## Why use Oracle In-Memory database from another perspective

A lot of people are talking about why or why not use Oracle In-memory database in their applications and most of them are too focused on the size of the database or whether it is an OLAP application. It seems that small and medium size databases are not suitable for using Oracle In-memory database option. But if your OLTP databases are suffering from performance bottleneck and you are looking for solutions, I think Oracle In-Memory database option should be on your solutions list, especially when you are planning to upgrade your hardware.

## The SQL optimizer plan space is getting bigger in Oracle In-Memory database

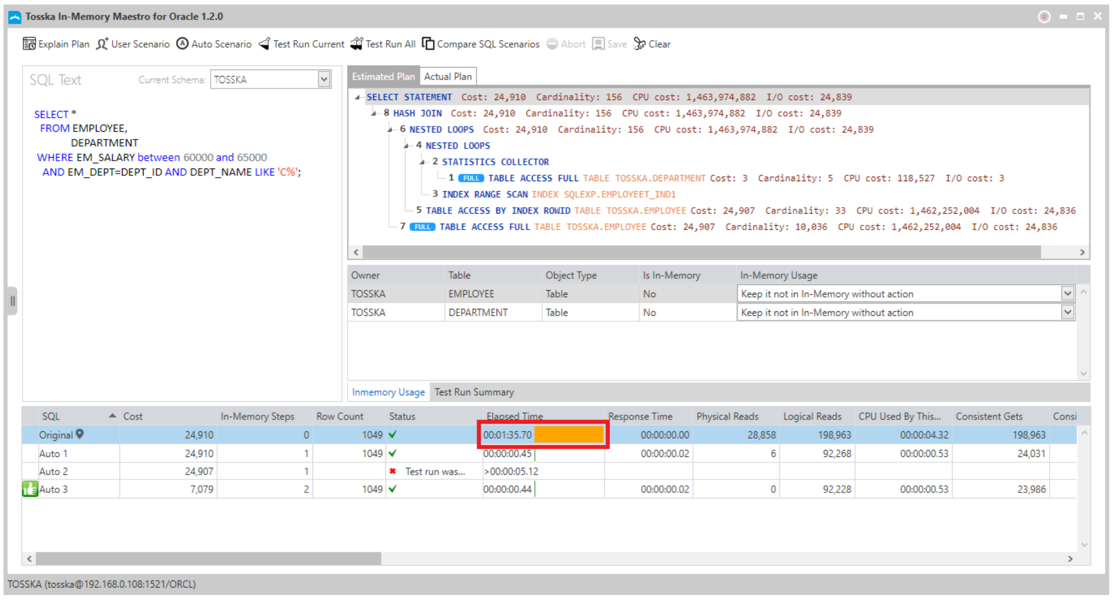
Plan space is the size of potential process methods that database SQL optimizer will consider before processing your input SQL statements. A bigger plan space size means that database SQL optimizer will consider more potential methods to process your SQL statements. So, your SQL statements have more chances to run faster. With Oracle's In-Memory database new In-Memory data access methods, Oracle SQL optimizer will consider a bigger plan space before the execution of your SQL statement. Due to the multiplier effect, a new In-Memory execution plan step might result in tremendous huge expansion of plan space size. For example, a SQL statement with 5 tables in In-Memory, a new "Table Access Inmemory Full" step to each table's access method, the result is C(5,1)+C(5,2)+C(5,3)+C(5,4)+C(5,5)=5+10+10+5+1=31. But don't forget this multiplier is applicable to every single execution plan in original plan space tree, it means the new plan space size might be potentially 31 times bigger than the original plan space.

## How OLTP database get advantage from In-Memory database?

An OLTP database is a transaction-oriented application which requires quick response time for each transaction, but it doesn't mean that there is no complex SQL for online report or data consolidation during business hours. If these kinds of slow SQL statements are running simultaneously with other online transactions, the overall performance will be affected. If hardware upgrade is one of the options, you should consider using Oracle In-Memory as one of the alternative solutions.

Here is an example that shows you how an OLTP SQL statement gets benefit from using Oracle In-Memory database option.

The following is a typical OLTP SQL with all tables analyzed, the original execution plan shows an Oracle adaptive plan that Hash Join or Nested Loop will be decided during the initial stage of execution. The elapsed time of this SQL is around 1 minute and 35 seconds.



