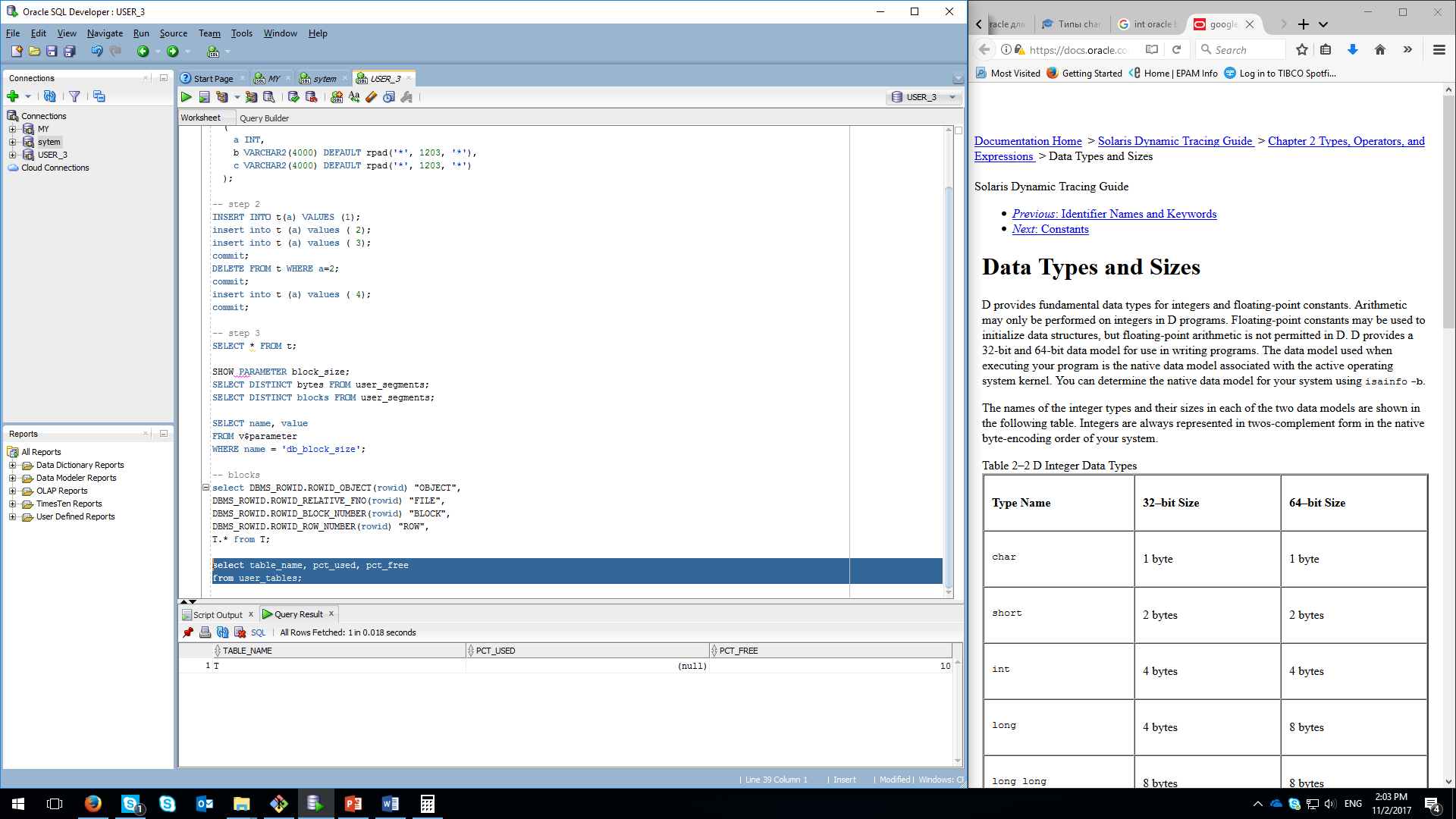
In 1st task, we’ve created default tablespace as a database file that was found on VM





--

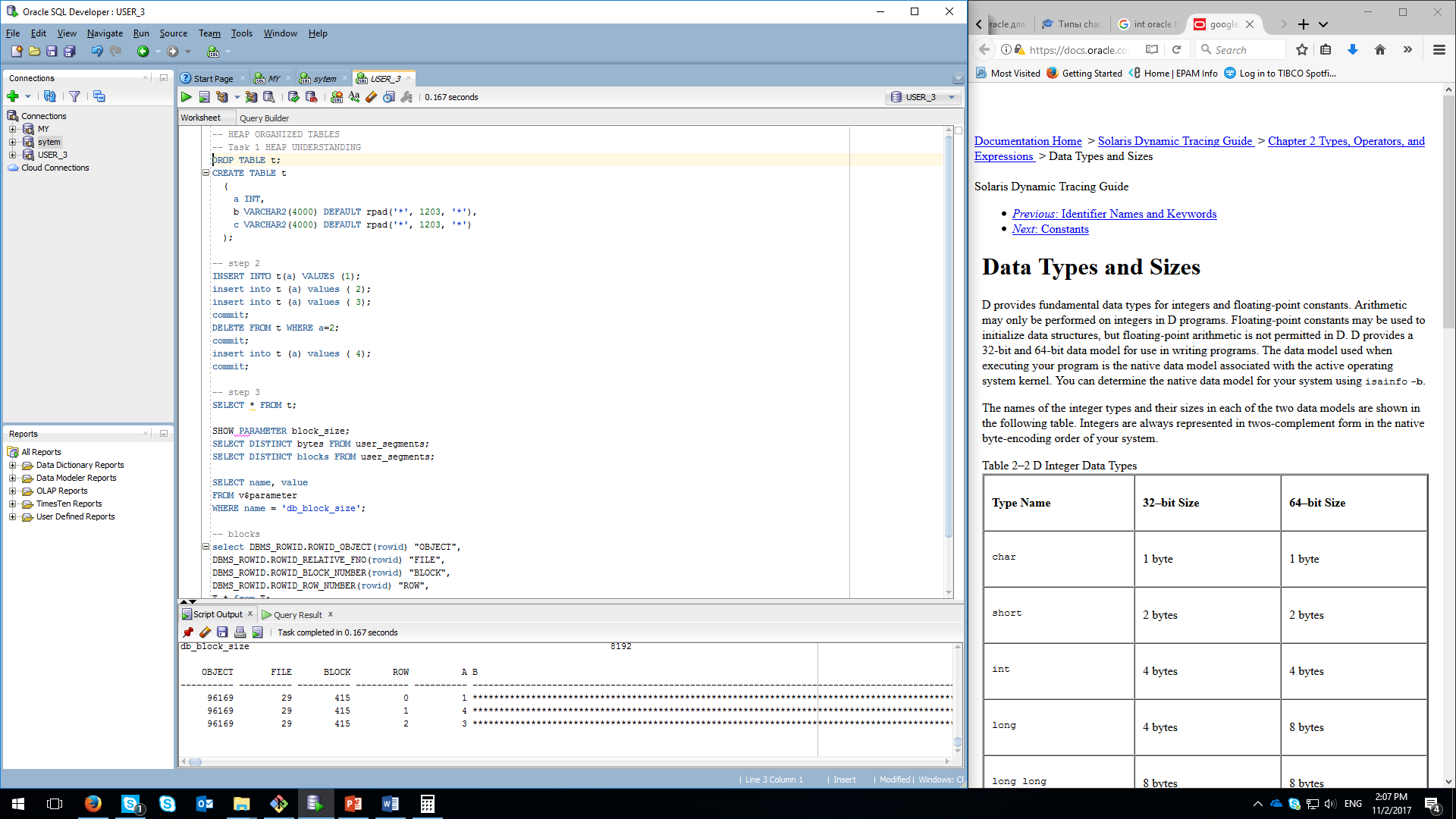
So as to make values be appropriate for one block size, changes to filed types should be made. In our example, we have 8 KBs with PCT free 10%. So it means that we have 90% free to use.



