****

**EPAM Training Center**

**DWH**

**Tasks 5**

**Report**

**Minsk, 2017**

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04.11.2017 | 0.1 | Description of workflow (Tasks 5) | Olga Hilko |
|  |  |  |  |

Contests

[Introduction 4](#_Toc497490080)

[Purpose 4](#_Toc497490081)

[1 Table access 4](#_Toc497490082)

[1.1 Full table scan 4](#_Toc497490083)

[2 Index scan 4](#_Toc497490084)

[2.1 Index Clustering Factor 4](#_Toc497490085)

[2.2 Index Unique Scan 6](#_Toc497490086)

[2.3 Index Range Scan 6](#_Toc497490087)

[2.4 Skip scan 7](#_Toc497490088)

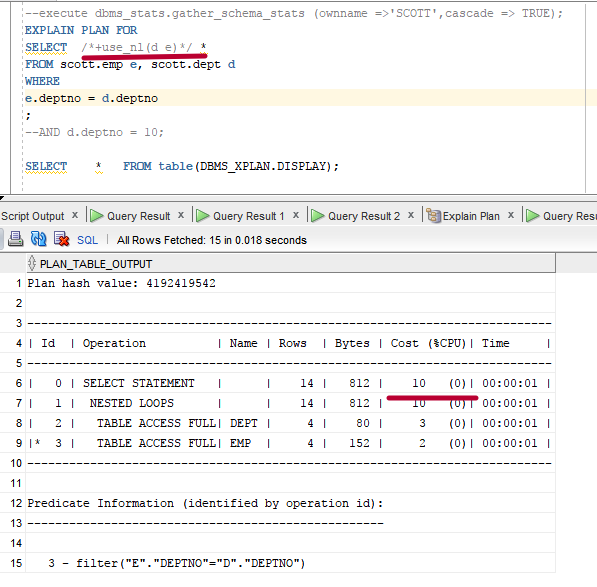
# Introduction

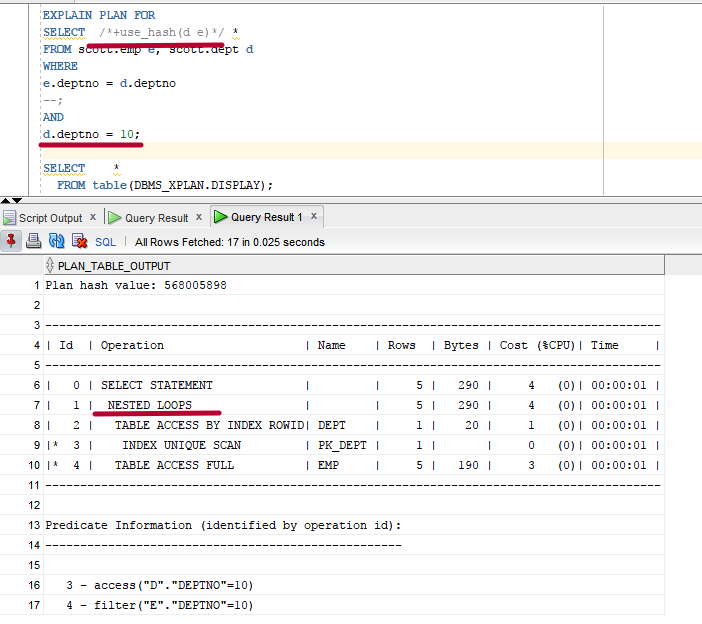
## Purpose

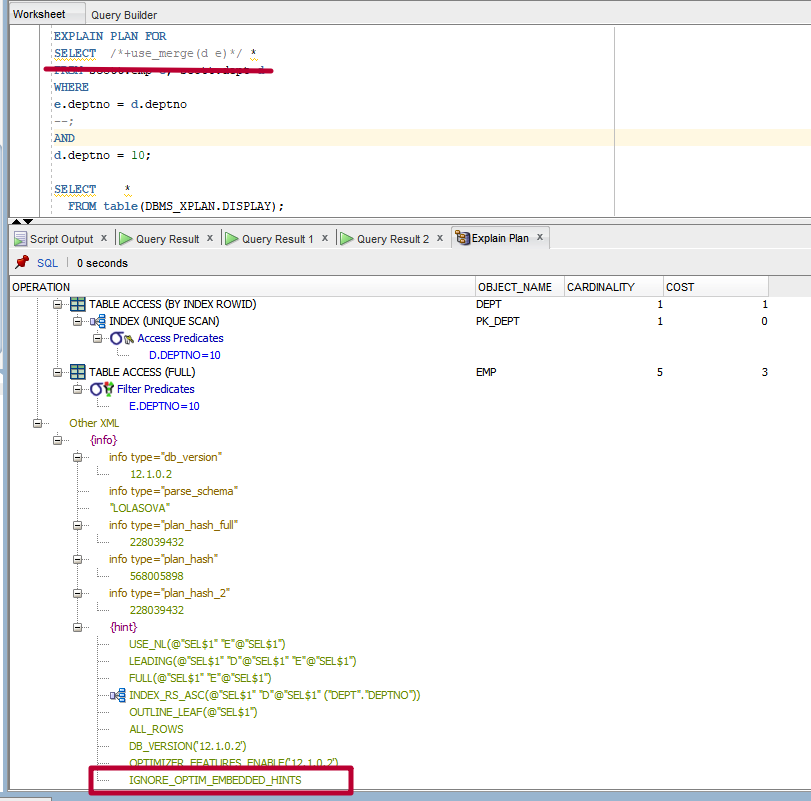
This document includes the results of completed task 5.