Tosska In-Memory Maestro for Oracle 1.2.0

Explain Plan | User Scenario | Auto Scenario | Test Run Current | Test Run All | Compare SQL Scenarios | Abort | Save | Clear

SQL Text — Current Schema: TOSSKA

```
SELECT *
 FROM EMPLOYEE,
      DEPARTMENT
WHERE EM_SALARY between 60000 and 65000
 AND EM_DEPT=DEPT_ID AND DEPT_NAME LIKE 'C%';
```

Estimated Plan | Actual Plan

```
SELECT STATEMENT  Cost: 24,910  Cardinality: 156  CPU cost: 1,463,974,882  I/O cost: 24,839
  8 HASH JOIN  Cost: 24,910  Cardinality: 156  CPU cost: 1,463,974,882  I/O cost: 24,839
    6 NESTED LOOPS  Cost: 24,910  Cardinality: 156  CPU cost: 1,463,974,882  I/O cost: 24,839
      4 NESTED LOOPS
        2 STATISTICS COLLECTOR
          1 FULL TABLE ACCESS FULL TABLE TOSSKA.DEPARTMENT Cost: 3  Cardinality: 5  CPU cost: 118,527  I/O cost: 3
        3 INDEX RANGE SCAN INDEX SQLEXP.EMPLOYEET_IND1
      5 TABLE ACCESS BY INDEX ROWID TABLE TOSSKA.EMPLOYEE Cost: 24,907  Cardinality: 33  CPU cost: 1,462,252,004  I/O cost: 24,836
    7 FULL TABLE ACCESS FULL TABLE TOSSKA.EMPLOYEE Cost: 24,907  Cardinality: 10,036  CPU cost: 1,462,252,004  I/O cost: 24,836
```

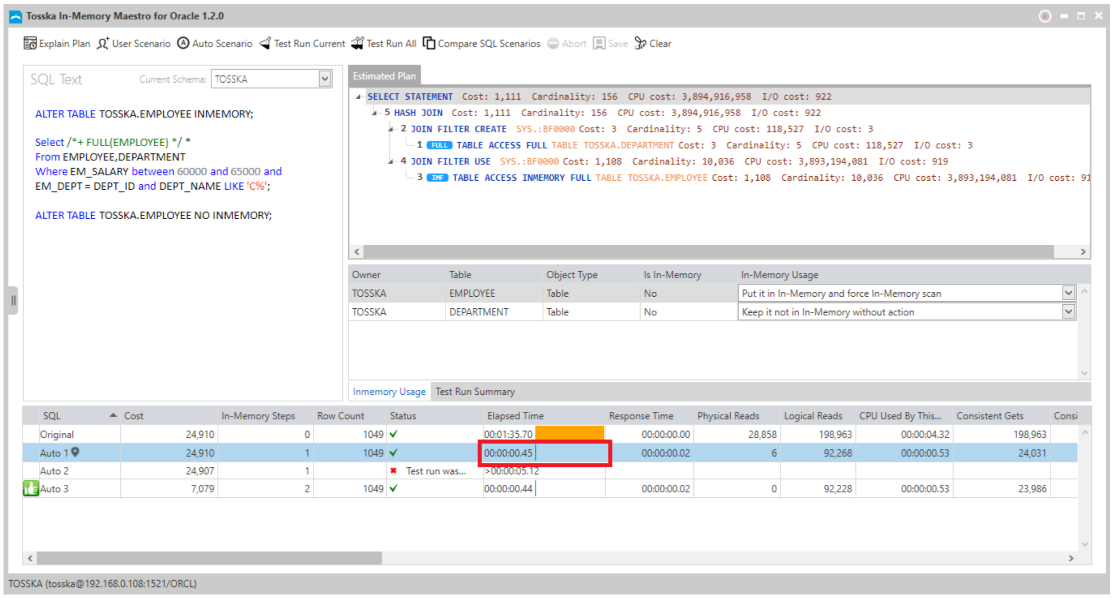
| Owner | Table | Object Type | Is In-Memory | In-Memory Usage |
|---|---|---|---|---|
| TOSSKA | EMPLOYEE | Table | No | Keep it not in In-Memory without action |
| TOSSKA | DEPARTMENT | Table | No | Keep it not in In-Memory without action |

Inmemory Usage | Test Run Summary

| SQL | Cost | In-Memory Steps | Row Count | Status | Elapsed Time | Response Time | Physical Reads | Logical Reads | CPU Used By This... | Consistent Gets | Consi |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Original | 24,910 | 0 | 1049 | ✓ | 00:01:35.70 | 00:00:00.00 | 28,858 | 198,963 | 00:00:04.32 | 198,963 | |
| Auto 1 | 24,910 | 1 | 1049 | ✓ | 00:00:00.45 | 00:00:00.02 | 6 | 92,268 | 00:00:00.53 | 24,031 | |
| Auto 2 | 24,907 | 1 | | ✗ Test run was... | >00:00:05.12 | | | | | | |
| Auto 3 | 7,079 | 2 | 1049 | ✓ | 00:00:00.44 | 00:00:00.02 | 0 | 92,228 | 00:00:00.53 | 23,986 | |

TOSSKA (tosska@192.168.0.108:1521/ORCL)

Let's put EMPLOYEE table into Oracle In-Memory with a force hint and Oracle use it with "Table Access Inmemory Full" scan, the following benchmark shows that the "Auto 1" SQL took only 0.45 second to finish with more than 200 times faster than the original execution plan. The solution indicates that the introduction of a new "Table Access Inmemory Full" scan to the EMPLOYEE table has actually created a much better execution plan compared with the original one.



