Case Study

Solving User Complaint For Long Running SQL

DB OPTimize
Oracle Performance Tuning & DBA Consulting

Customer was complaining **a query in production is running extremely slow**.

When checking the query, it was seen that execution is taking **57 minutes!** Almost all of the query wait time (96%) was wasted **on I/O Wait**.

Looking at the SQL text, I learned that the query performs a join between two tables, and that **a function was being called** in its where clause.

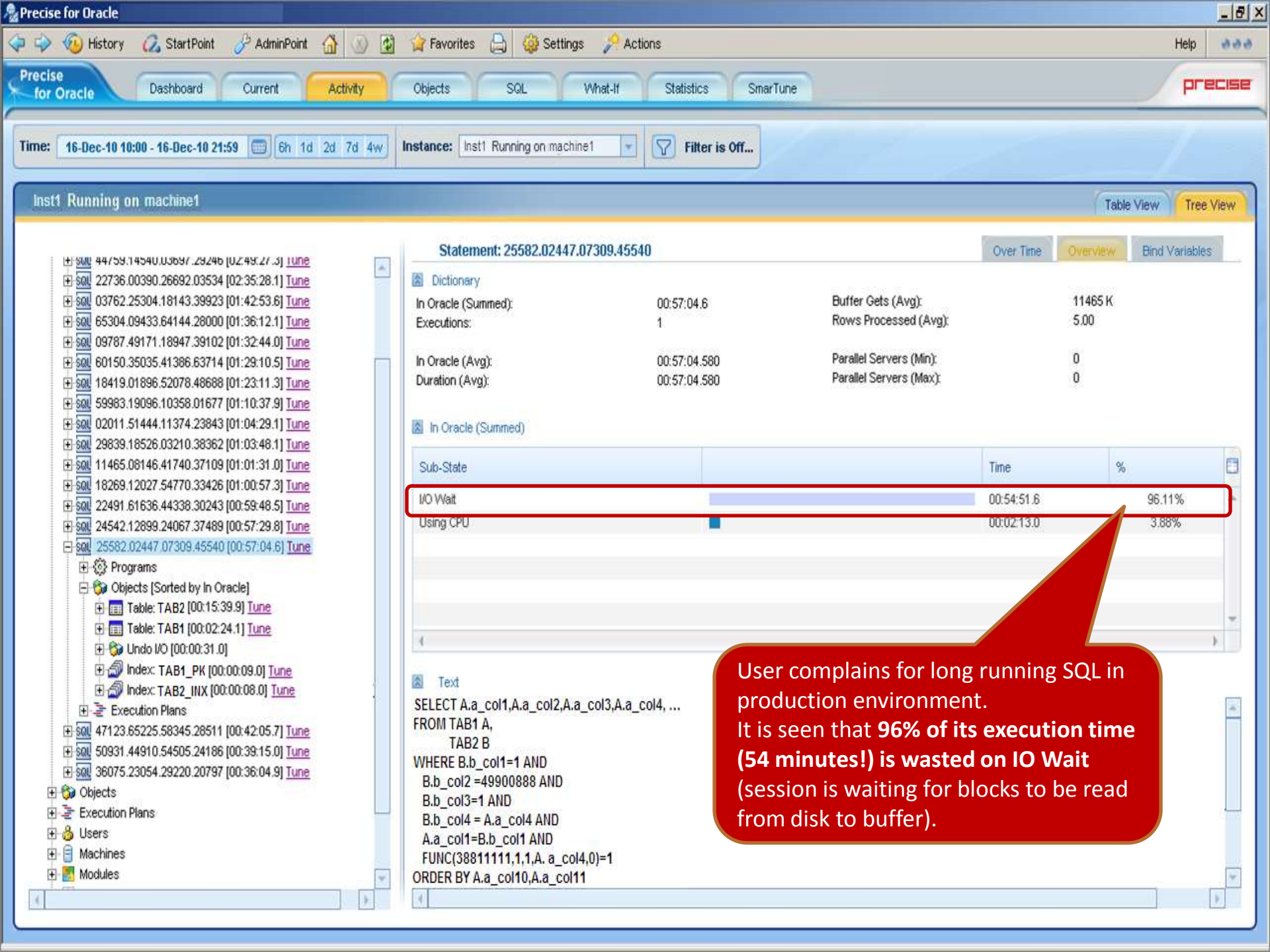
Then I have tried to identified in which part of the query the time was spent. I saw that only 31% of I/O Activity was exposed (27% on sequential I/O on one table, and another 4% on sequential I/O on the other table). That was fishy since I saw that statement was waiting 96% on I/O Wait, therefore it made me wonder where does all the rest goes?