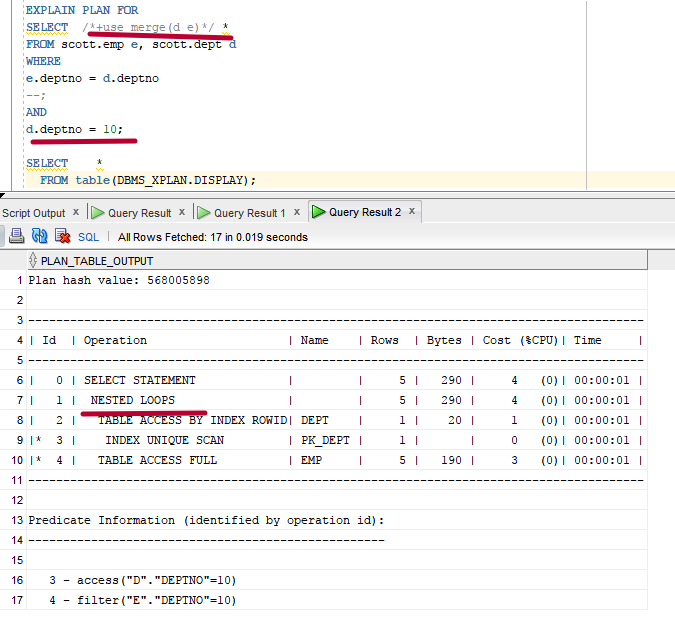
# Nested Loop

Nested loop without index and pointed where-clouse







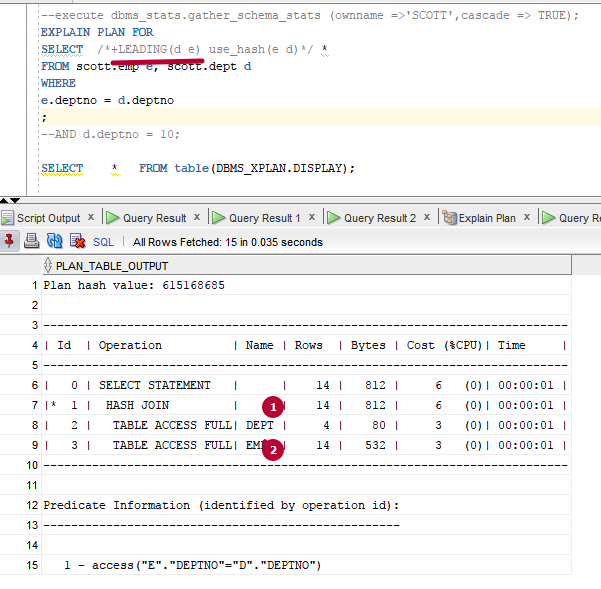


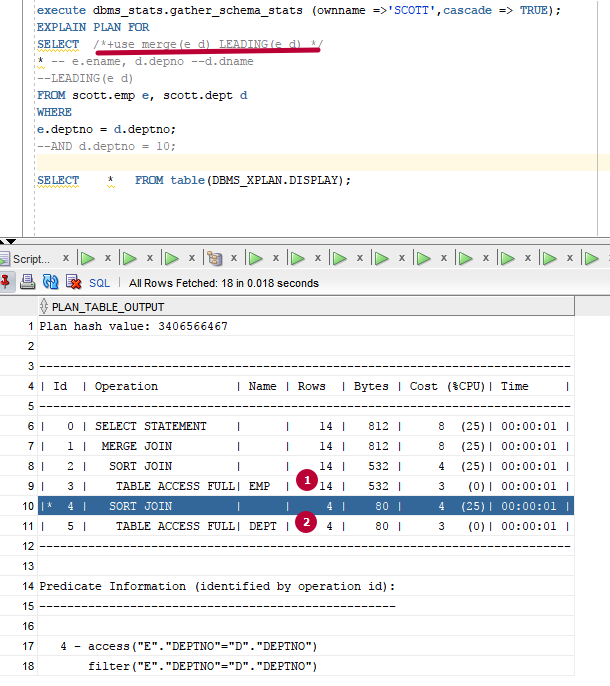
## 

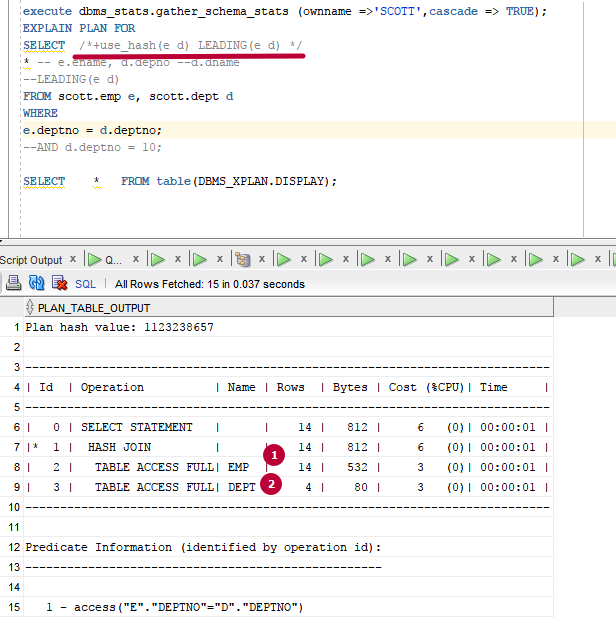
We have two little tables and both equal search condition on indexed field and inner join by the same field. Hints are ignored in all occasions, because they are incorrect in these cases and useless. Only nested loop is preferable.