Tosska In-Memory Maestro for Oracle 1.2.0

Explain Plan | User Scenario | Auto Scenario | Test Run Current | Test Run All | Compare SQL Scenarios | Abort | Save | Clear

SQL Text — Current Schema: TOSSKA

```
ALTER TABLE TOSSKA.EMPLOYEE INMEMORY;

Select /*+ FULL(EMPLOYEE) */ *
From EMPLOYEE,DEPARTMENT
Where EM_SALARY between 60000 and 65000 and
EM_DEPT = DEPT_ID and DEPT_NAME LIKE 'C%';

ALTER TABLE TOSSKA.EMPLOYEE NO INMEMORY;
```

Estimated Plan

```
SELECT STATEMENT  Cost: 1,111  Cardinality: 156  CPU cost: 3,894,916,958  I/O cost: 922
  5 HASH JOIN  Cost: 1,111  Cardinality: 156  CPU cost: 3,894,916,958  I/O cost: 922
    2 JOIN FILTER CREATE  SYS.:BF0000 Cost: 3  Cardinality: 5  CPU cost: 118,527  I/O cost: 3
      1 FULL TABLE ACCESS FULL TABLE TOSSKA.DEPARTMENT Cost: 3  Cardinality: 5  CPU cost: 118,527  I/O cost: 3
    4 JOIN FILTER USE  SYS.:BF0000 Cost: 1,108  Cardinality: 10,036  CPU cost: 3,893,194,081  I/O cost: 919
      3 IMF TABLE ACCESS INMEMORY FULL TABLE TOSSKA.EMPLOYEE Cost: 1,108  Cardinality: 10,036  CPU cost: 3,893,194,081  I/O cost: 91
```

| Owner | Table | Object Type | Is In-Memory | In-Memory Usage |
|---|---|---|---|---|
| TOSSKA | EMPLOYEE | Table | No | Put it in In-Memory and force In-Memory scan |
| TOSSKA | DEPARTMENT | Table | No | Keep it not in In-Memory without action |

Inmemory Usage | Test Run Summary

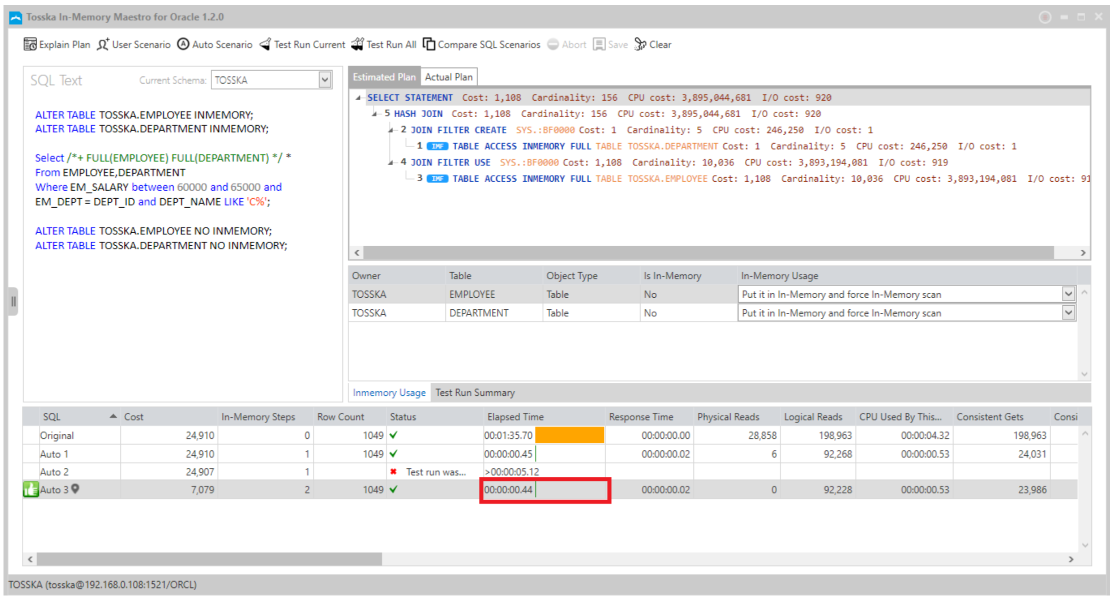
| SQL | Cost | In-Memory Steps | Row Count | Status | Elapsed Time | Response Time | Physical Reads | Logical Reads | CPU Used By This... | Consistent Gets | Consi |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Original | 24,910 | 0 | 1049 | ✓ | 00:01:35.70 | 00:00:00.00 | 28,858 | 198,963 | 00:00:04.32 | 198,963 | |
| Auto 1 | 24,910 | 1 | 1049 | ✓ | 00:00:00.45 | 00:00:00.02 | 6 | 92,268 | 00:00:00.53 | 24,031 | |
| Auto 2 | 24,907 | 1 | | ✗ Test run was... | >00:00:05.12 | | | | | | |
| Auto 3 | 7,079 | 2 | 1049 | ✓ | 00:00:00.44 | 00:00:00.02 | 0 | 92,228 | 00:00:00.53 | 23,986 | |