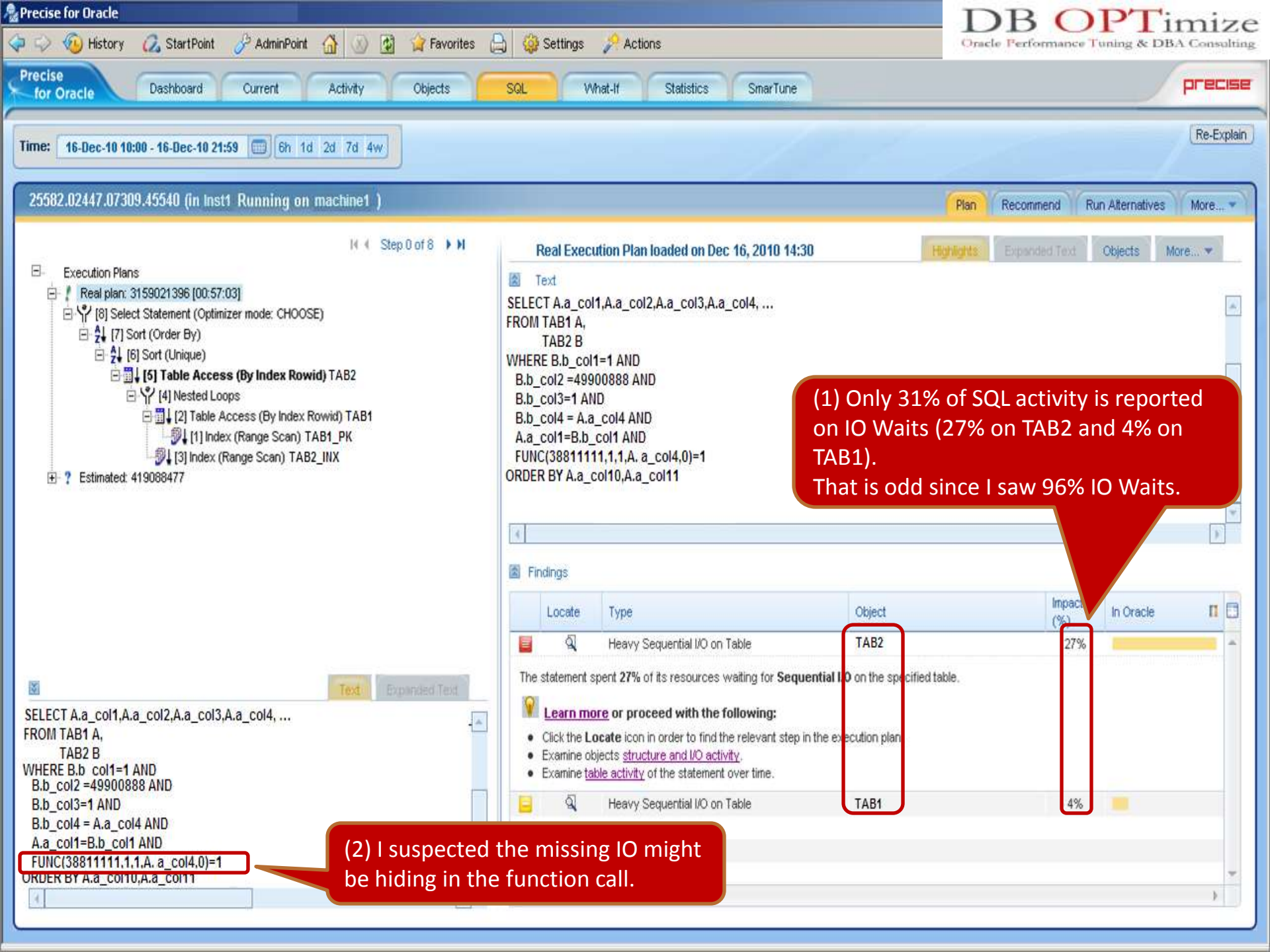
At this point I suspected time is spent on behalf of the used function.