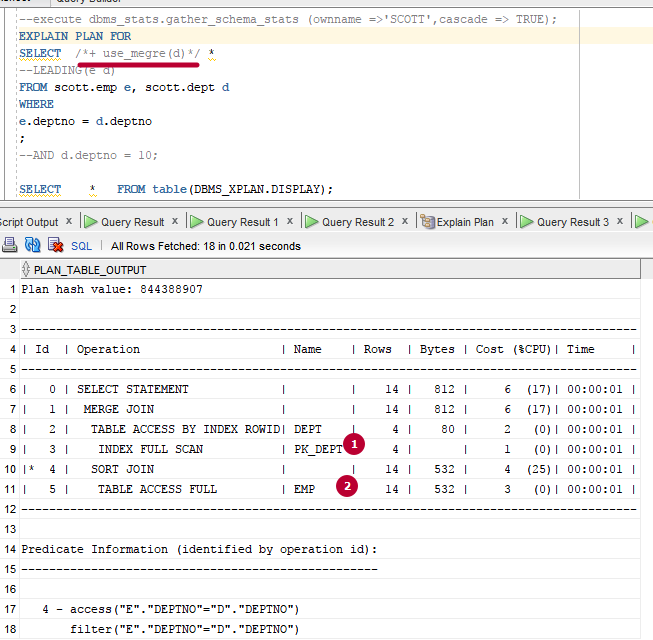
## Hash and merge equal-join

Now we change the query and take off the filter condition (only join is left in where-clause). Investigate all join-methods with different order. Both tables are little.



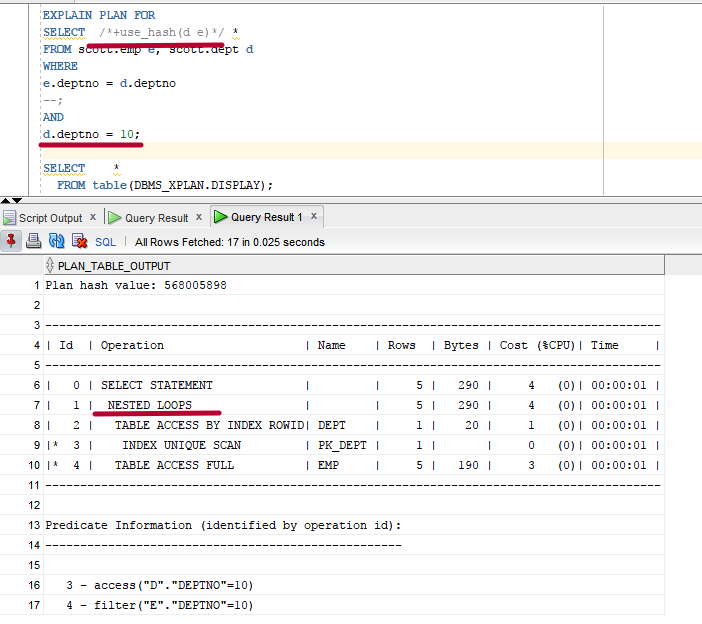


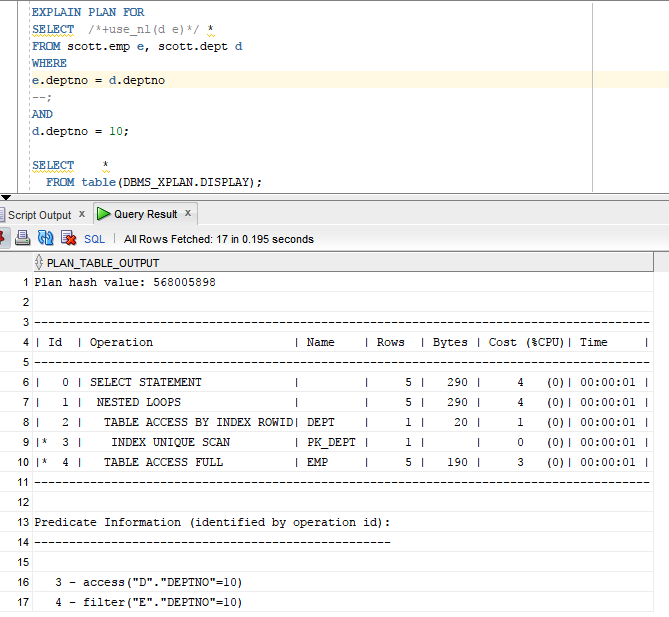




In the case of hash use – no difference in the performance, because both of the tables have the same size and are little. But the case with merge if we have the primary key on the driving table (all other data is taken by rowId) it works faster than inverse join order (in that case index is not used).

If we use the direct condition in where-clause only nested loop is used. Hints are ignored.

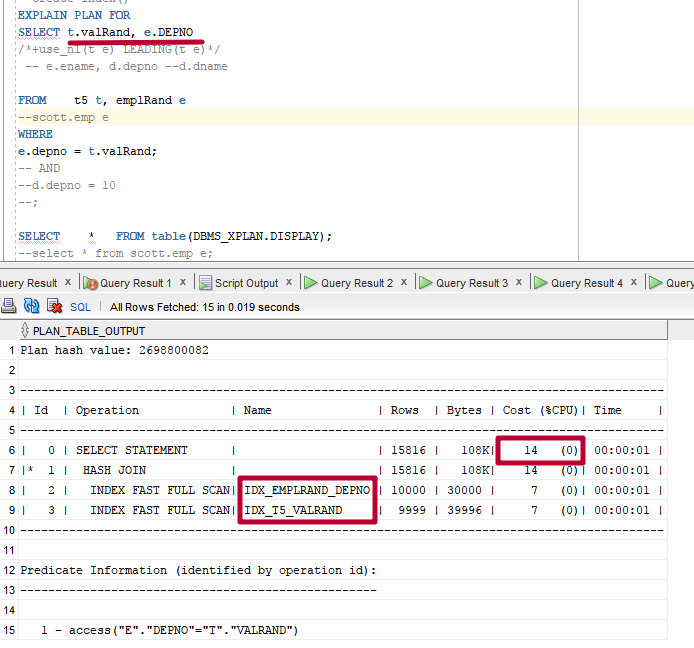


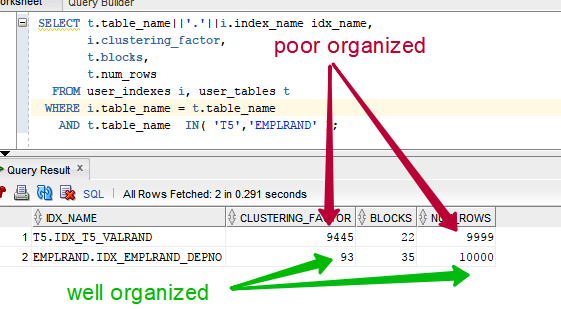


Indexes were created. One of them is well organized, the second is poor organized.