So we have around of 7 KBs filled with 3 columns:

* a (1-4 B);
* b (4000 B);
* c (3000 B).

When our default values are less than ‘1204’, Oracle moves all rows in the same block:

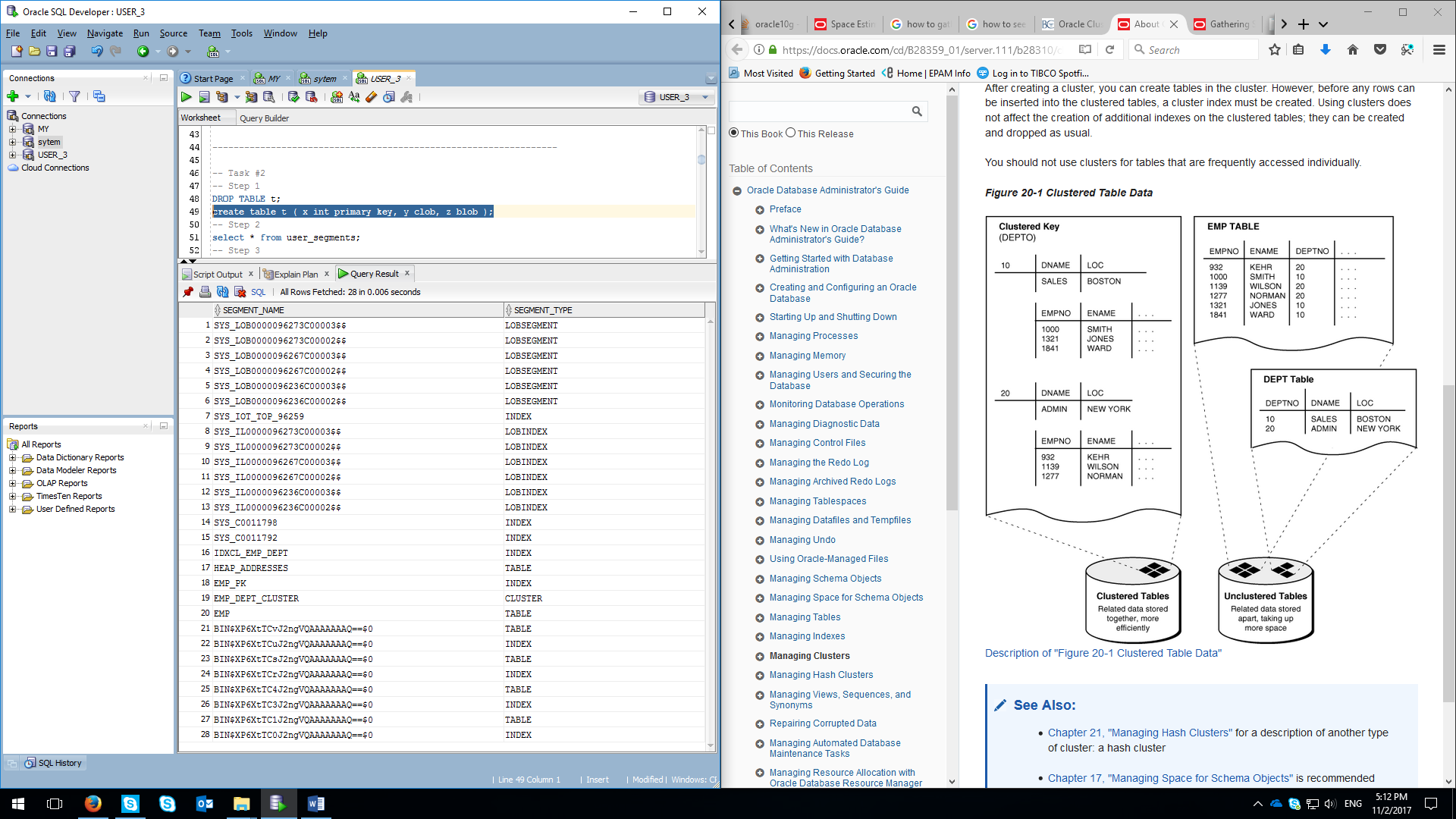


So we have to make our db columns be no more than 1203.

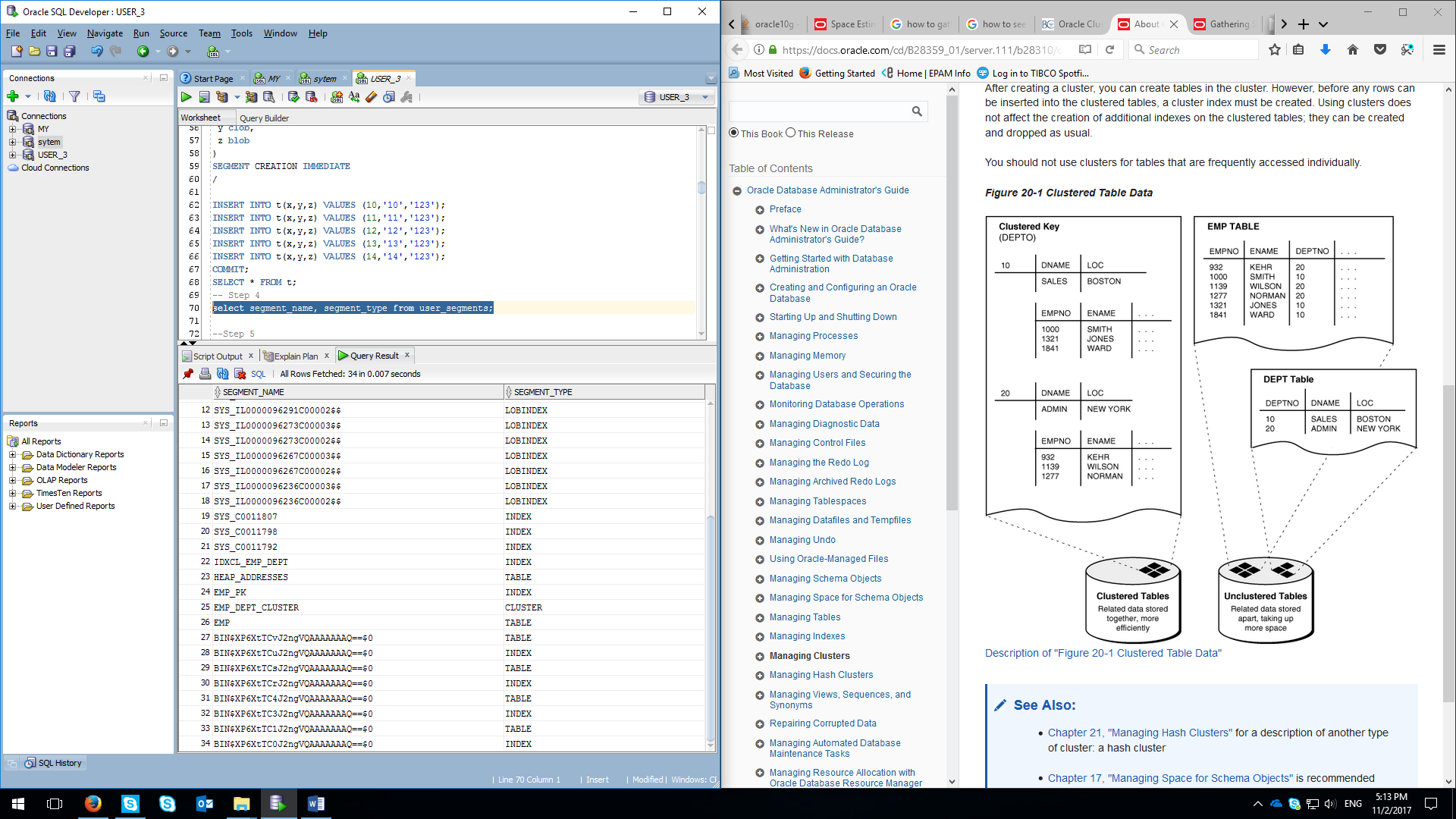
**Task 2 -** Understanding Heap Table Segments