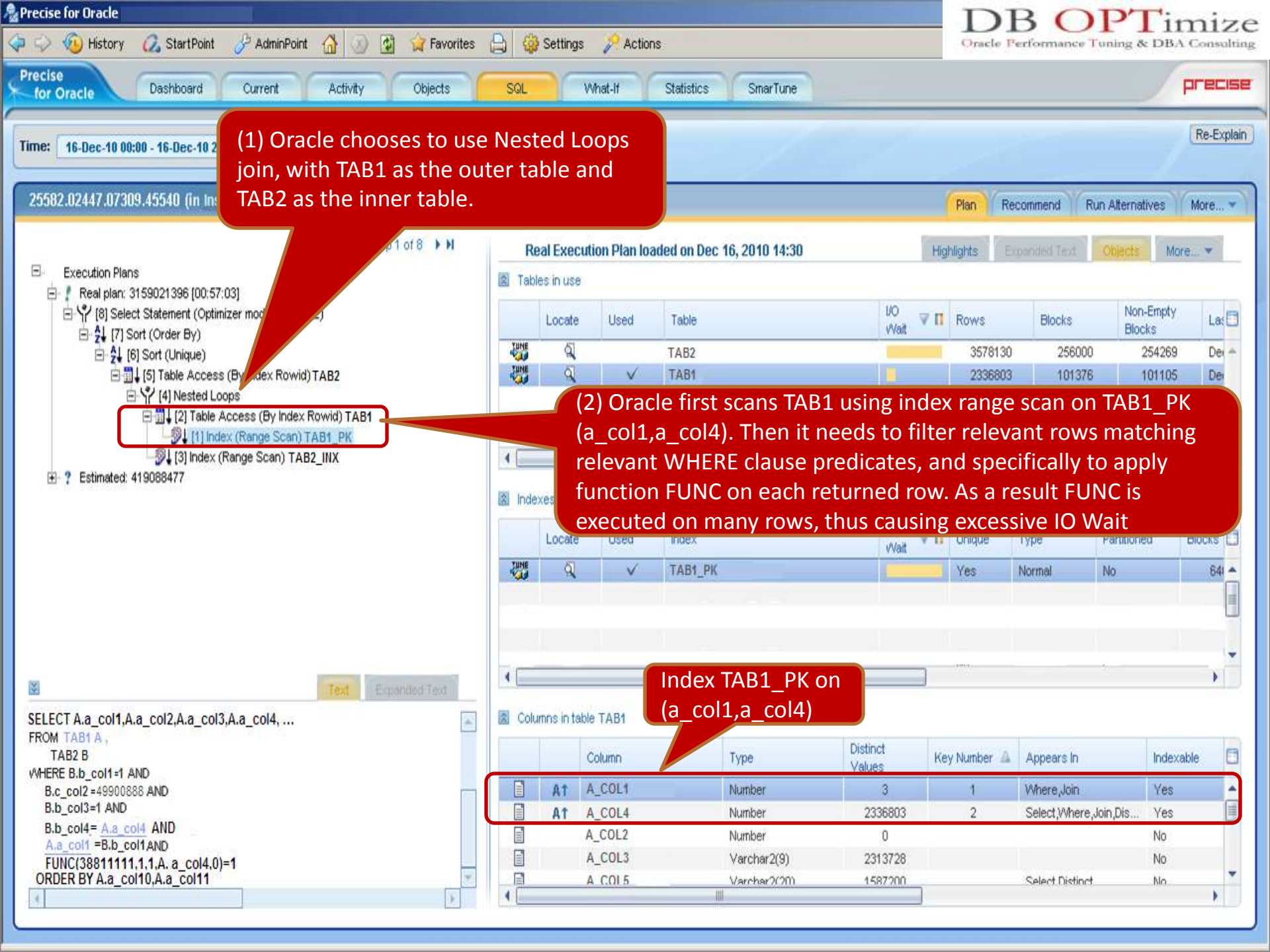
When trying to run the same query without the function, it manage to run in no time. Again, it made me think the function in responsible for the SQL poor performance.