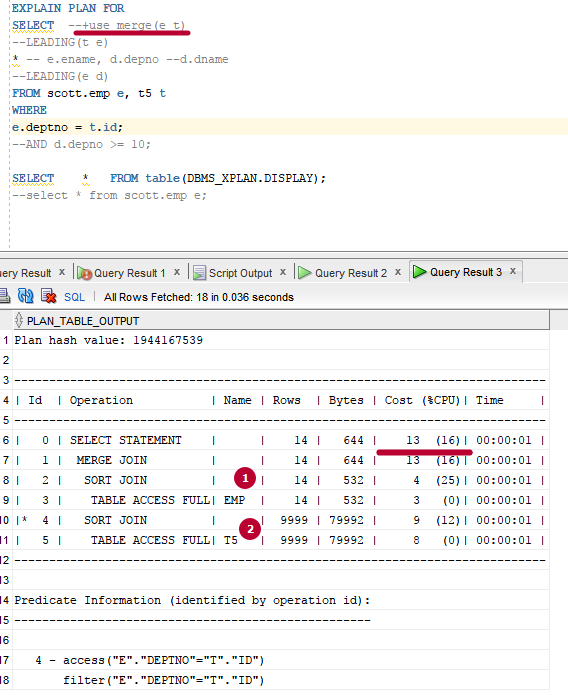
The query joins two big tables, indexes are used, join is provided by indexes.

Hint is proposed, but is ignored. Hash join by indexed is preferred (tables are big).

