**Report**

**on the**

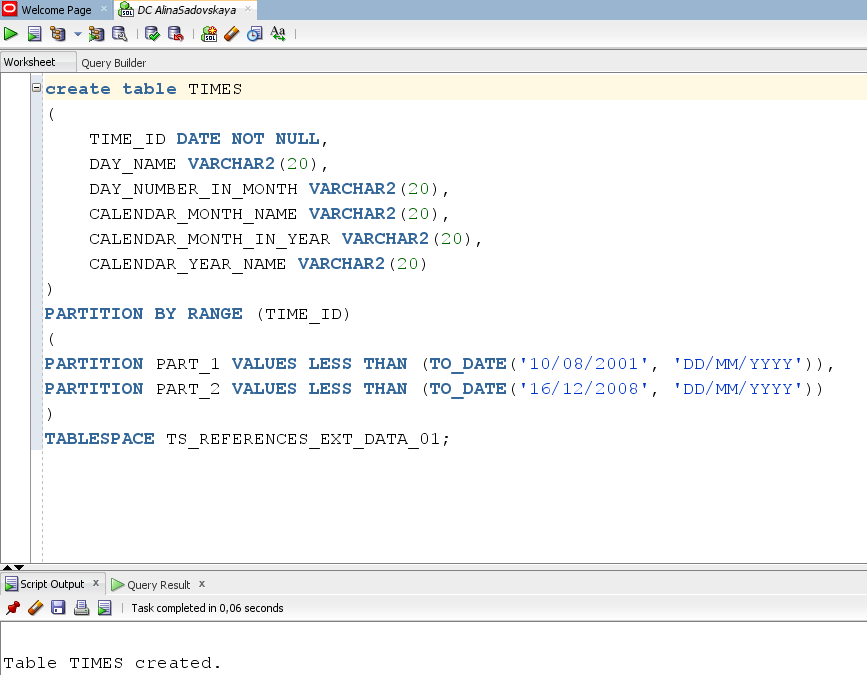
U1M9.LW.Partitioning

**Alina Sadovskaya**

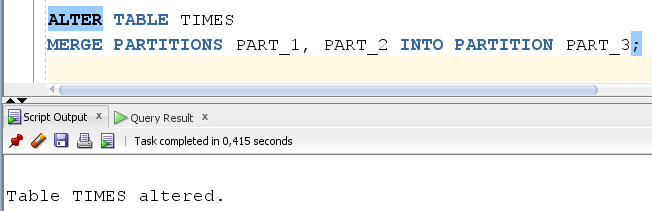
# 2. Oracle Architecture - Partitioning

2.1. CREATE Example of Range partitioning

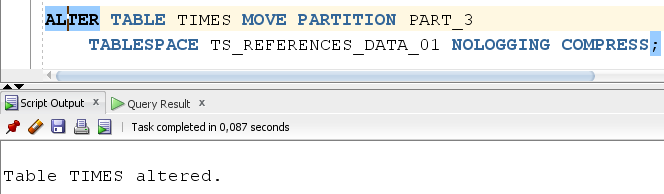
* Let's create a table using Range partitions:



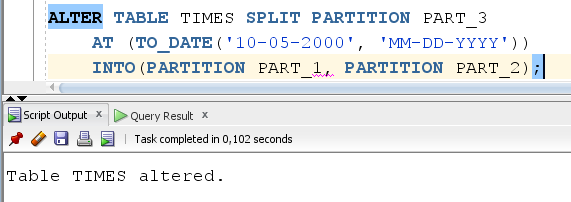
* Let's merge partitions



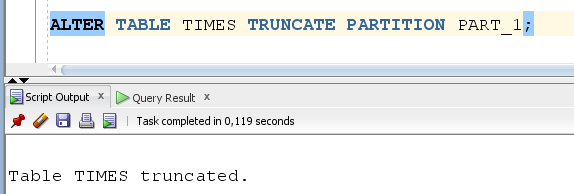
* Let’s move partitions



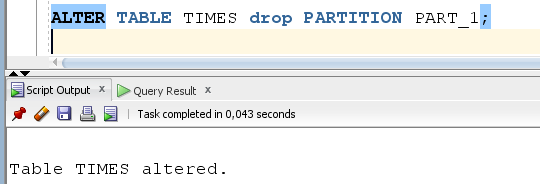
* Let’s split partitions



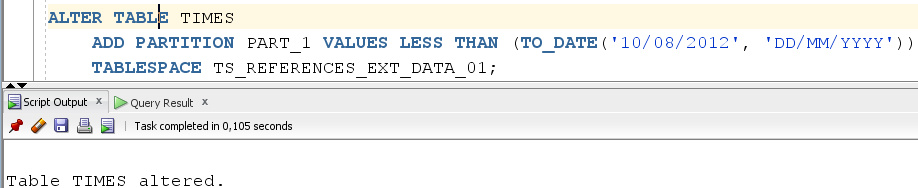
* Let’s truncate partition:



* Let’s drop partition:



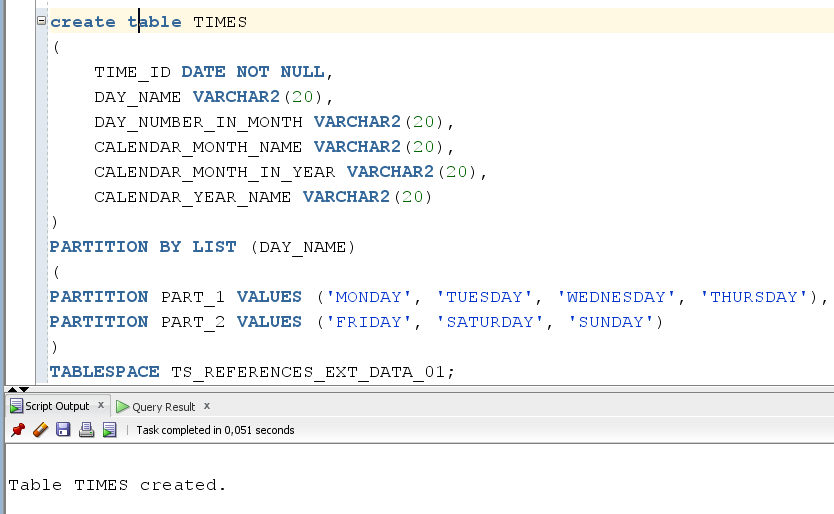
* Let’s add partition:



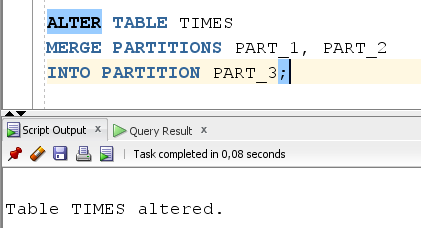
Drop table TIMES and create new table with LIST partitioning.

2.2. CREATE Example of LIST partitioning

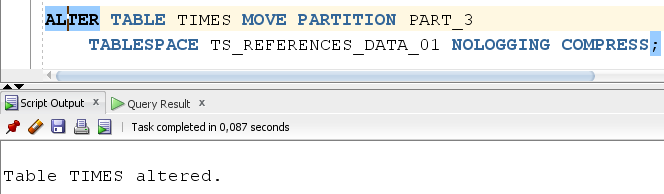
* Let's create a table using LIST partitioning:



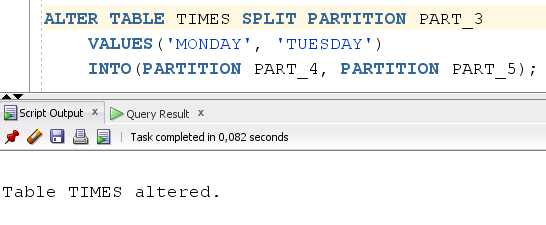
* Let's merge partitions



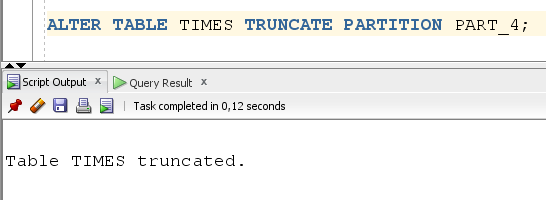
* Let’s move partitions



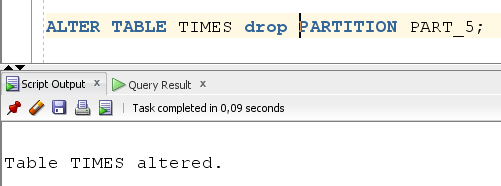
* Let’s split partitions



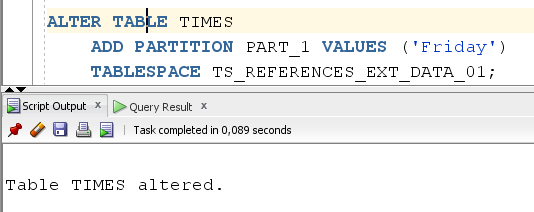
* Let’s truncate partition:



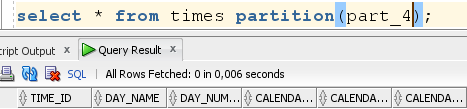
* Let’s drop partition:



* Let’s add partition:



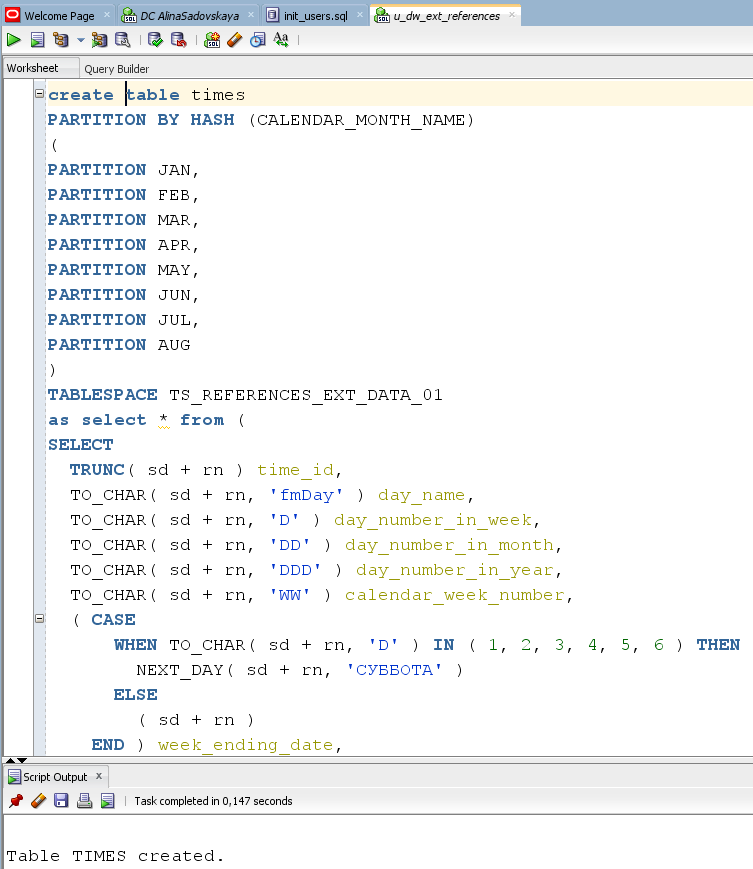
* We can select data by partition:



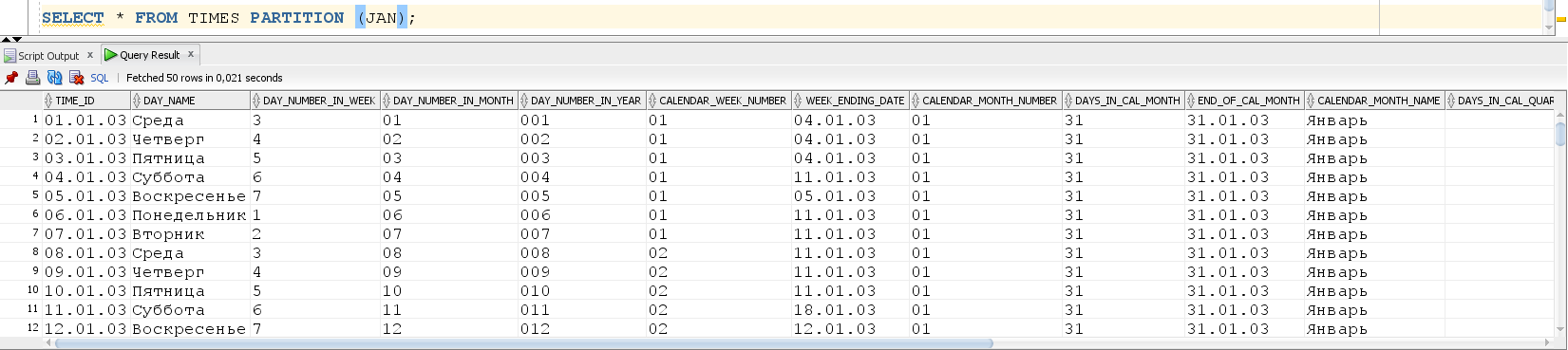
Drop table TIMES and create new table with HASH partitioning.

2.3. CREATE Example of HASH partitioning

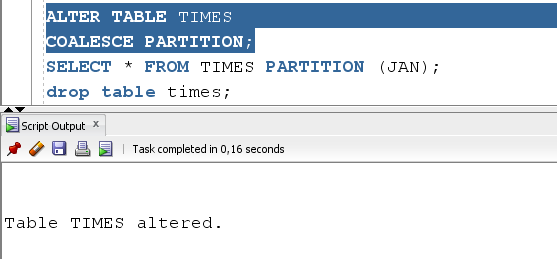
* Let's create a table using HASH partitioning from the user u\_dw\_ext\_references:



* Let’s select data by partition:

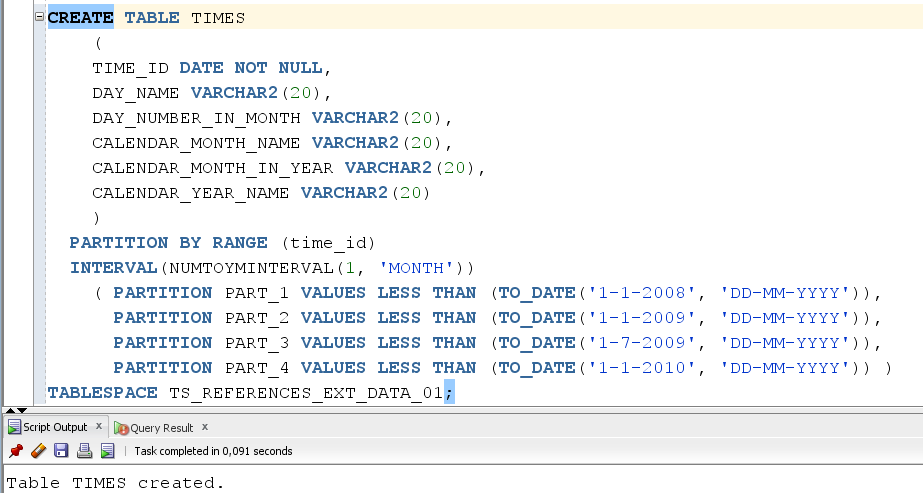


We can perform the MOVE\_PARTITION and TRUNCATE\_PARTITION operations, but they are performed in the same way as with other types of partitioning. it is much more interesting to consider the COALESCING\_PARTITIONS function:

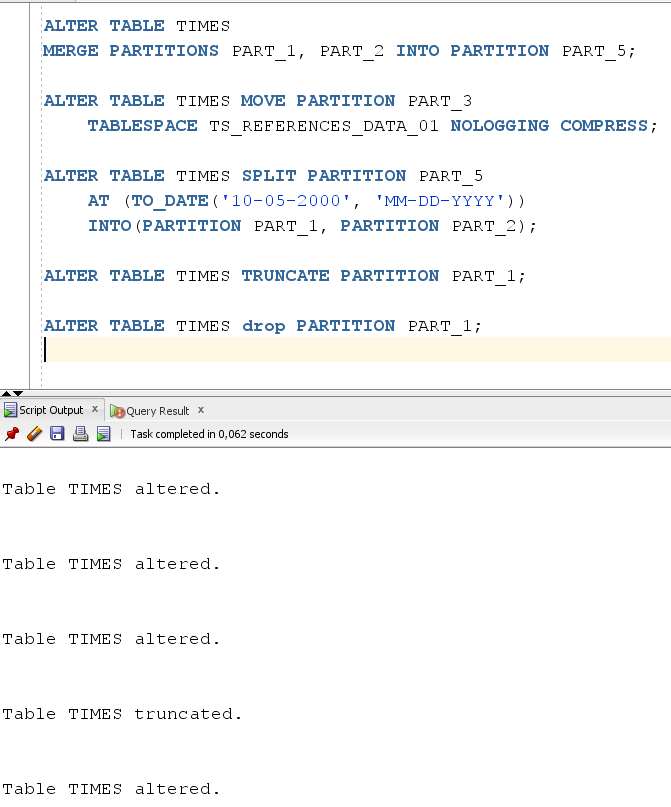


2.4. CREATE Example of INTERVAL partitioning

* Let's create a table using INTERVAL partitioning:

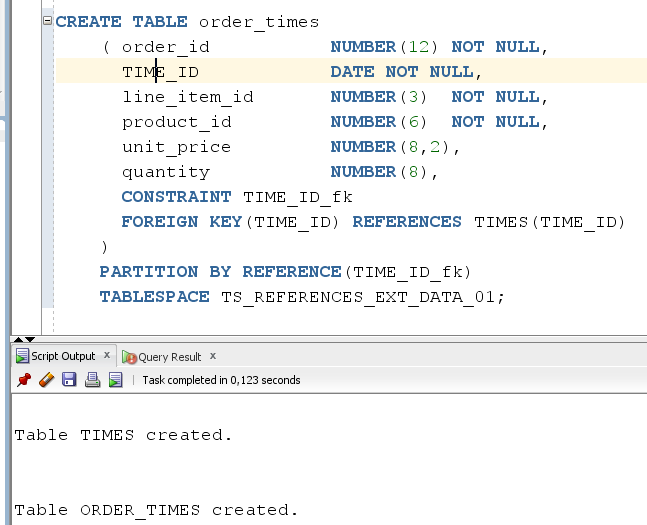
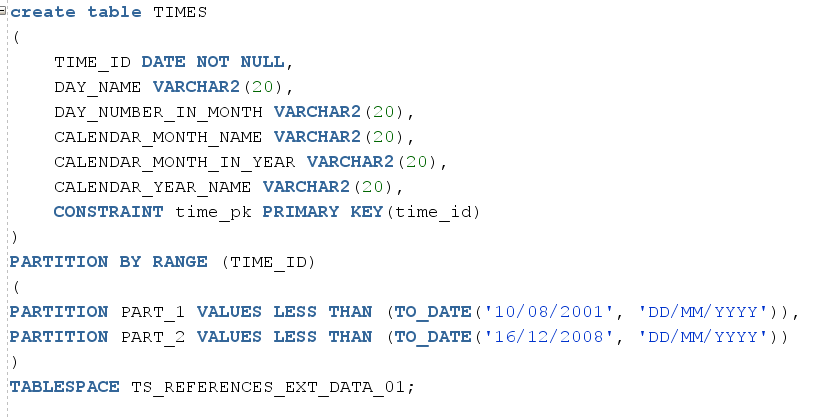


* We can merge, move, split, truncate, drop partition:

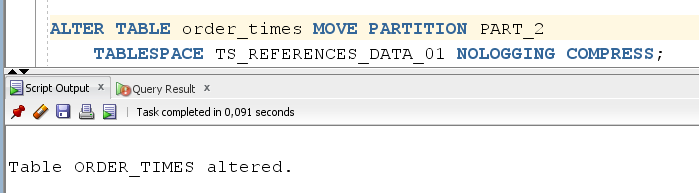


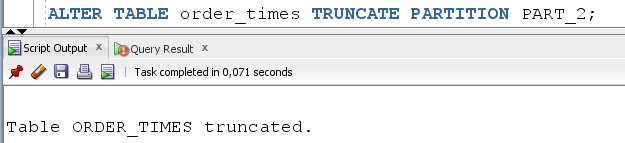
2.4. CREATE Example of REFERENCE partitioning

* Let's create a table using REFERENCE partitioning:



* We can move partition:



* Truncate partition:

# 3. Business Task - Partitioning Facts

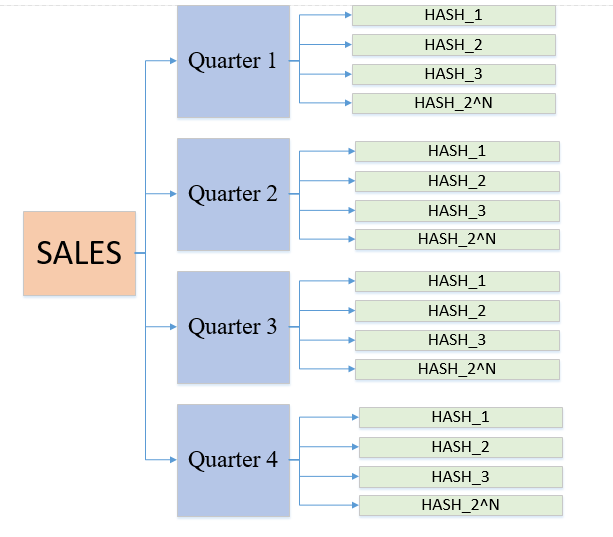
## 3.1. Partitioning Facts

The query execution speed, which works with fact table, can be increased with partitioning.

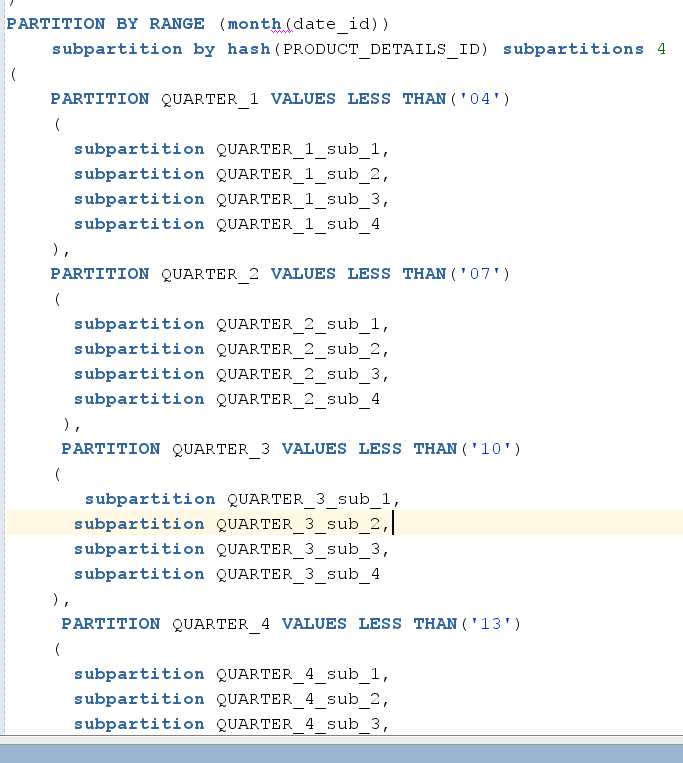
In my opinion, the most optimal way is to divide our data into 4 parts (quarterly-information for 3 months).

* Range Partition by DATE\_ID. The sales are divided by quarters.
* Hash partition of every quarter by PRODUCT\_DETAILS\_ID (number of sub partitions is 2^N(in my case I chose 4), because it should be the degree of 2)

Example of a result table of facts with partition taken into account:



Here is a table with the suggested partition option:



The code for creating a fact table is stored in the document "scripts/LW\_Mark\_Formelle".