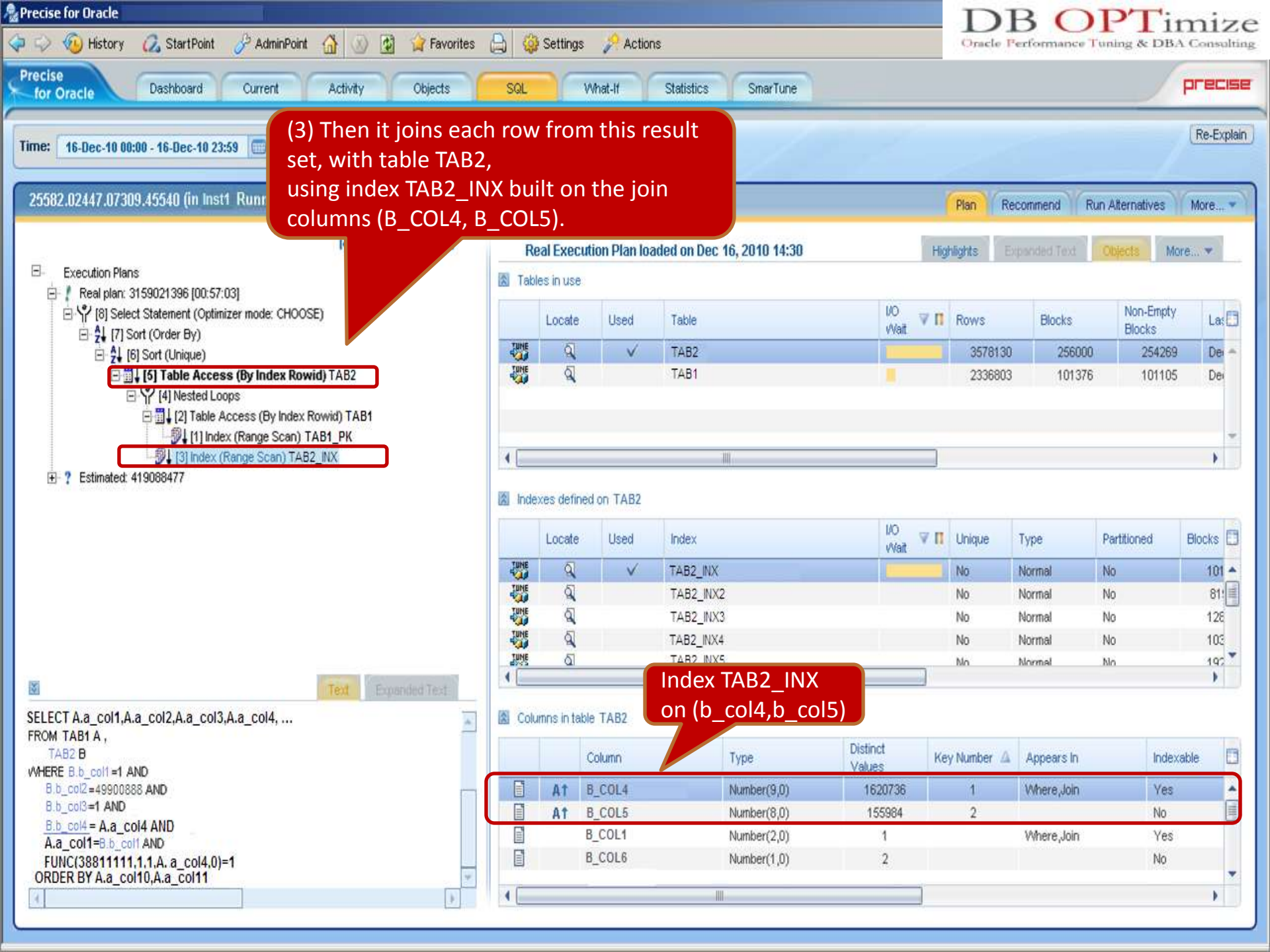
I wrapped the query in an inline view, than applied the function outside the inline view. This caused **execution time to go down from 57 minutes (!!!) to 0.03 sec (!!!).**