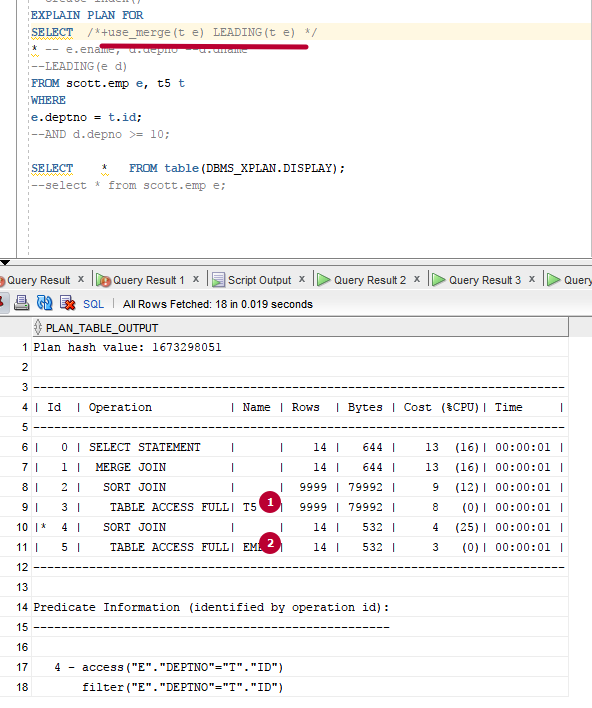




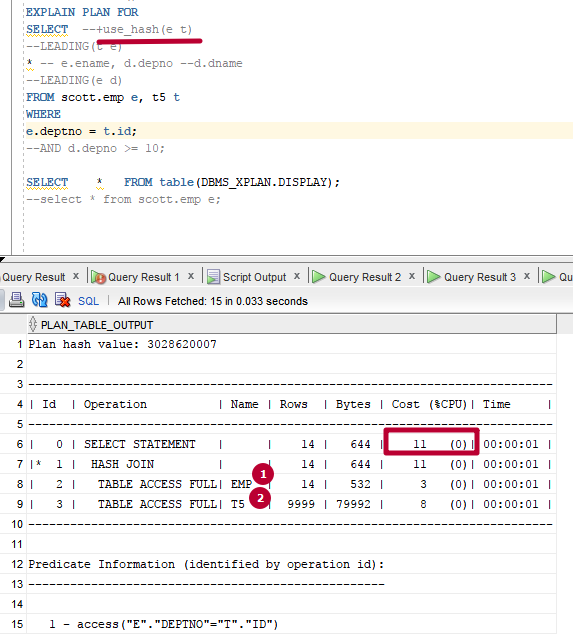
Merge little – big equi without/ index



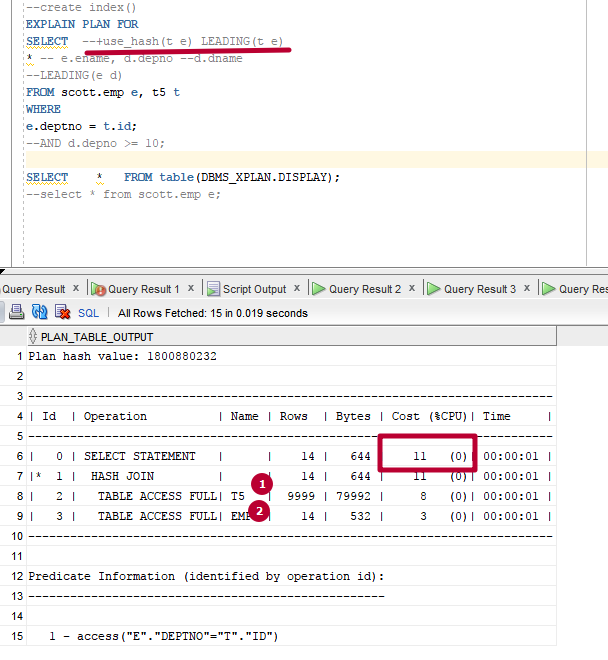
Merge big – little equi without/index



Hash Little – big equi without/ index



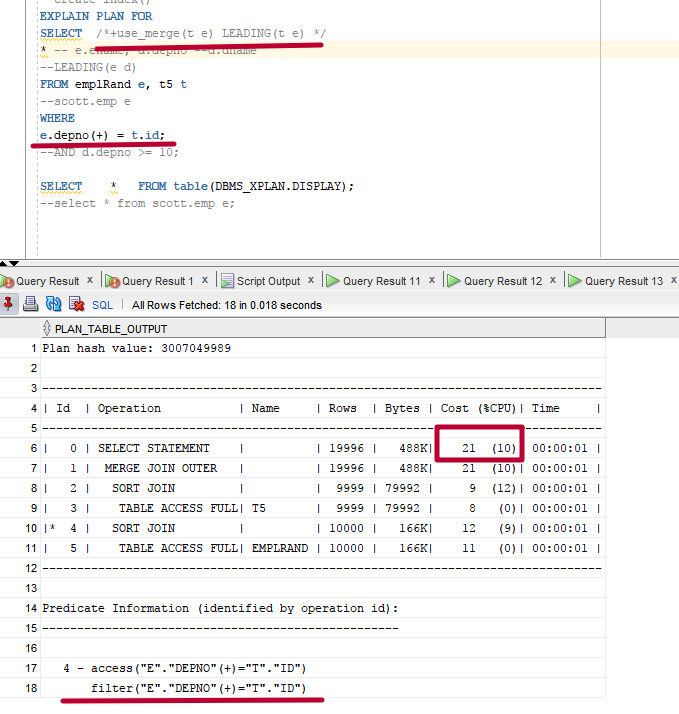
Hash big –Little equi without/ index



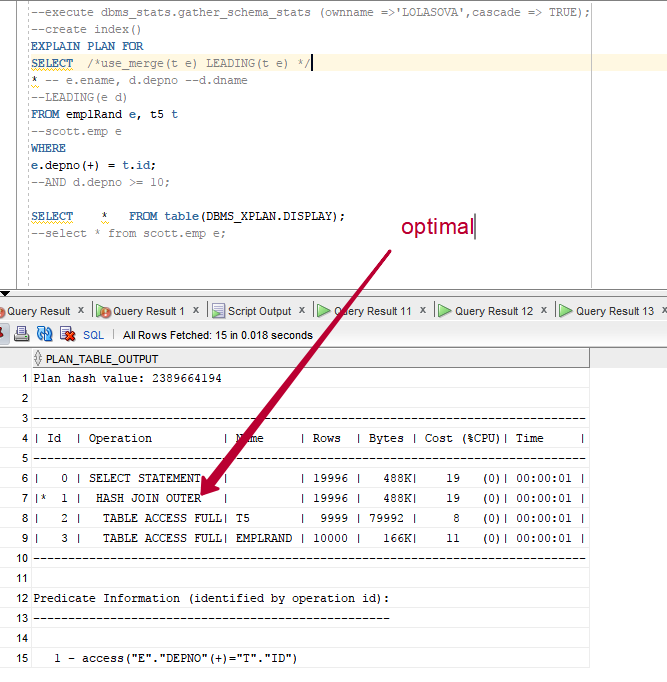
In these cases there is no difference which table is the first, but hash method if faster, because sorting if avoided.

# Left/right join

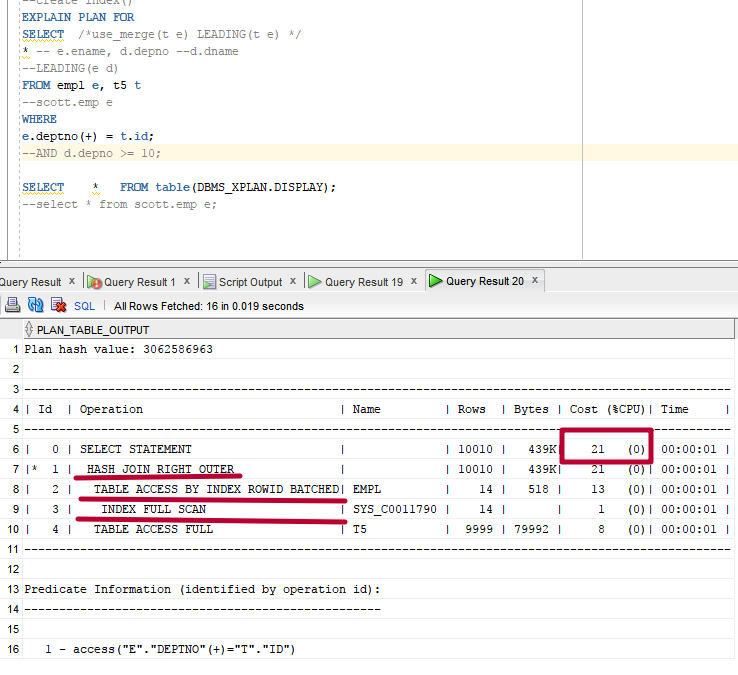
Big-big equal + without/index (right join)



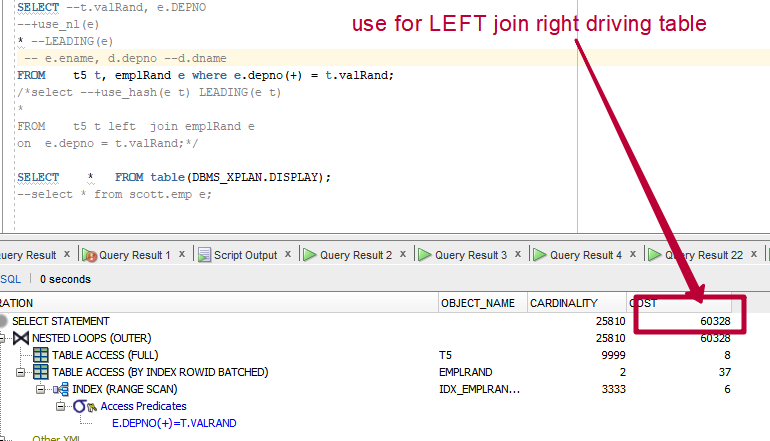
When no hints are used optimal plan is chosen as hash.