Oracle do the segmentation automatically much better that it is made manually due to the gain in free space. By creating table with “segment immediate creation”, we reserve the place but there is no confidence, that we gonna use it by inserting values into it. So it is better not to reserve space in case we wouldn’t use it.

Automatically created SEGMENT:



Manually created SEGMENT. Here we can see additional rows with segment type like LOBINDEX, TABLE, INDEX.

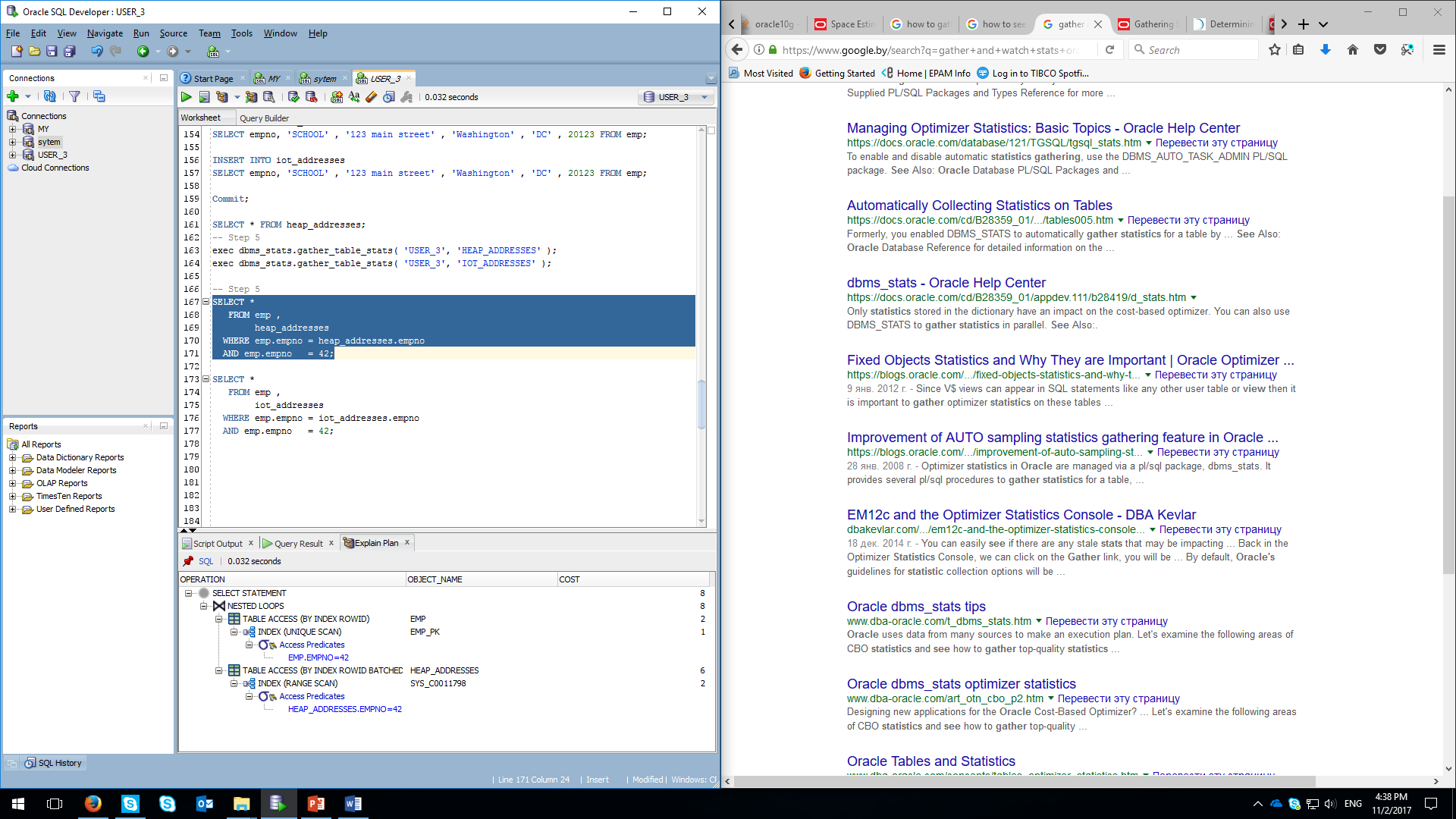


\* Additional “training” records can be seen on the screenshots because of the rebuilding the exercise