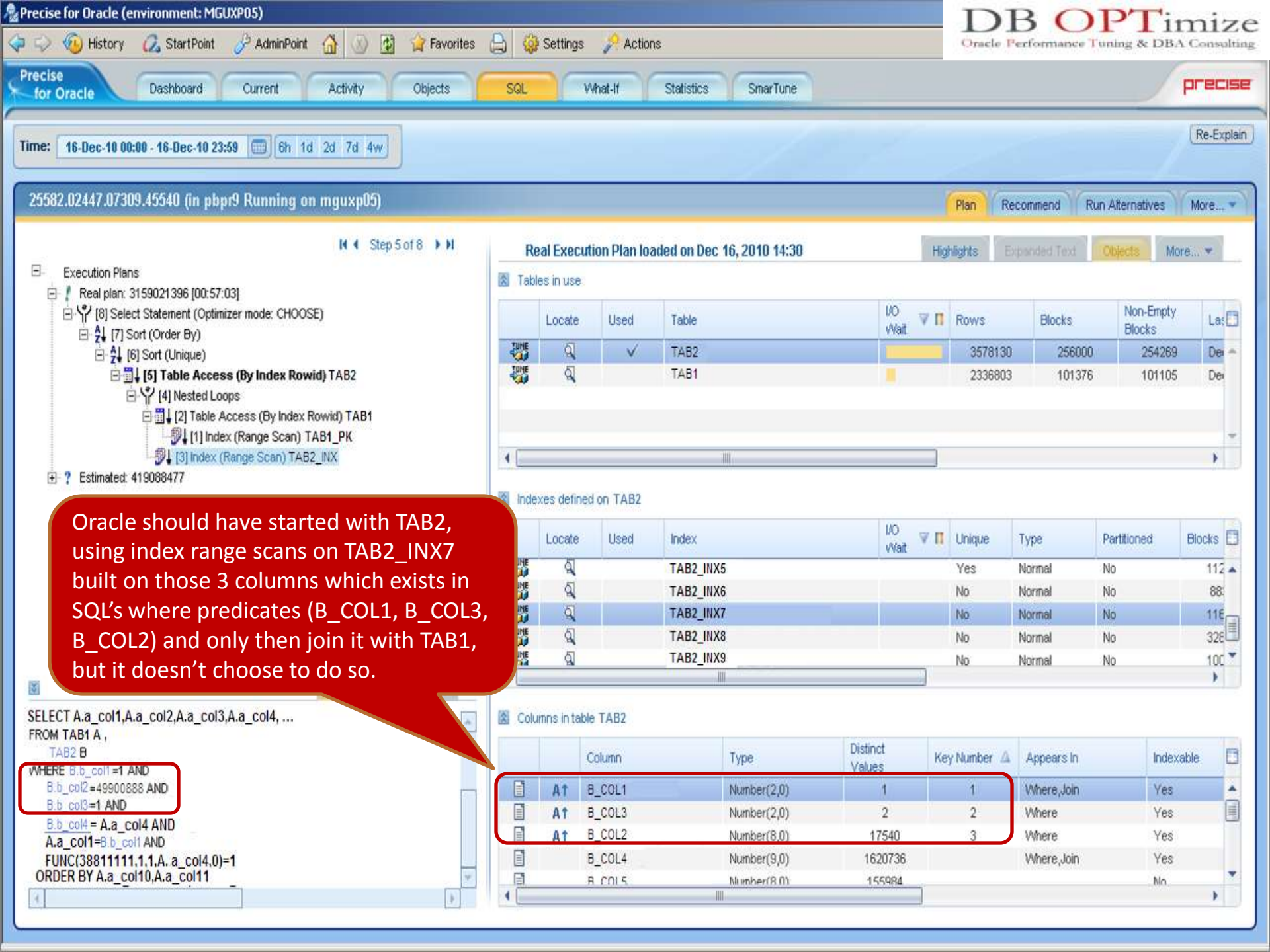
With this conclusion I went back to the developer, recommend him to change the query so function will be applied outside an inline view.


Precise for Oracle
History
StartPoint
AdminPoint
Favorites
Settings
Actions
Help
Precise for Oracle
Dashboard
Current
Activity
Objects
SQL
What-If
Statistics
SmarTune
precise
Time:
16-Dec-10 10:00 - 16-Dec-10 21:59
6h 1d 2d 7d 4w
Instance:
Inst1 Running on machine1
Filter is Off...
Inst1 Running on machine1
Table View
Tree View
22736.00390.26692.03534 [02:35:28.1] Tune
03762.25304.18143.39923 [01:42:53.6] Tune
65304.09433.64144.28000 [01:36:12.1] Tune
09787.49171.18947.39102 [01:32:44.0] Tune
60150.35035.41386.63714 [01:29:10.5] Tune
18419.01896.52078.48688 [01:23:11.3] Tune
59983.19096.10358.01677 [01:10:37.9] Tune
02011.51444.11374.23843 [01:04:29.1] Tune
29839.18526.03210.38362 [01:03:48.1] Tune
11465.08146.41740.37109 [01:01:31.0] Tune
18269.12027.54770.33426 [01:00:57.3] Tune
22491.61636.44338.30243 [00:59:48.5] Tune
24542.12899.24067.37489 [00:57:29.8] Tune
25582.02447.07309.45540 [00:57:04.6] Tune
Programs
Objects [Sorted by In Oracle]
Table: TAB2 [00:15:39.9] Tune
Table: TAB1 [00:02:24.1] Tune
Undo I/O [00:00:31.0]
Index: TAB1_PK [00:00:09.0] Tune
Index: TAB2_INX [00:00:08.0] Tune
Execution Plans
47123.65225.58345.28511 [00:42:05.7] Tune
50931.44910.54505.24186 [00:39:15.0] Tune
36075.23054.29220.20797 [00:36:04.9] Tune
Objects
Execution Plans
Users
Machines
Modules
Statement: 25582.02447.07309.45540
Over Time
Overview
Bind Variables
Dictionary
In Oracle (Summed): 00:57:04.6
Executions: 1
In Oracle (Avg): 00:57:04.580
Duration (Avg): 00:57:04.580
Buffer Gets (Avg): 11465 K
Rows Processed (Avg): 5.00
Parallel Servers (Min): 0
Parallel Servers (Max): 0
In Oracle (Summed)
Sub-State
Time
%
I/O Wait
00:54:51.6
96.11%
Using CPU
00:02:13.0
3.88%
Text
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, ...
FROM TAB1 A,
TAB2 B
WHERE B.b_col1=1 AND
B.b_col2 =49900888 AND
B.b_col3=1 AND
B.b_col4 = A.a_col4 AND
A.a_col1=B.b_col1 AND
FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11
User complains for long running SQL in production environment.
It is seen that 96% of its execution time (54 minutes!) is wasted on IO Wait (session is waiting for blocks to be read from disk to buffer).