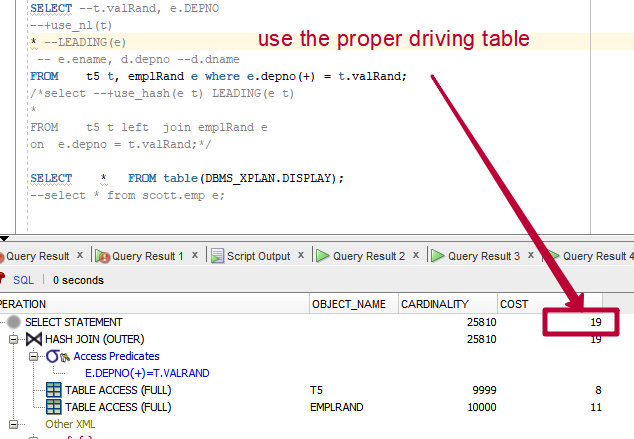


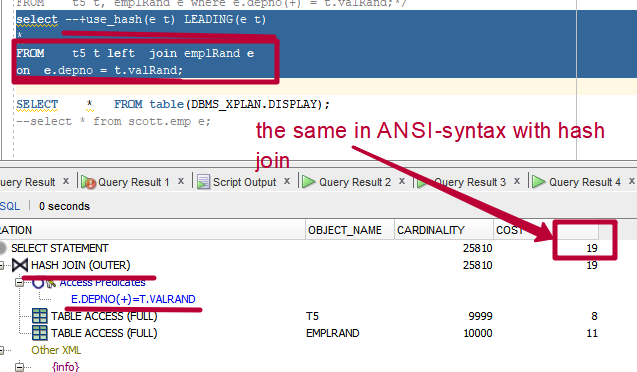
little-big equal + w/I (right join)