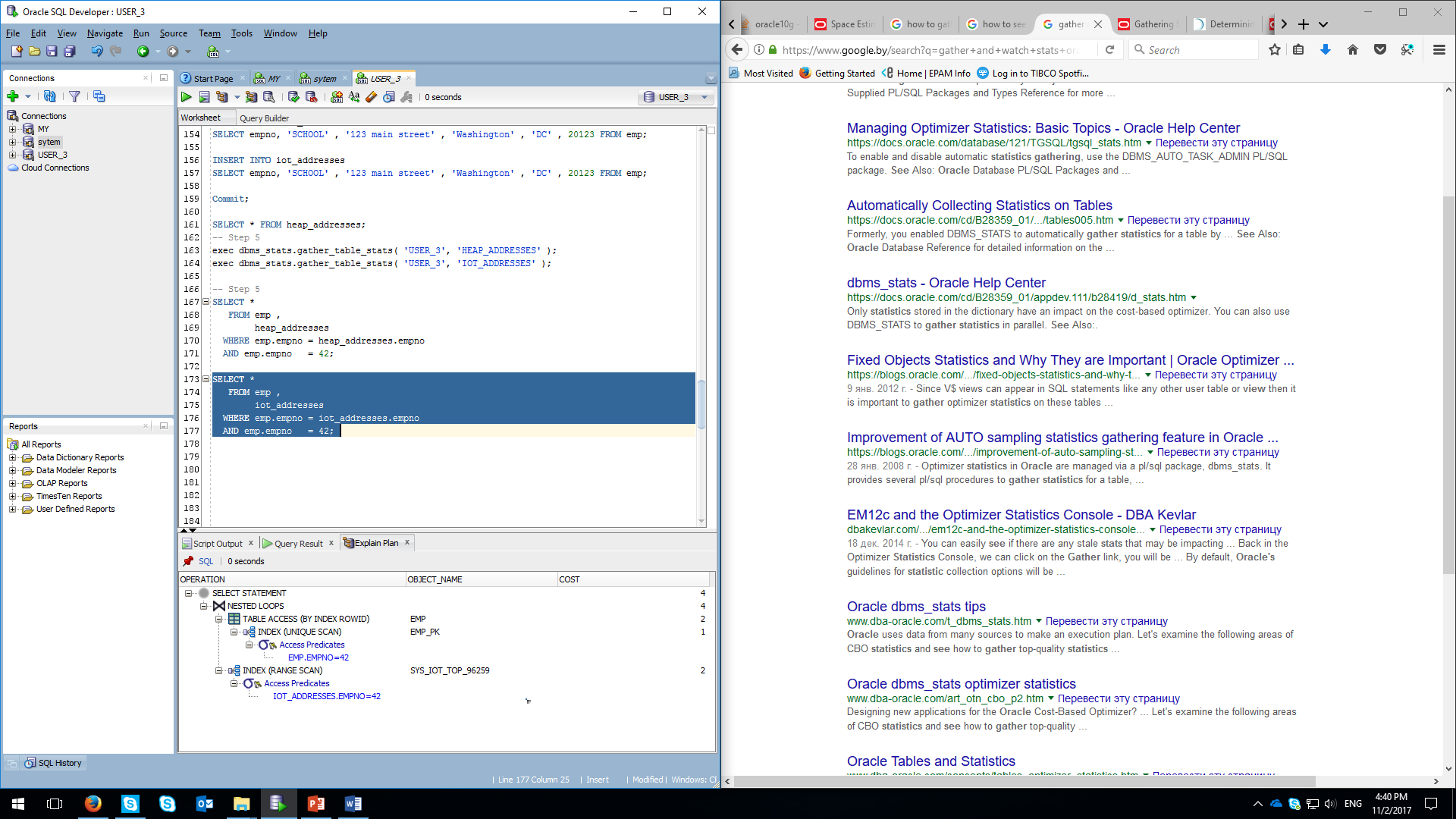
**Task 3 -** Index Organized Tables

Screenshots below can show the result of executing queries from 2 tables. As we can see from the screenshot from execution plan, **heap** organizedhas much less performance comparing to index-organized table.

Heap-organized table:

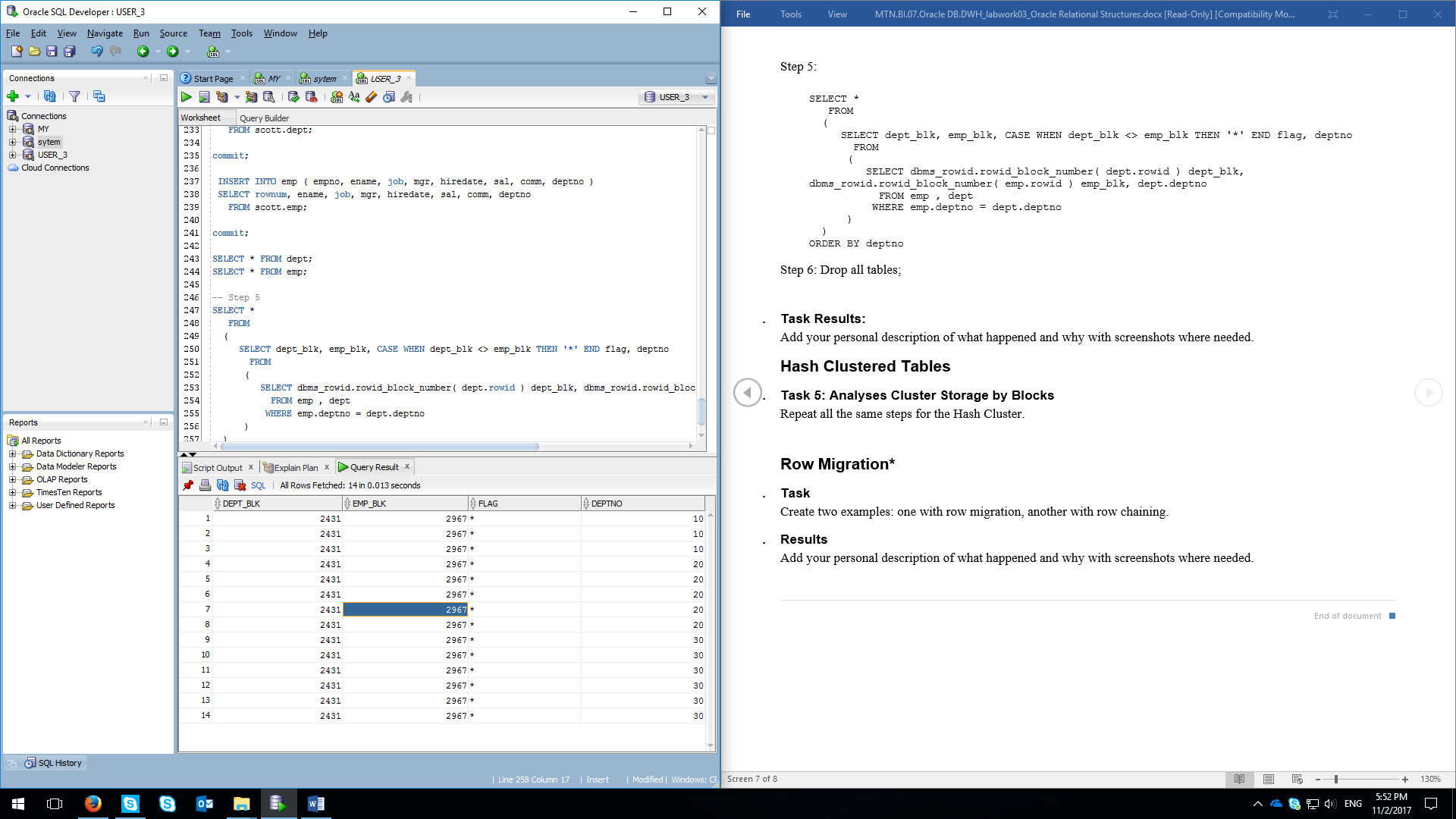


Index-organized table:

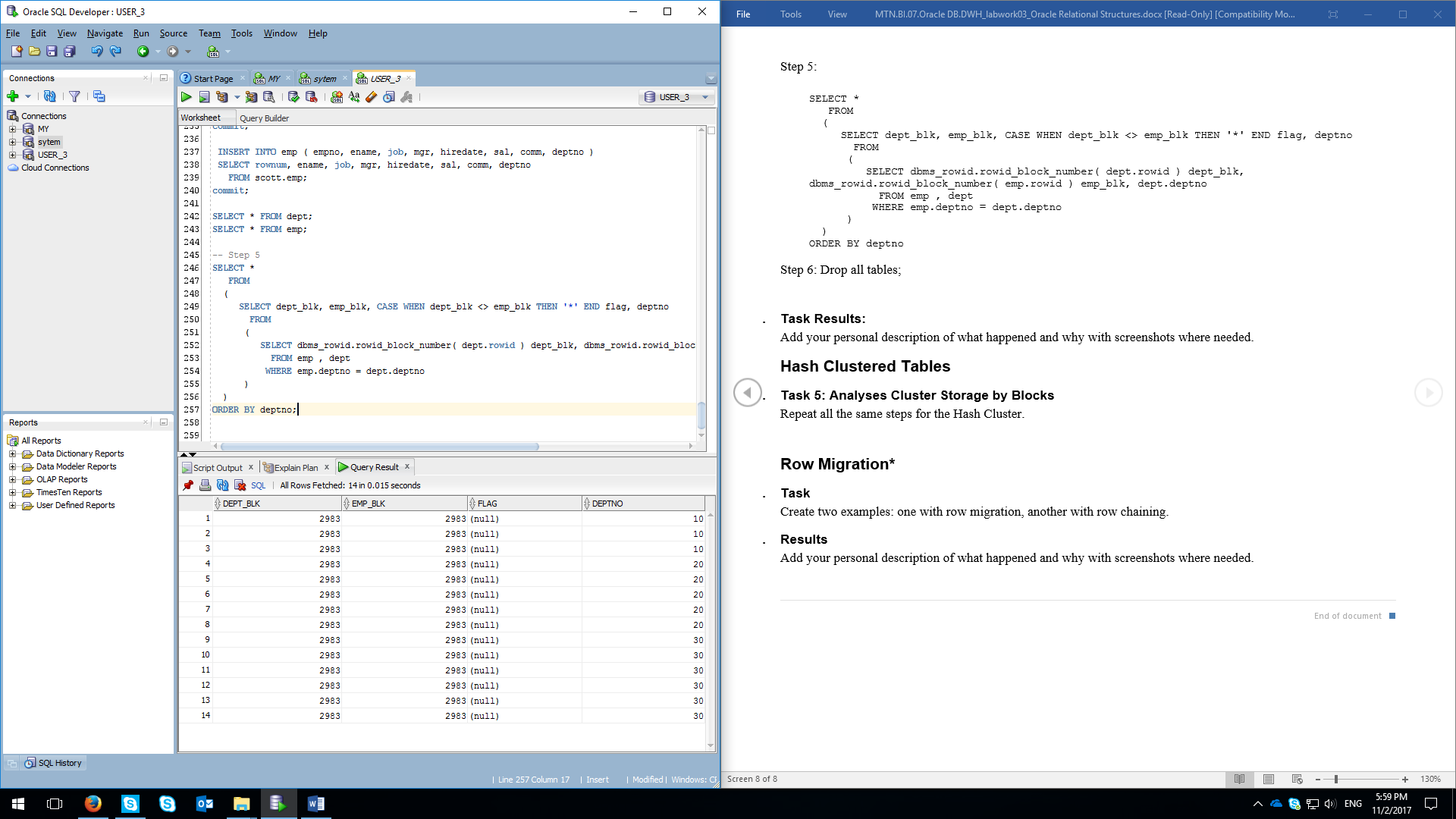


**Task 4 -** Cluster Storage by Blocks

4th task help us to make a cluster. Cluster helps to organize more than 1 table in one block. Screenshot below shows the result of creating table without cluster, where different datablocks are used.

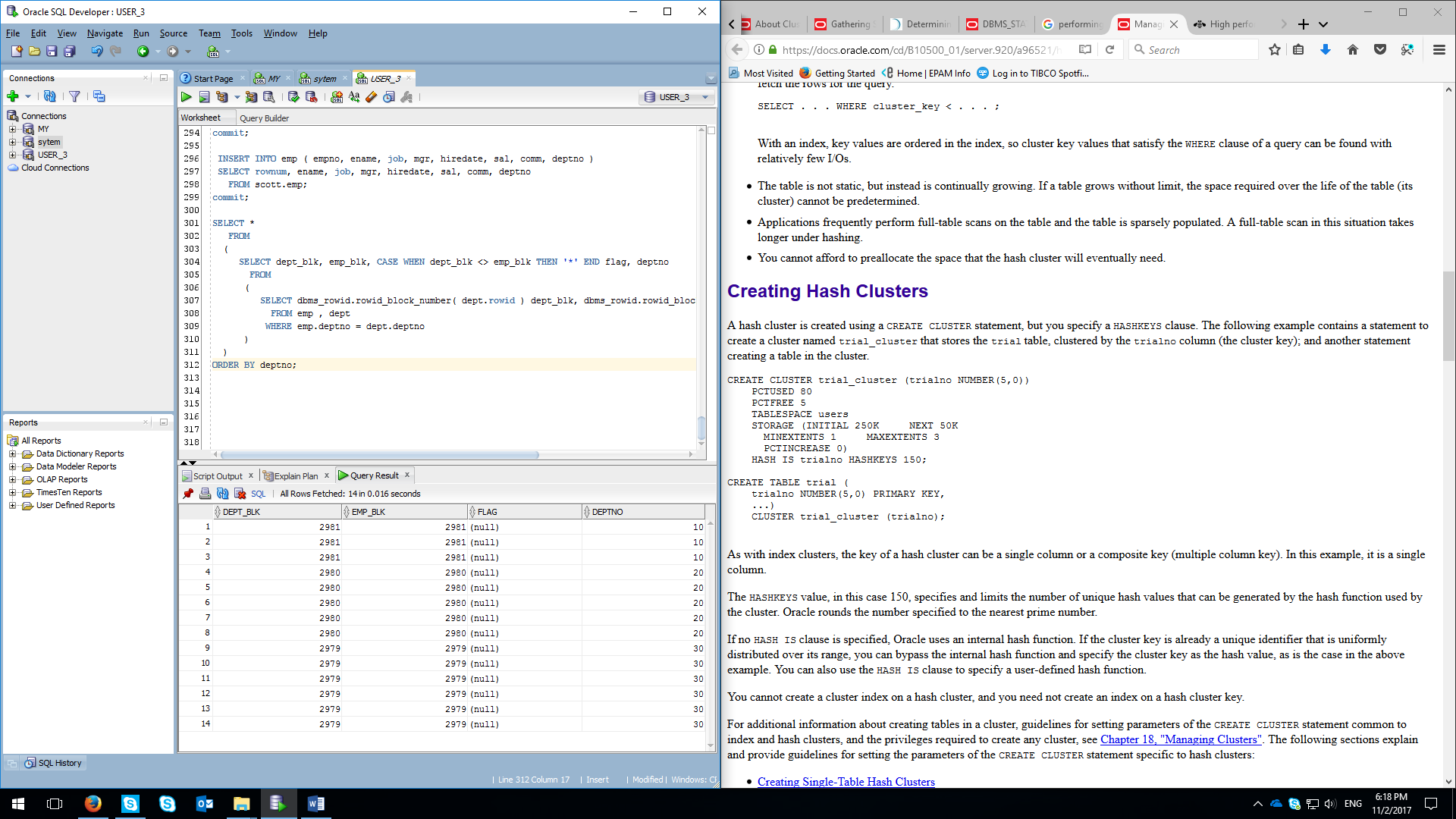


And screenshot below shows us the result of cluster storage organization, where all records from 2 tables (included in cluster) are moved to the same block:



**Task 5 – Hash clustering**

Hash clustering makes data to be stored in the data block according to the chosen field – deptno, as shown in our example.