TOSSKA (tosska@192.168.0.108:1521/ORCL)

## There is no need to put all tables into In-Memory for OLTP's SQL

When using an OLTP database, we don't want to introduce too much overhead to the online transactions by populating all tables into In-Memory. So, the goal of In-Memory SQL tuning for OLTP SQL is not to select the best performance solution, but the most cost-effective solution with fewer tables In-Memory tables but still acceptable performance improvement instead.

The following "Auto 3" solution with all tables are put into In-Memory, but the improvement is just 0.01 second better than the "Auto 1" which only requires one table to be put into In-Memory. Therefore, it is obvious that "Auto 1" is a much more preferable choice.



```
ALTER TABLE TOSSKA.EMPLOYEE INMEMORY;
ALTER TABLE TOSSKA.DEPARTMENT INMEMORY;

Select /*+ FULL(EMPLOYEE) FULL(DEPARTMENT) */ *
From EMPLOYEE,DEPARTMENT
Where EM_SALARY between 60000 and 65000 and
EM_DEPT = DEPT_ID and DEPT_NAME LIKE 'C%';

ALTER TABLE TOSSKA.EMPLOYEE NO INMEMORY;
ALTER TABLE TOSSKA.DEPARTMENT NO INMEMORY;
```

```
SELECT STATEMENT  Cost: 1,108  Cardinality: 156  CPU cost: 3,895,044,681  I/O cost: 920
  5 HASH JOIN  Cost: 1,108  Cardinality: 156  CPU cost: 3,895,044,681  I/O cost: 920
    2 JOIN FILTER CREATE  SYS.:BF0000 Cost: 1  Cardinality: 5  CPU cost: 246,250  I/O cost: 1
      1 IMF TABLE ACCESS INMEMORY FULL TABLE TOSSKA.DEPARTMENT Cost: 1  Cardinality: 5  CPU cost: 246,250  I/O cost: 1
    4 JOIN FILTER USE  SYS.:BF0000 Cost: 1,108  Cardinality: 10,036  CPU cost: 3,893,194,081  I/O cost: 919
      3 IMF TABLE ACCESS INMEMORY FULL TABLE TOSSKA.EMPLOYEE Cost: 1,108  Cardinality: 10,036  CPU cost: 3,893,194,081  I/O cost: 91
```

| Owner | Table | Object Type | Is In-Memory | In-Memory Usage |
|---|---|---|---|---|
| TOSSKA | EMPLOYEE | Table | No | Put it in In-Memory and force In-Memory scan |
| TOSSKA | DEPARTMENT | Table | No | Put it in In-Memory and force In-Memory scan |

| SQL | Cost | In-Memory Steps | Row Count | Status | Elapsed Time | Response Time | Physical Reads | Logical Reads | CPU Used By This... | Consistent Gets | Consi |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Original | 24,910 | 0 | 1049 | ✓ | 00:01:35.70 | 00:00:00.00 | 28,858 | 198,963 | 00:00:04.32 | 198,963 | |
| Auto 1 | 24,910 | 1 | 1049 | ✓ | 00:00:00.45 | 00:00:00.02 | 6 | 92,268 | 00:00:00.53 | 24,031 | |
| Auto 2 | 24,907 | 1 | | ✖ Test run was... | >00:00:05.12 | | | | | | |
| Auto 3 | 7,079 | 2 | 1049 | ✓ | 00:00:00.44 | 00:00:00.02 | 0 | 92,228 | 00:00:00.53 | 23,986 | |

TOSSKA (tosska@192.168.0.108:1521/ORCL)

## Oracle should charge their In-Memory option by In-Memory size

Actually, Oracle In-Memory database is not only beneficial to OLAP users, but it is also a good SQL performance- enhancement tool for all database users like OLTP users. In view of high pricing of Oracle In-Memory database option which limits the rapid adaptation of this new technology, it is recommended that Oracle should decide the price according to the In-Memory size. In other words, a lower size with lower price will definitely help Oracle In-Memory database to penetrate widely into all databases users, including OLTP users.

*Author: Richard To (Richard.to@tosska.com), CTO of Tosska Technologies Limited*