## Using oracle and ANSI-syntax

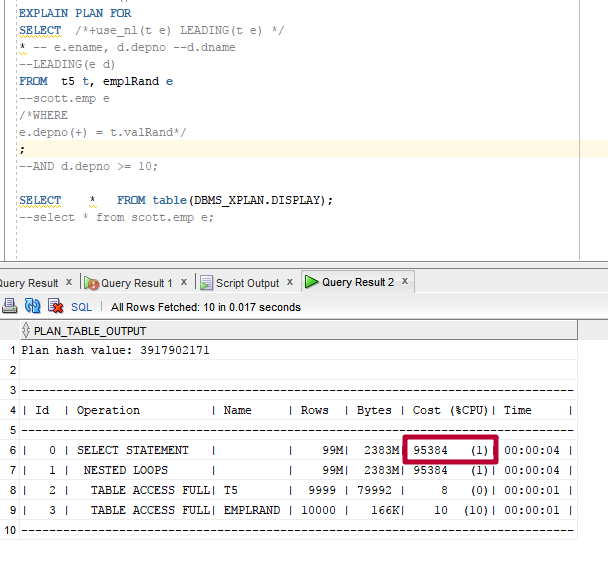


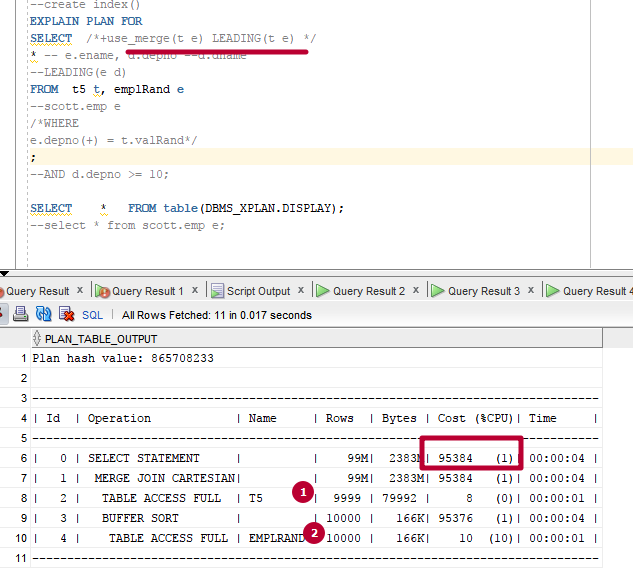


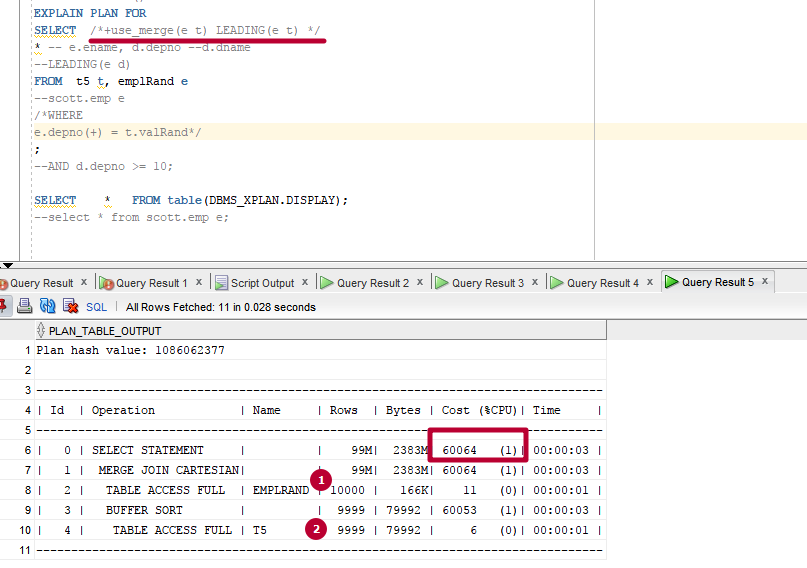


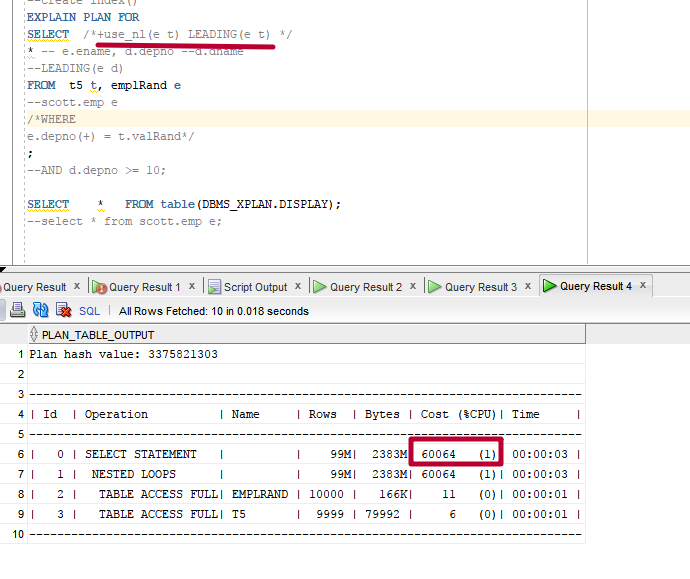
# Cross join

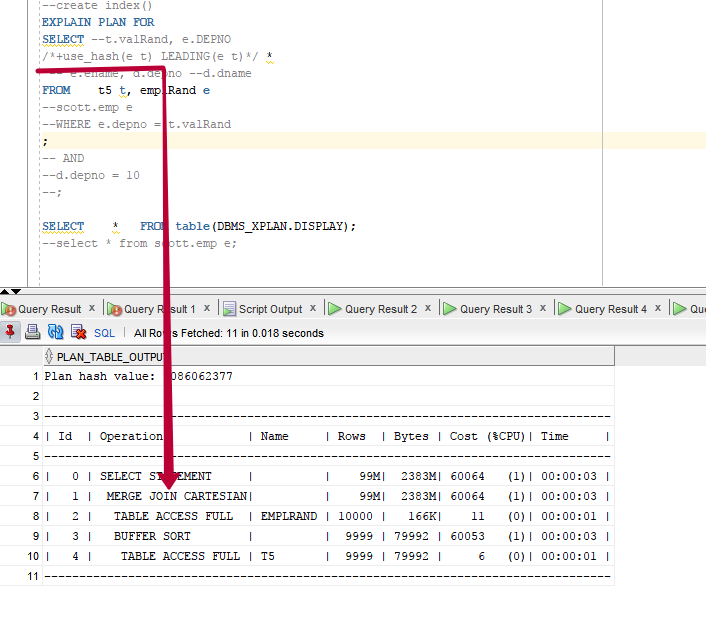
Two big tables are cross joined, different methods of join are used. No difference in comparison between nested loop and merge join. But there is a big difference in the driving and driven-on tables. The fastest is the case when the first (driving) table has bigger row size and the driven table.





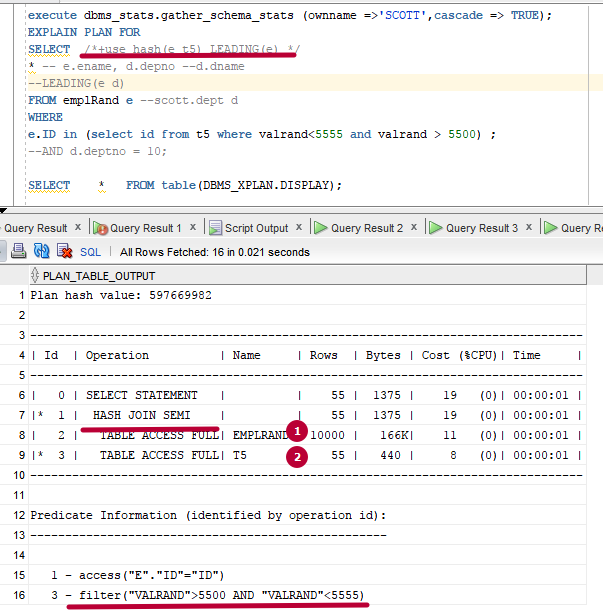


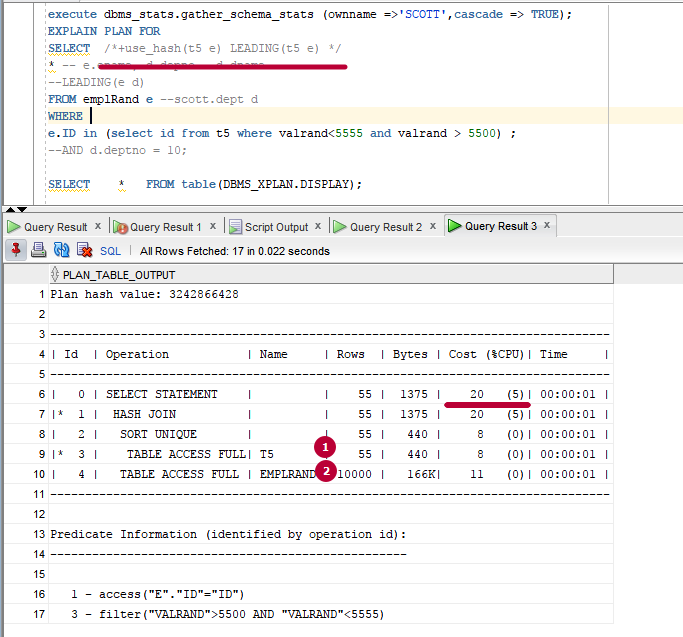




The hash join in not effective to be used in this case, hint is ignored, the merge join method is used.