**Task 6 – ROW migration**

**ROW migration** starts with creating table with fixed filed values (using CHAR instead of asterisks as an example):

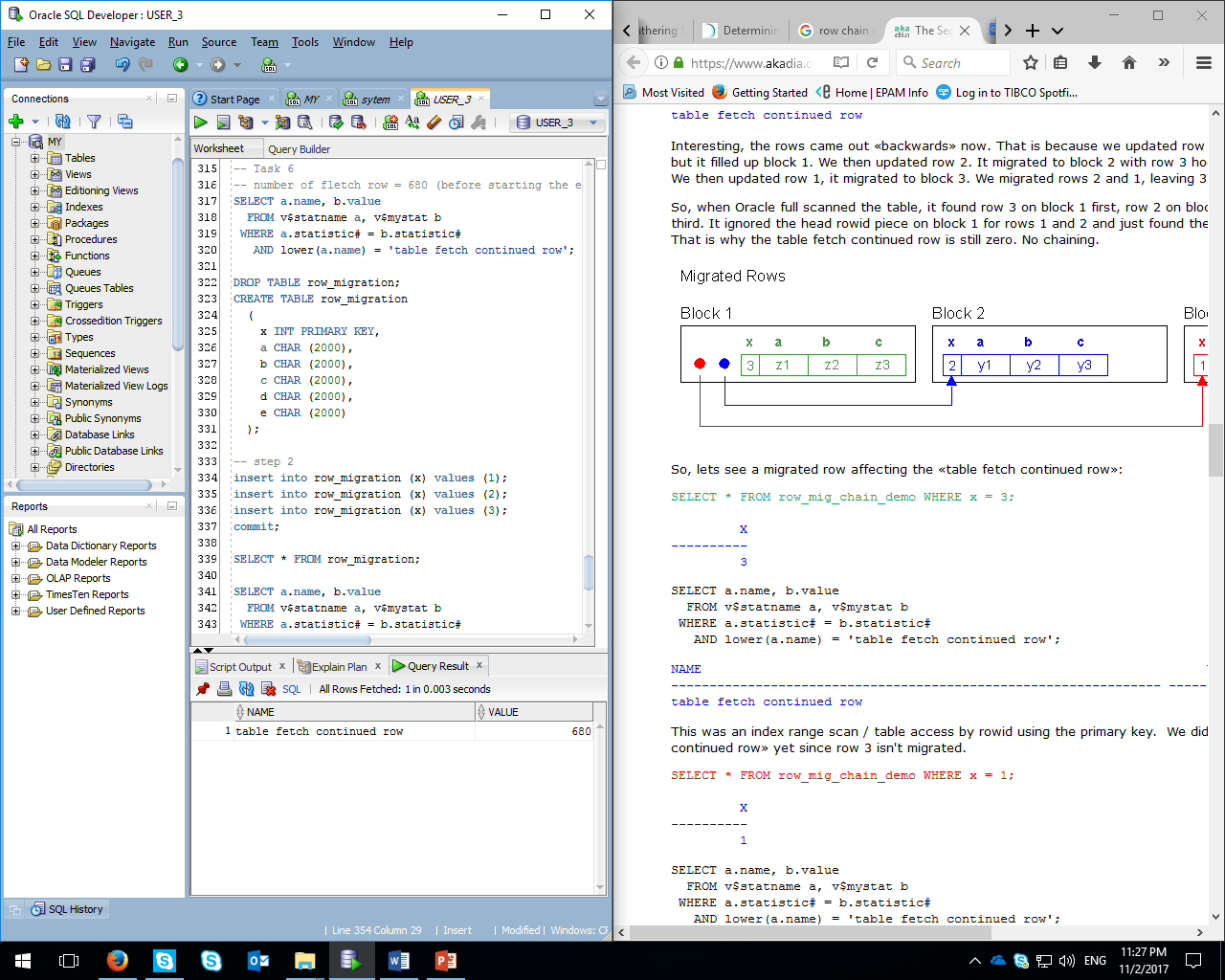
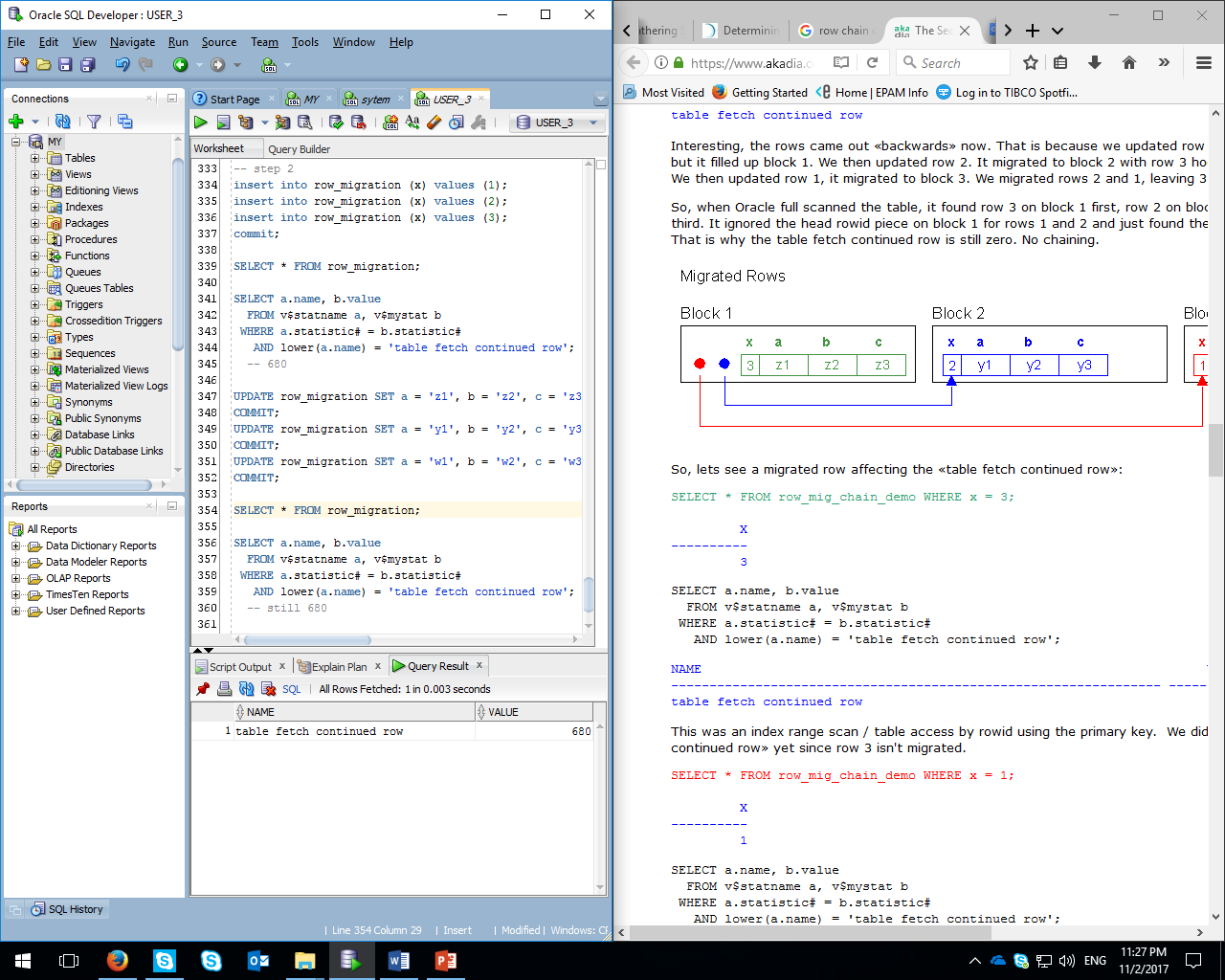
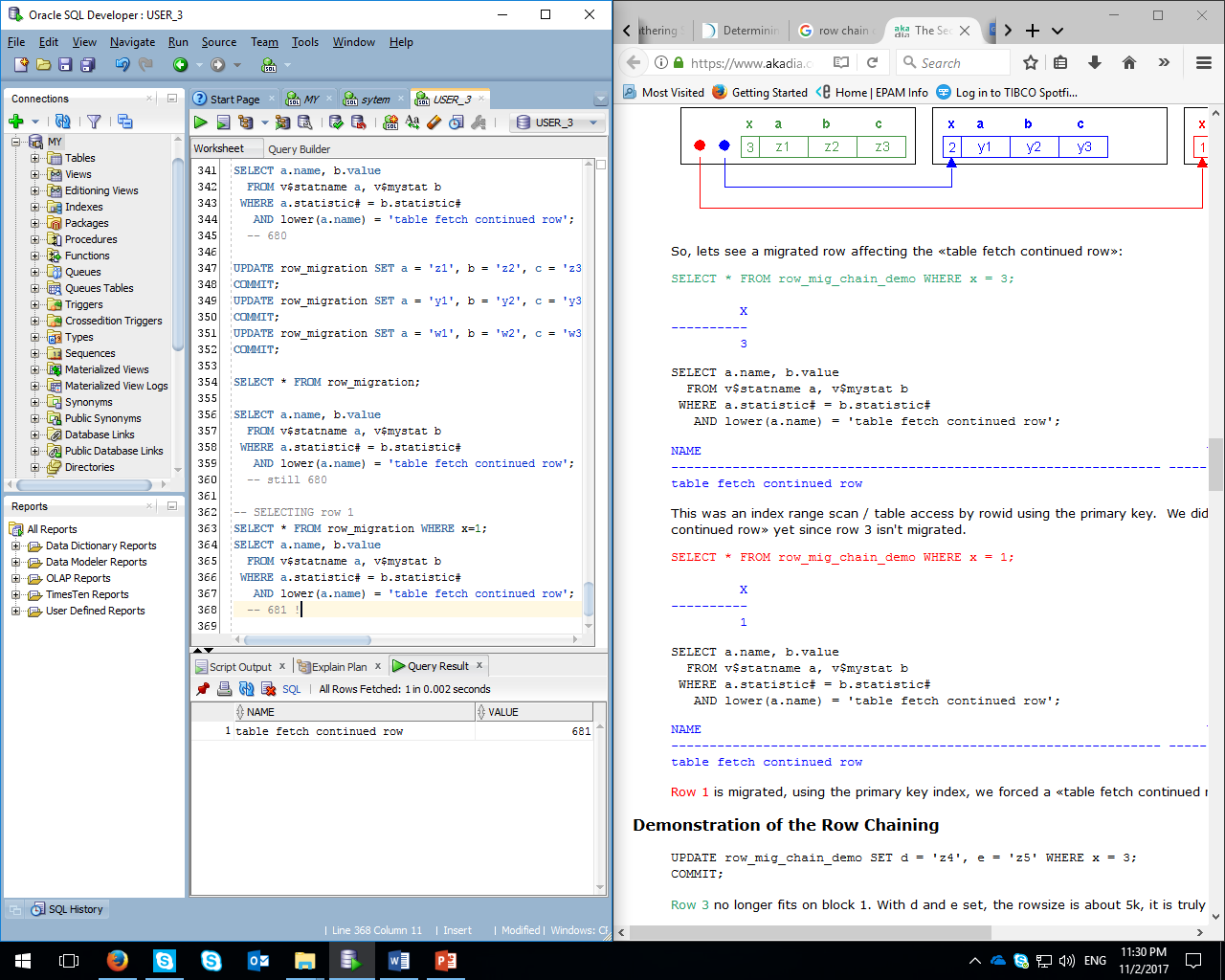