Precise for Oracle
History
StartPoint
AdminPoint
Favorites
Settings
Actions
DB OPTimize
Oracle Performance Tuning & DBA Consulting
Precise for Oracle
Dashboard
Current
Activity
Objects
SQL
What-If
Statistics
SmarTune
precise
Time:
16-Dec-10 10:00 - 16-Dec-10 21:59
6h 1d 2d 7d 4w
Re-Explain
25582.02447.07309.45540 (in Inst1 Running on machine1)
Plan
Recommend
Run Alternatives
More...
Step 0 of 8
Execution Plans
Real plan: 3159021396 [00:57:03]
[8] Select Statement (Optimizer mode: CHOOSE)
[7] Sort (Order By)
[6] Sort (Unique)
[5] Table Access (By Index Rowid) TAB2
[4] Nested Loops
[2] Table Access (By Index Rowid) TAB1
[1] Index (Range Scan) TAB1_PK
[3] Index (Range Scan) TAB2_INX
Estimated: 419088477
Real Execution Plan loaded on Dec 16, 2010 14:30
Highlights
Expanded Text
Objects
More...
Text
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, ...
FROM TAB1 A,
TAB2 B
WHERE B.b_col1=1 AND
B.b_col2 =49900888 AND
B.b_col3=1 AND
B.b_col4 = A.a_col4 AND
A.a_col1=B.b_col1 AND
FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11
(1) Only 31% of SQL activity is reported on IO Waits (27% on TAB2 and 4% on TAB1).
That is odd since I saw 96% IO Waits.
Findings
Locate
Type
Object
Impact (%)
In Oracle
Heavy Sequential I/O on Table
TAB2
27%
The statement spent 27% of its resources waiting for Sequential I/O on the specified table.
Learn more or proceed with the following:
Click the Locate icon in order to find the relevant step in the execution plan
Examine objects structure and I/O activity.
Examine table activity of the statement over time.
Heavy Sequential I/O on Table
TAB1
4%
Text
Expanded Text
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, ...
FROM TAB1 A,
TAB2 B
WHERE B.b_col1=1 AND
B.b_col2 =49900888 AND
B.b_col3=1 AND
B.b_col4 = A.a_col4 AND
A.a_col1=B.b_col1 AND
FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11
(2) I suspected the missing IO might be hiding in the function call.