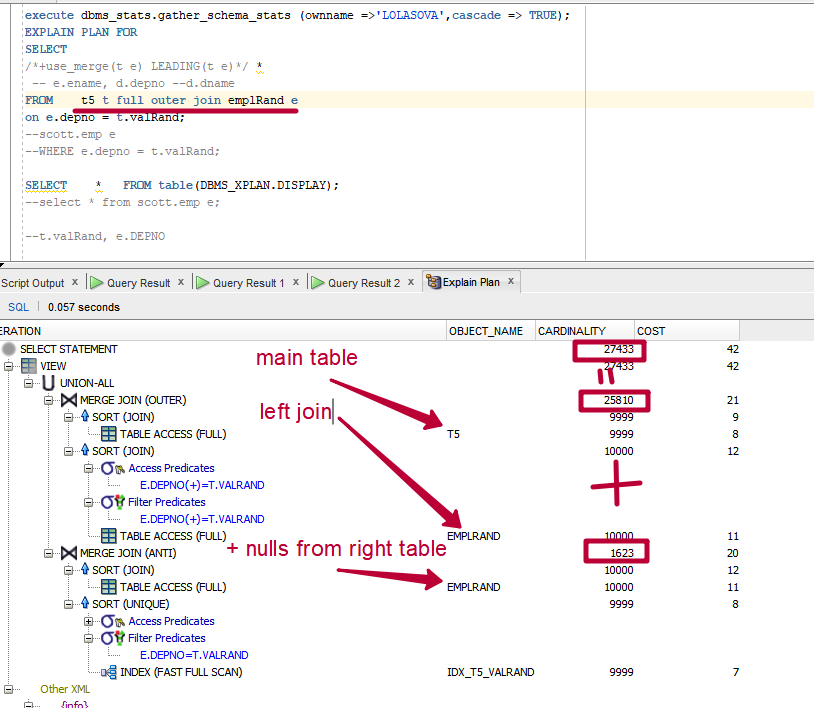
# Semi join

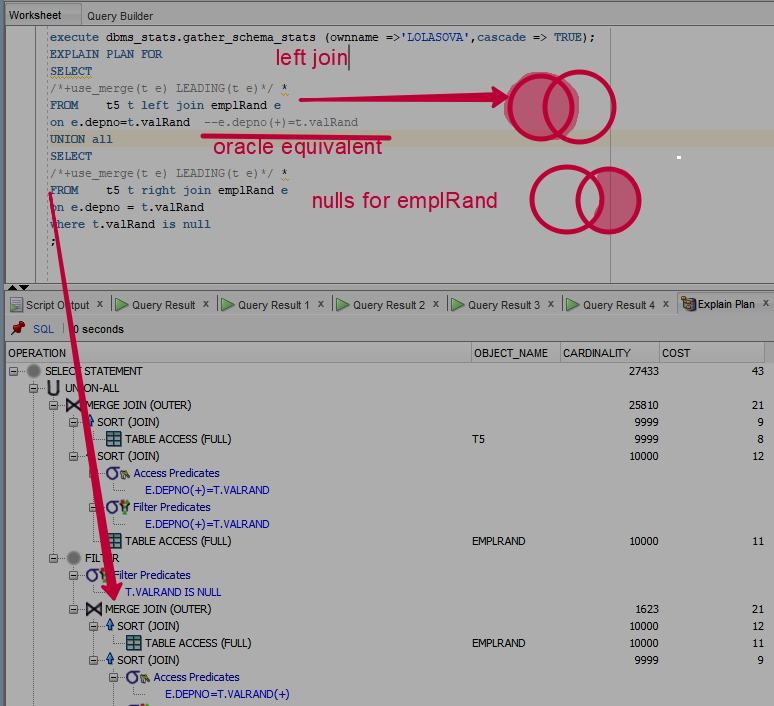




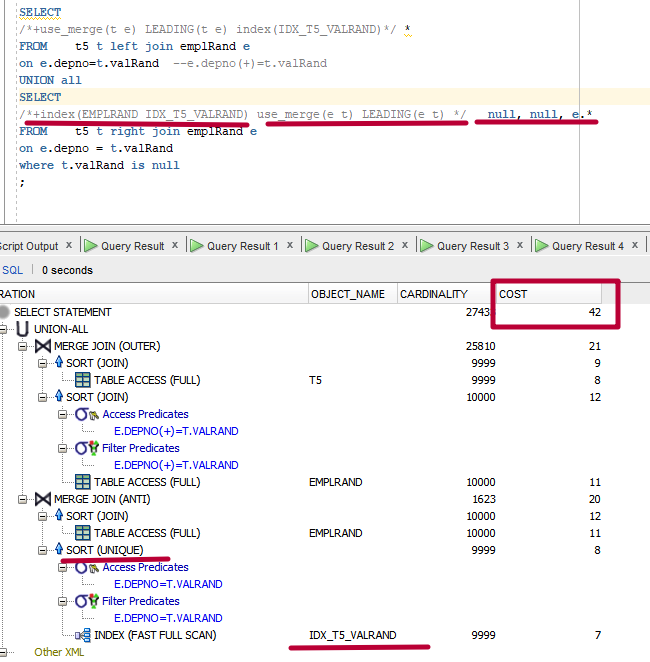
The choice driven table in hash table join influences the semi-join plan.

# Full outer join





Exact ANSI-equivalent



# In conclusion

I have no more forces to make the table, that is why I’d like to ask you to give me the opportunity to make the conclusion in my own words.

We have investigated how to manage execution plans using the hints and studied different methods of joins.

The fasters is nested loop, but it is effective when there is an equi-clause in where predicate (or little subset from big table) or inner join-type and we have index in this field. If there is no index nested loop is less effective, then hash or sorted merge.

Hash method if faster then merge, because sorting if avoided, but it can be used only in equi-clauses and hashing table must have the proper size to be allocated in the buffer.

Merge sort is effective when we join by indexed fields and sorting is not needed (indexes are already sorted). They work in the best way if the table is IOT (no need to go to the main table to get all necessary data for the query. This method is also used if there are big tables and they can not be allocated in the buffer or we have non-unique joins on big tables.