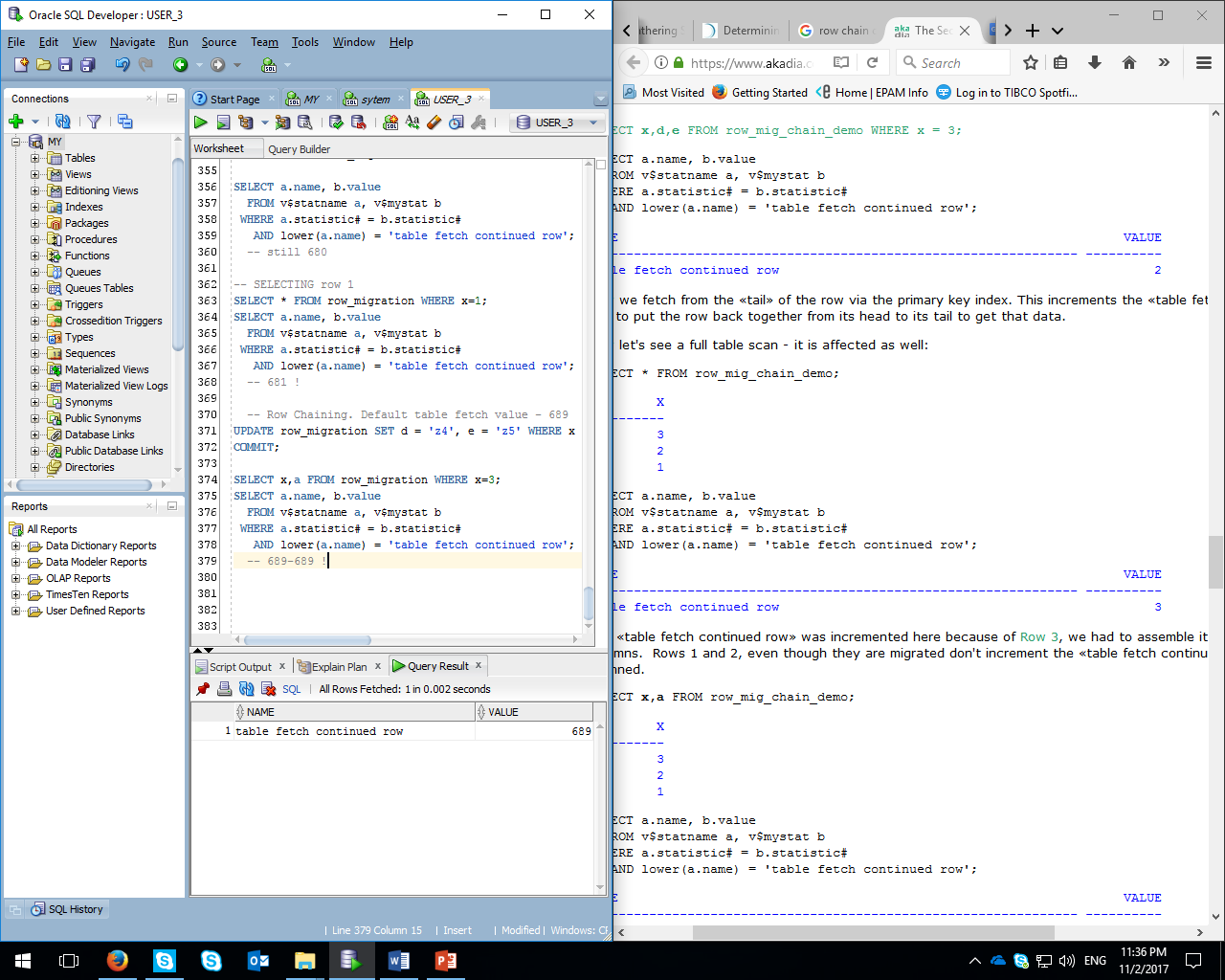
Table fletch continued row is still 680 (default in my example after performing previous tasks) and hadn’t been increased after select:



Row 1 is migrated using the PK index and see the results in fetch continued row – 681 (+1):