Precise for Oracle
History
StartPoint
AdminPoint
Favorites
Settings
Actions
DB OPTimize
Oracle Performance Tuning & DBA Consulting
Precise for Oracle
Dashboard
Current
Activity
Objects
SQL
What-If
Statistics
SmarTune
precise
Time:
16-Dec-10 00:00 - 16-Dec-10 2
Re-Explain
25582.02447.07309.45540 (in Ins
Plan
Recommend
Run Alternatives
More...
(1) Oracle chooses to use Nested Loops join, with TAB1 as the outer table and TAB2 as the inner table.
Execution Plans
Real plan: 3159021396 [00:57:03]
[8] Select Statement (Optimizer mod
[7] Sort (Order By)
[6] Sort (Unique)
[5] Table Access (By Index Rowid) TAB2
[4] Nested Loops
[2] Table Access (By Index Rowid) TAB1
[1] Index (Range Scan) TAB1_PK
[3] Index (Range Scan) TAB2_INX
Estimated: 419088477
Real Execution Plan loaded on Dec 16, 2010 14:30
Highlights
Expanded Text
Objects
More...
Tables in use
Locate
Used
Table
I/O Wait
Rows
Blocks
Non-Empty Blocks
TAB2
3578130
256000
254269
TAB1
2336803
101376
101105
(2) Oracle first scans TAB1 using index range scan on TAB1_PK (a_col1,a_col4). Then it needs to filter relevant rows matching relevant WHERE clause predicates, and specifically to apply function FUNC on each returned row. As a result FUNC is executed on many rows, thus causing excessive IO Wait
Indexes
Index
Unique
Type
Partitioned
Blocks
TAB1_PK
Yes
Normal
No
Text
Expanded Text
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, ...
FROM TAB1 A ,
TAB2 B
WHERE B.b_col1=1 AND
B.c_col2 =49900888 AND
B.b_col3=1 AND
B.b_col4= A.a_col4 AND
A.a_col1 =B.b_col1 AND
FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11
Index TAB1_PK on (a_col1,a_col4)
Columns in table TAB1
Column
Type
Distinct Values
Key Number
Appears In
Indexable
A_COL1
Number
3
1
Where,Join
Yes
A_COL4
Number
2336803
2
Select,Where,Join,Dis...
Yes
A_COL2
Number
0
No
A_COL3
Varchar2(9)
2313728
No
A_COL5
Varchar2(20)
1587200
Select,Distinct
No


Precise for Oracle
DB OPTimize
Oracle Performance Tuning & DBA Consulting
History
StartPoint
AdminPoint
Favorites
Settings
Actions
Precise for Oracle
Dashboard
Current
Activity
Objects
SQL
What-If
Statistics
SmarTune
precise
Time:
16-Dec-10 00:00 - 16-Dec-10 23:59
Re-Explain
(3) Then it joins each row from this result set, with table TAB2, using index TAB2_INX built on the join columns (B_COL4, B_COL5).
25582.02447.07309.45540 (in Inst1 Runn
Plan
Recommend
Run Alternatives
More...
Execution Plans
Real plan: 3159021396 [00:57:03]
[8] Select Statement (Optimizer mode: CHOOSE)
[7] Sort (Order By)
[6] Sort (Unique)
[5] Table Access (By Index Rowid) TAB2
[4] Nested Loops
[2] Table Access (By Index Rowid) TAB1
[1] Index (Range Scan) TAB1_PK
[3] Index (Range Scan) TAB2_INX
Estimated: 419088477
Real Execution Plan loaded on Dec 16, 2010 14:30
Highlights
Expanded Text
Objects
More...
Tables in use
Locate
Used
Table
I/O Wait
Rows
Blocks
Non-Empty Blocks
TAB2
3578130
256000
254269
TAB1
2336803
101376
101105
Indexes defined on TAB2
Index
Unique
Type
Partitioned
Blocks
TAB2_INX
No
Normal
No
TAB2_INX2
TAB2_INX3
TAB2_INX4
Text
Expanded Text
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, ...
FROM TAB1 A ,
TAB2 B
WHERE B.b_col1 =1 AND
B.b_col2 =49900888 AND
B.b_col3=1 AND
B.b_col4 = A.a_col4 AND
A.a_col1=B.b_col1 AND
FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11
Index TAB2_INX on (b_col4,b_col5)
Columns in table TAB2
Column
Type
Distinct Values
Key Number
Appears In
Indexable
B_COL4
Number(9,0)
1620736
1
Where,Join
Yes
B_COL5
Number(8,0)
155984
2
No
B_COL1
Number(2,0)
Where,Join
B_COL6
Number(1,0)


Precise for Oracle (environment: MGUXP05)
History
StartPoint
AdminPoint
Favorites
Settings
Actions
DB OPTimize
Oracle Performance Tuning & DBA Consulting
Precise for Oracle
Dashboard
Current
Activity
Objects
SQL
What-If
Statistics
SmarTune
precise
Time:
16-Dec-10 00:00 - 16-Dec-10 23:59
6h 1d 2d 7d 4w
Re-Explain
25582.02447.07309.45540 (in pbpr9 Running on mguxp05)
Plan
Recommend
Run Alternatives
More...
Step 5 of 8
Execution Plans
Real plan: 3159021396 [00:57:03]
[8] Select Statement (Optimizer mode: CHOOSE)
[7] Sort (Order By)
[6] Sort (Unique)
[5] Table Access (By Index Rowid) TAB2
[4] Nested Loops
[2] Table Access (By Index Rowid) TAB1
[1] Index (Range Scan) TAB1_PK
[3] Index (Range Scan) TAB2_INX
Estimated: 419088477
Real Execution Plan loaded on Dec 16, 2010 14:30
Highlights
Expanded Text
Objects
More...
Tables in use
Locate
Used
Table
I/O Wait
Rows
Blocks
Non-Empty Blocks
TAB2
3578130
256000
254269
TAB1
2336803
101376
101105
Indexes defined on TAB2
Index
Unique
Type
Partitioned
Blocks
TAB2_INX5
Yes
Normal
No
TAB2_INX6
No
Normal
No
TAB2_INX7
No
Normal
No
TAB2_INX8
No
Normal
No
TAB2_INX9
No
Normal
No
Oracle should have started with TAB2, using index range scans on TAB2_INX7 built on those 3 columns which exists in SQL's where predicates (B_COL1, B_COL3, B_COL2) and only then join it with TAB1, but it doesn't choose to do so.
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, ...
FROM TAB1 A ,
TAB2 B
WHERE B.b_col1 =1 AND
B.b_col2 =49900888 AND
B.b_col3 =1 AND
B.b_col4 = A.a_col4 AND
A.a_col1=B.b_col1 AND
FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11
Columns in table TAB2
Column
Type
Distinct Values
Key Number
Appears In
Indexable
B_COL1
Number(2,0)
1
1
Where,Join
Yes
B_COL3
Number(2,0)
2
2
Where
Yes
B_COL2
Number(8,0)
17540
3
Where
Yes
B_COL4
Number(9,0)
1620736
Where,Join
Yes

In the original query, the function FUNC is used in query's WHERE clause. Since Oracle choose to use NL join with TAB1 as the outer table, FUNC needs to be apply on every row returned from TAB1, before joining it with TAB2.

This yields poor query performance and unnecessary high I/O Waits.

```
SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, …
    FROM TAB1 A
        TAB2 B
    WHERE B.b_col1=1 AND
        B.b_col2 =49900888 AND
        B.b_col3=1 AND
        B.b_col4 = A.a_col4 AND
        A.a_col1=B.b_col1 AND
        FUNC(38811111,1,1,A.a_col4,0)=1
    ORDER BY A.a_col10,A.a_col11;
```

I needed to make Oracle apply the function FUNC only when necessary, meaning on all relevant rows that answer everything else but the function. I figured it should yield a small result set, thus function FUNC will be called only a small amount of times.

Wrapping the query with an inline view, **and applying the function on the inline** view gave me exactly that.

Execution time went down from 57 minutes to **0.03 sec!** What a change

```
SELECT * FROM ( SELECT A.a_col1,A.a_col2,A.a_col3,A.a_col4, …
    FROM TAB1 A
        TAB2 B
    WHERE B.b_col1=1 AND
        B.b_col2 =49900888 AND
        B.b_col3=1 AND
        B.b_col4 = A.a_col4 AND
        A.a_col1=B.b_col1 AND
)
WHERE  FUNC(38811111,1,1,A. a_col4,0)=1
ORDER BY A.a_col10,A.a_col11 ;
```

Oracle chooses NL, but now the outer table is TAB2 using index range scan on TAB2_INX7 (as wanted), then joins it with TAB1, then apply the function on the result set.

Elapsed: 00:00:00.03

Execution Plan

```
----------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                                | Rows  | Bytes | Cost  |
----------------------------------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT               |                                     |  3449 |   579K |  1019 |
|*  1 |  VIEW                          |                                     |  3449 |   579K |  1019 |
|   2 |   SORT UNIQUE                  |                                     |  3449 |   350K |  1019 |
|   3 |    NESTED LOOPS                |                                     |  3449 |   350K |   920 |
|   4 |     TABLE ACCESS BY INDEX ROWID| TAB2                                |  3449 | 75878 |   231 |
|*  5 |      INDEX RANGE SCAN          | TAB2_INX7                           |  3449 |       |     2 |
|   6 |     TABLE ACCESS BY INDEX ROWID| TAB1                                |     1 |    82 |     1 |
|*  7 |      INDEX UNIQUE SCAN         | TAB1_PK                             |     1 |       |     1 |

----------------------------------------------------------------------------------------------------------------
```

DB OPTimize

Oracle **P**erformance **T**uning & DBA Consulting

www.dboptimize.co.il

merav@dboptimize.co.il