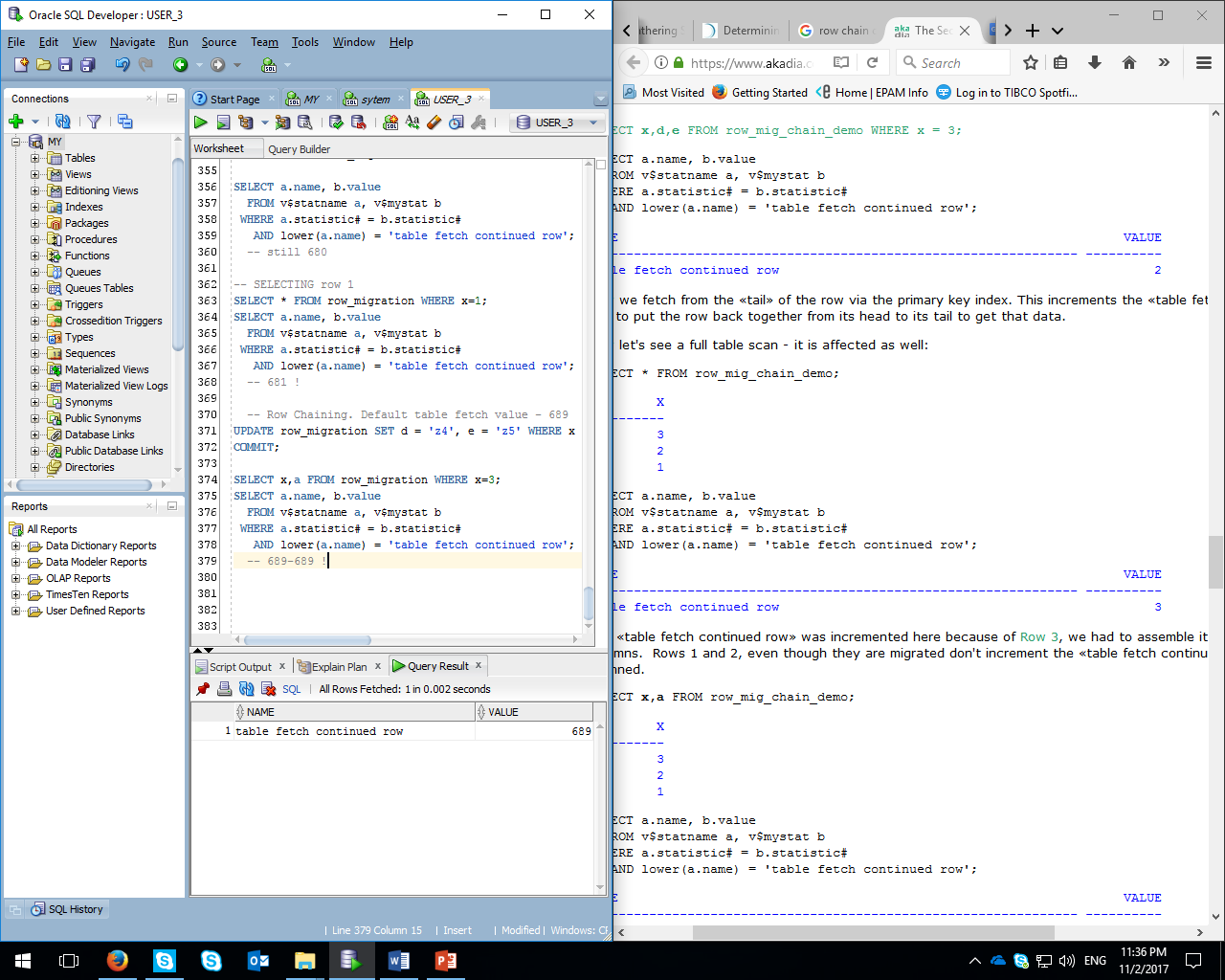


**Row chaining**



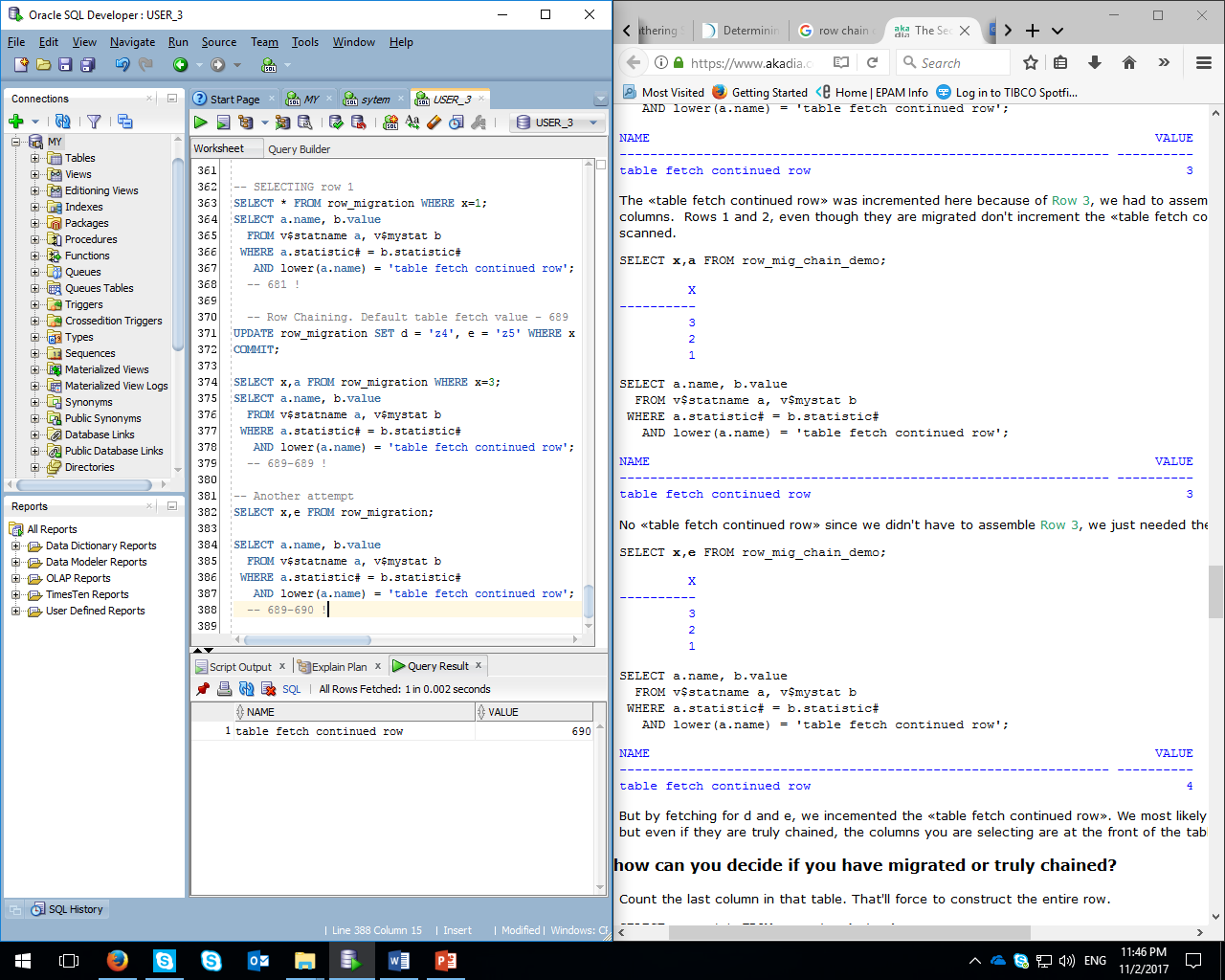
UPDATING table with adding additional values to empty columns WHERE x=3.

Executing statement with x=3 and getting +0 in table fetch continued row (689 is default value after training):



We’ve reached the +0 result because of the “X” & “a” cols situating in the “head” area – still in the 1st data block.

But if we choose columns “X” and “E” for example, well get an increase, because we have them in different blocks. It is so, because then can’t be put into one block by their sizes (2000 (a) + 2000 (b) + 2000 (c) + 2000 (d) … even (d) goes in another block because it fills our block up to 100%):



Rows that are chained can be also seen through the table analysis:

