SQL Metrics

- DO ANALYZE FASTER -

AWR ANALYZER

For ORACLE database

Abstrakt

SQL Developer plug-in which rapidly reduce time for analysis in user friendly GUI

SQL Metrics
awr@sqlmetrics.com

AWR ANALYZER

AWR@SQLMETRICS.COM

+421 901 123 456

BRATISLAVA

INTRODUCTION

Dear Customer,

Following pages describe the product of SQL Metrics company which was designed to make the work with the Oracle® database systems much easier.

.

AWR Analyzer is distributed as a plug-in for the Oracle® SQL Developer tool.

It provides users with a quick SQL analysis and problems detection in attractive GUI without requiring an installation of any database procedures, triggers or functions. It is fully operational right after the installation.

Plug-in also contains the basic SPM (SQL PLAN MANAGEMENT) functionality.

Analytic tool AWR Analyzer is developed for Oracle® databases of version 11.2 or higher and both Standard and Enterprise editions (AWR or STATSPACK).

Regards,

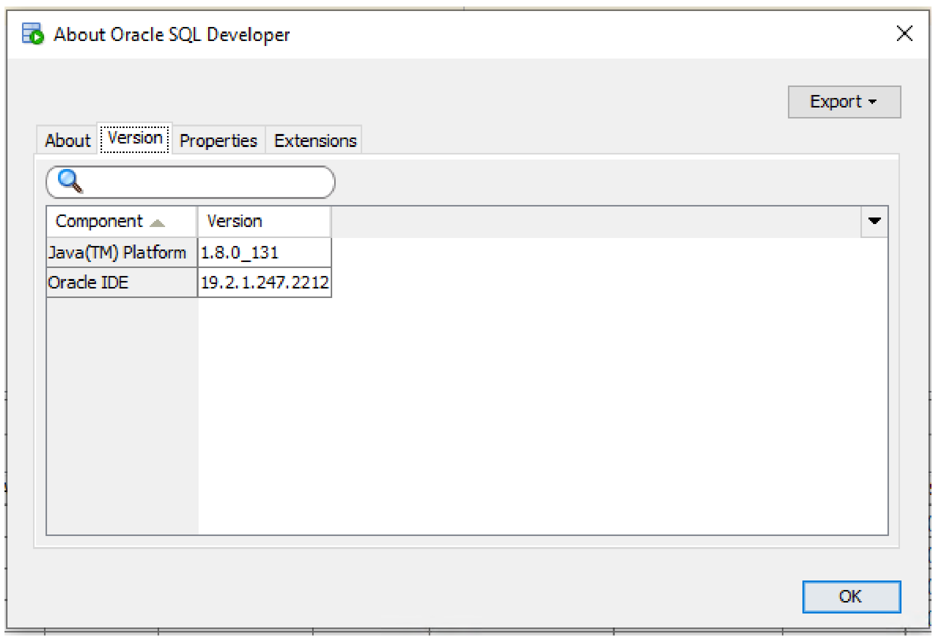
team SQL Metrics

TABLE OF CONTENTS

INSTALLATION REQUIREMENTS

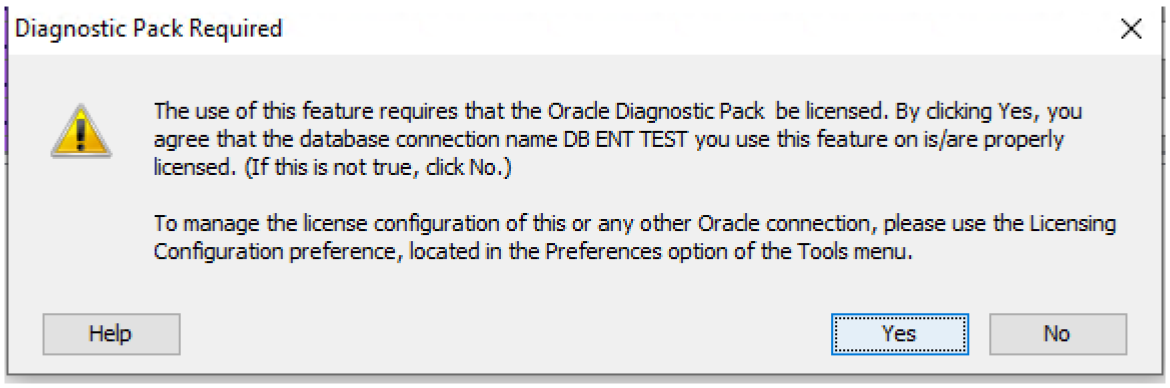
Minimum requirements for the installation of AWR Analyzer:

- Operating system: Mac OS®, Windows® a Linux®
- SQL Developer: version 19.2.* or higher



Component	Version
Java(TM) Platform	1.8.0_131
Oracle IDE	19.2.1.247.2212

Note: Plug-in doesn't provide an Oracle license for using AWR (Automatic Workload Repository).
To use AWR functionality it is necessary to own the Oracle Diagnostic Pack license.
You can find license fees in official Oracle price list or your Oracle distributer.



The use of this feature requires that the Oracle Diagnostic Pack be licensed. By clicking Yes, you agree that the database connection name DB ENT TEST you use this feature on is/are properly licensed. (If this is not true, click No.)

To manage the license configuration of this or any other Oracle connection, please use the Licensing Configuration preference, located in the Preferences option of the Tools menu.

STATSPACK service is a predecessor of the AWR.
In solutions using Standard Edition (SE) or 12c Standard Edition (SE2) provides STACKPACK 95% of data which provides AWR and is 100% free for installation & use.

AWR Analyzer works with both Oracle RAC and non-RAC systems from the version 11.2.

For the correct functionality of the plug-in is necessary to have installed a STATSPACK or AWR package and grant User Rights for the objects.

it is possible to provide plug-in for older versions after the consultation. Minimum number of licenses is 20.

INTRODUCTION

The standard STATSPACK/AWR analyzer provided by Oracle corporation as a component of the Oracle database has to be executed as a script and only provides the output in the text form which is complicated to read. Also it is not possible to see SQL execution plan without creating new output. Text output is divided to several sections and to find the links between data user has to scroll between particular sections.

Example of the standard AWR analyzer output after execution:
For AWR:

```
SQL> @$ORACLE_HOME/rdbms/admin/awrrpt.sql
```

For STATSPACK:

```
SQL>  @?/rdbms/admin/spreport
```

```
     CPU                  CPU per             Elapsd                    Old
  Time (s)    Executions  Exec (s)  %Total   Time (s)    Buffer Gets  Hash Value
---------- ------------ ---------- ------ ---------- --------------- ----------
Module: DBMS_SCHEDULER
MERGE /*+ dynamic_sampling(4) dynamic_sampling_est_cdn  */ INTO
SYS.MON_MODS_ALL$ MM USING ( SELECT OBJ#, SUM(FLGS) FLGS FROM (
SELECT OBJ#, :B1 FLGS FROM ( SELECT DISTINCT TAB.OBJ# FROM (SELE
CT T.OBJ# OBJ#, T.OBJ# BO#, T.ANALYZETIME FROM SYS.TAB$ T WHERE

      9.40          72        0.13     3.5       10.94        2,992,477    62099832
Module: DBMS_SCHEDULER
INSERT INTO WRI$_HEATMAP_TOPN_DEP2 SELECT TABLE_NAME, OWNER, 'LO
BS', SUM(SEGMENT_COUNT), SUM(OBJECT_SIZE) FROM DBA_LOBS, WRI$_HE
ATMAP_TOPN_DEP1 WHERE WRI$_HEATMAP_TOPN_DEP1.OBJECT_NAME = DBA_L
OBS.SEGMENT_NAME AND WRI$_HEATMAP_TOPN_DEP1.OBJECT_OWNER = DBA_L
```

WHY USE AWR ANALYZER

Analyze AWR plug-in represents data in much more user friendly GUI:

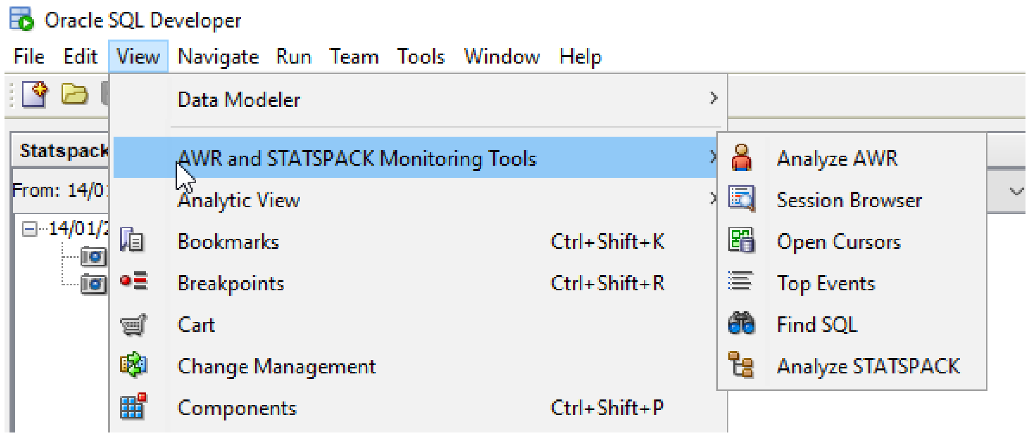
- Tabular data output
- Filter data by SQL Module
- Details of the SQL query accessible on one click with possibility to open details in the new tab on double-click.
- Display of SQL statement in formatted form
- SQL execution plan
- Exporting of the selected data into pdf or excel file or into clipboard
- Possibility to define quantity of displayed data

In addition it provides the data and the tools which are not components of the original AWR:

- #Of Exec. Plans - number of the distinct execution plans used for the particular SQL statement
- Binding Variables – logical true/false value which represents whether given SQL has stored binding variables during the run
- SQL Plan Management - tool for maintaining the execution plans (detailed description of SPM functionality is available on the official Oracle web-page)

ACCESS TO AWR ANALYZER

Access to AWR Analyzer plug-in is located in SQL Developer "View" menu under "AWR and STATSPACK Monitoring Tools" menu item



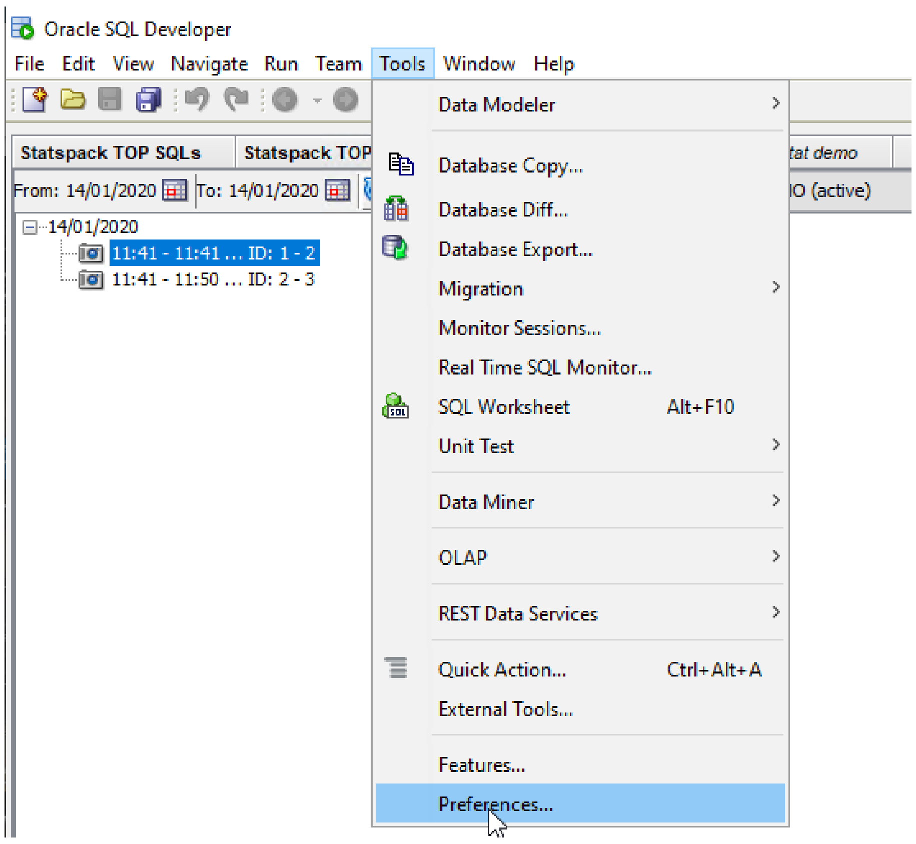
USERS PREFERENCES

Plug-in allows user to make his own individual setting:

- Colors and the number of displayed rows
- Language version

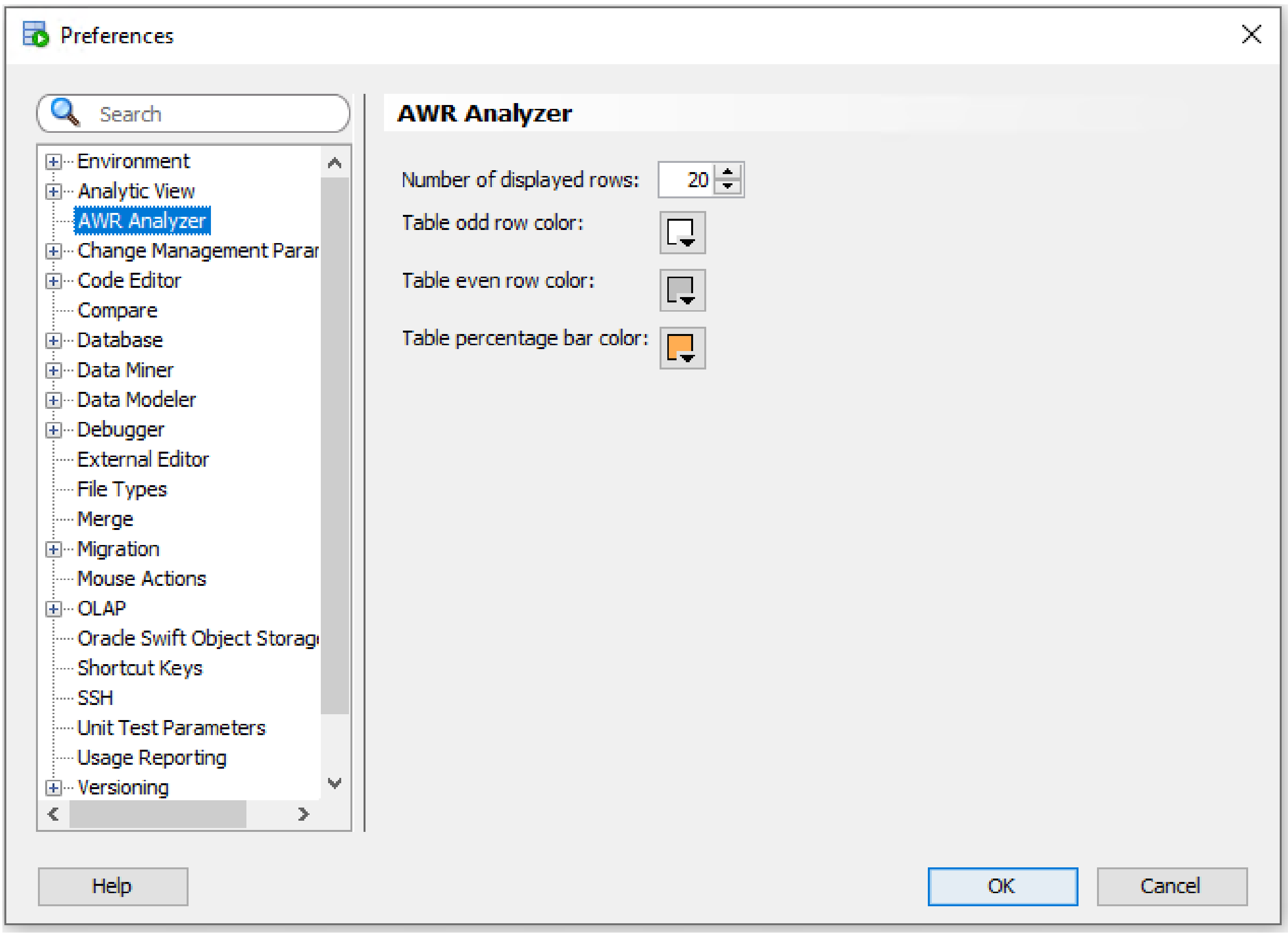
PANEL NASTAVENÍ

To open the preferences panel click on „Tools“ -> “Preferences“



In Preferences panel you can adjust:

- Number of displayed rows for statistics tables
- Table odd rows color
- Table even rows color
- Table percentage bar color



AWR Analyzer

Number of displayed rows: 20

Table odd row color:

Table even row color:

Table percentage bar color:

LANGUAGE VERSIONS

Language selection is managed by SQL Developer itself.
On the Internet there are several manuals how to change the language in SQL Developer. For example on this link.

The plug-in is originally delivered in two language mutations:

- English
- Spanish

Some other language mutations may be added in the future. Or can be added on demand after previous agreement.

ANALYZE STATSPACK / ANALYZE AWR

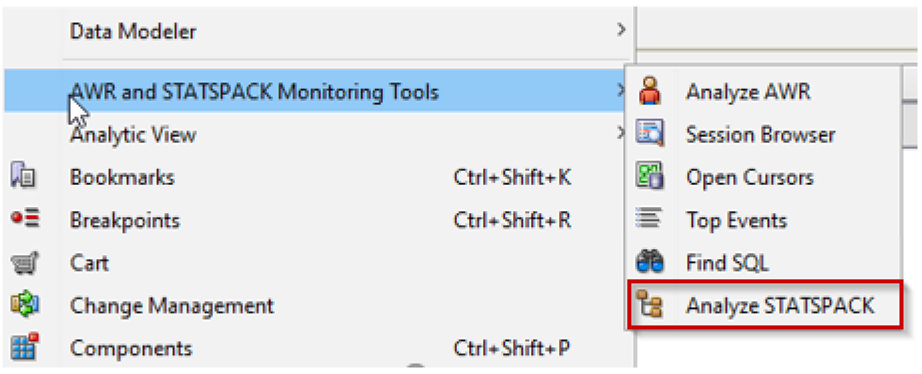
(Panels are almost identical. Differences are mentioned in separate section.)

Access to both tools is from the menu "AWR and STATSPACK Monitoring Tools" and:

“Analyze STATSPACK”

or

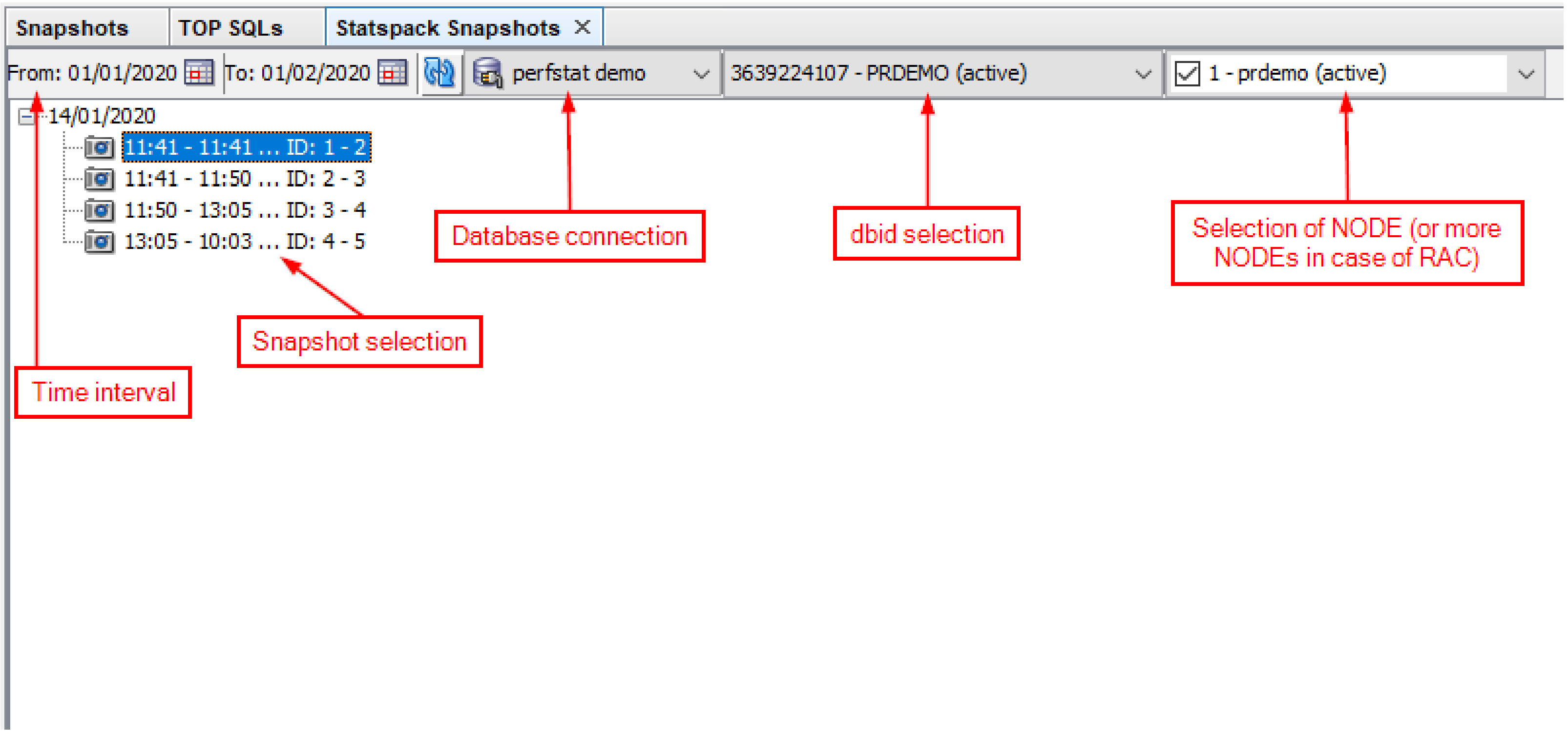
“Analyze AWR”



Panel “Statspack Snapshots” is opened with basic criteria for selection of:

- time interval for the monitored SQL events (it is possible to select more consecutive intervals)
- database server
- database
- “node” in the selected database.

Selection criteria allow selecting from more databases and nodes. This is possible when there are some imported STATPACKs from other databases. For example from production database which can be physically located on different place or if user from any reason just doesn’t have an access to it.
AWR Analyzer plug-in allows to work with imported data from external sources.


Snapshots
TOP SQLs
Statspack Snapshots
From: 01/01/2020
To: 01/02/2020
perfstat demo
3639224107 - PRDEMO (active)
☑ 1 - prdemo (active)
14/01/2020
11:41 - 11:41 ... ID: 1 - 2
11:41 - 11:50 ... ID: 2 - 3
11:50 - 13:05 ... ID: 3 - 4
13:05 - 10:03 ... ID: 4 - 5

Database connection

dbid selection

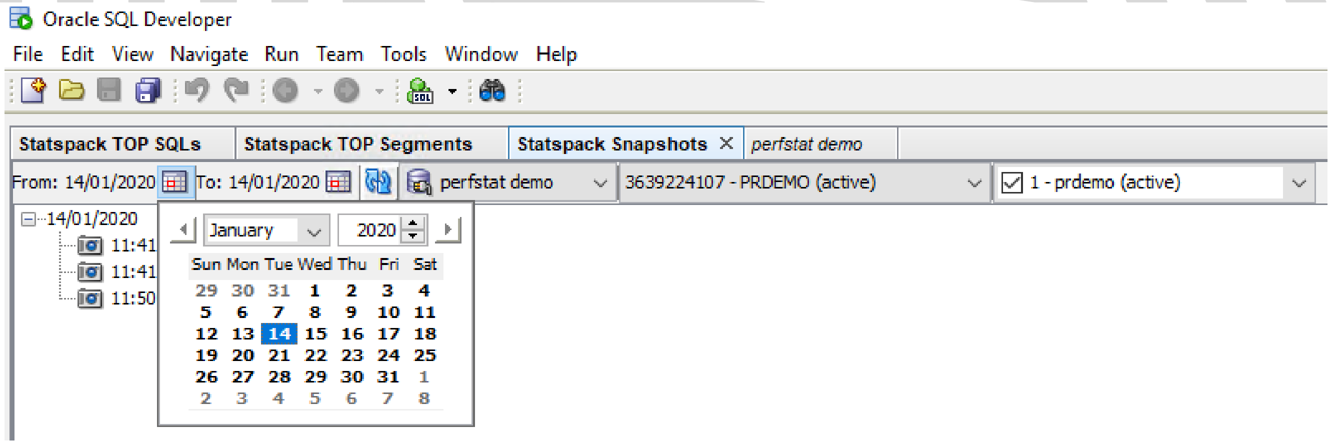
Selection of NODE (or more NODEs in case of RAC)

Snapshot selection

Time interval

TIME INTERVAL SELECTION

User can select any time interval



From: 14/01/2020 To: 14/01/2020 perfstat demo 3639224107 - PRDEMO (active) ☑ 1 - prdemo (active)

14/01/2020
11:41
11:41
11:50

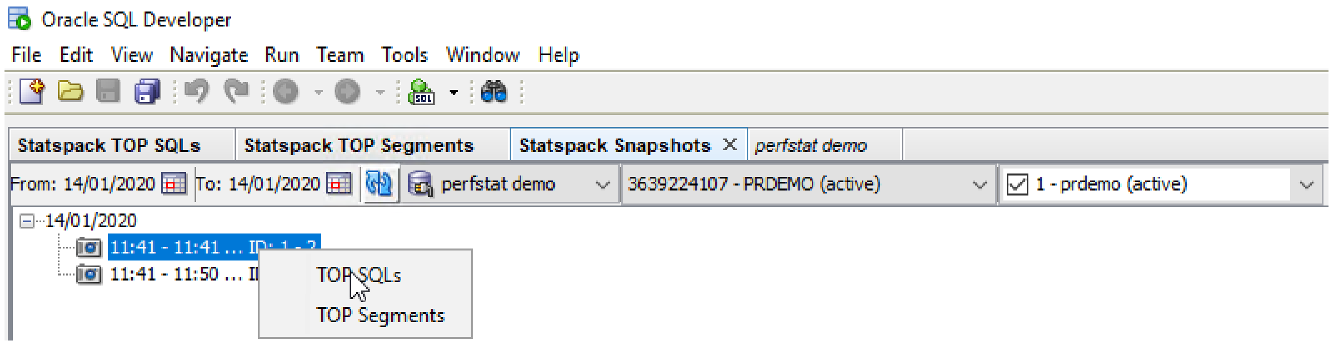
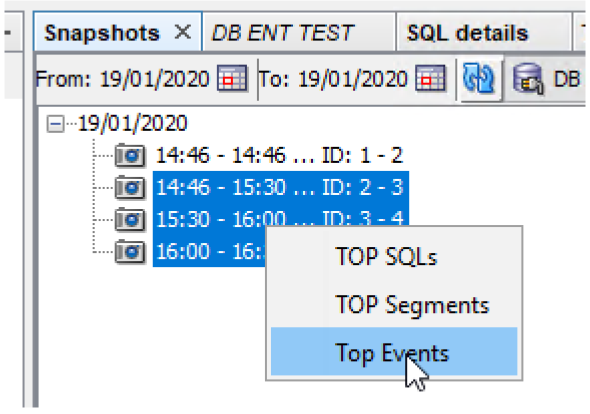
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Click on the calendar icons opens pop-up for selection of the From-To dates for required time interval.

All snapshots from the selected time interval will be listed grouped by days and ordered by date.

CONSULTATION SELECTION

Selection of basic consultations for the selected snapshot is accessible after right-click in the context menu.

Analyze STATSPACK	Analyze AWR
- TOP SQLs - TOP Segments	- TOP SQLs - TOP Segments - Top Events
	

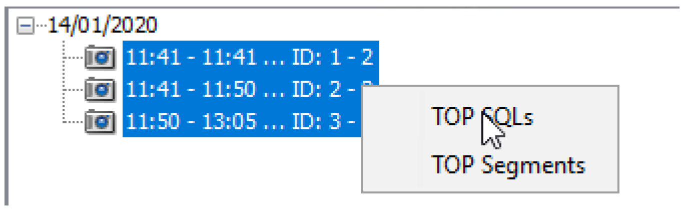
From: 14/01/2020 To: 14/01/2020 perfstat demo 3639224107 - PRDEMO (active) ☑ 1 - prdemo (active)

14/01/2020
11:41 - 11:41 ... ID: 1 - 2
11:41 - 11:50 ... II

From: 19/01/2020 To: 19/01/2020

19/01/2020
14:46 - 14:46 ... ID: 1 - 2
14:46 - 15:30 ... ID: 2 - 3
15:30 - 16:00 ... ID: 3 - 4
16:00 - 16:

Note: For selecting more snapshot (also from different dates) press SHIFT and select rows with left-click. Right-click opens context menu for selection of consultations.



14/01/2020
11:41 - 11:41 ... ID: 1 - 2
11:41 - 11:50 ... ID: 2 -
11:50 - 13:05 ... ID: 3 -

TOP SQLS

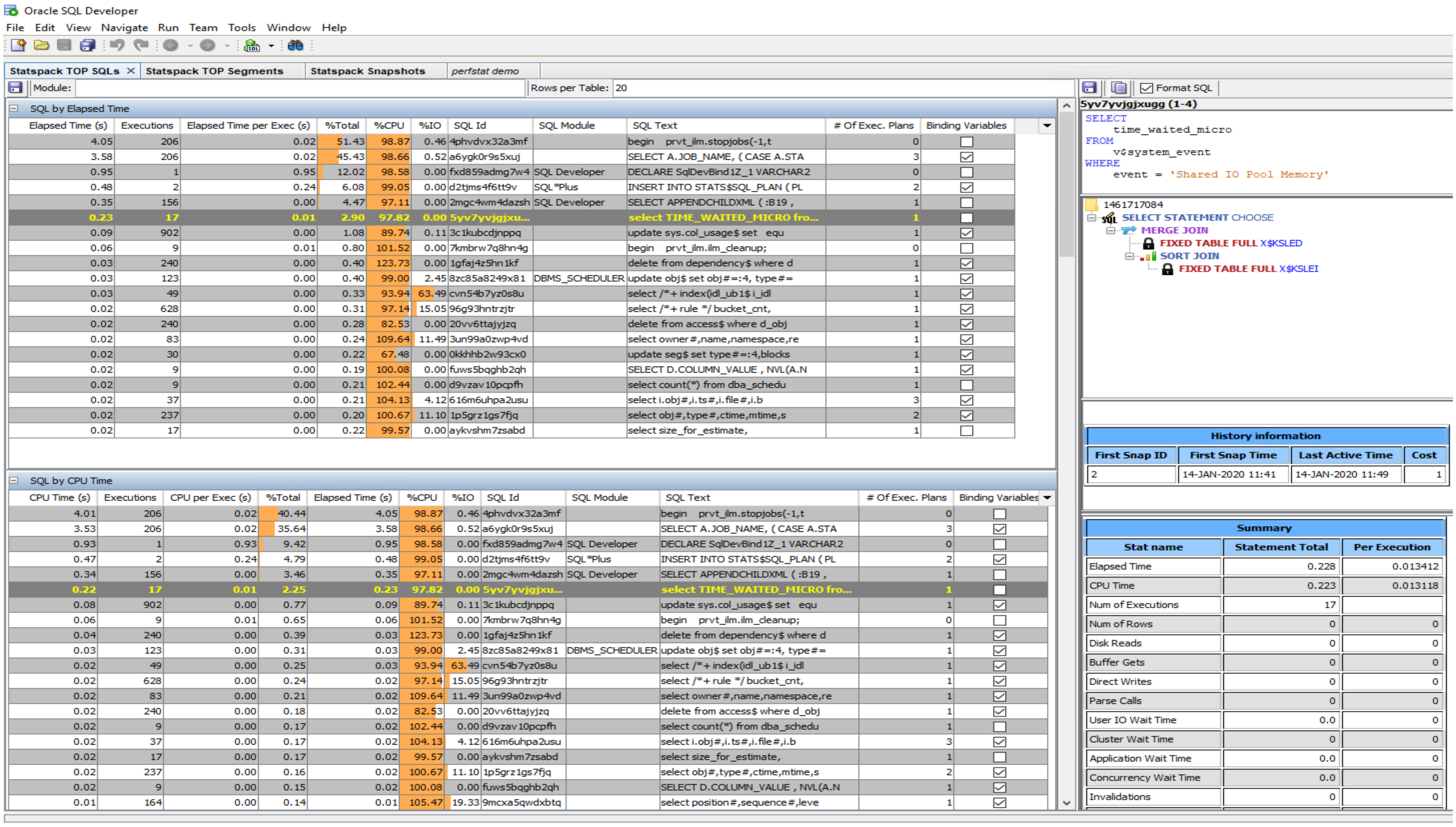
Consultation TOP SQLs shows in tabular form data for following statistics:

- SQL ordered by Elapsed Time
- SQL ordered by CPU Time
- SQL ordered by User I/O Wait Time
- SQL ordered by Gets
- SQL ordered by Reads
- SQL ordered by Physical Reads (UnOptimized)
- SQL ordered by Executions
- SQL ordered by Parse Calls
- SQL ordered by Sharable Memory
- SQL ordered by Version Count
- SQL ordered by Cluster Wait Time - available only for RAC environment

Panel TOP SQLs depending on the number of selected instances automatically opens as:

- Single instance view – for one selected instance
- RAC view - for more selected instances

SINGLE INSTANCE VIEW



Module:

Rows per Table: 20

SQL by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.05	206	0.02	51.43	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	□
3.58	206	0.02	45.43	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	☑
0.95	1	0.95	12.02	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	□
0.48	2	0.24	6.08	99.05	0.00	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	☑
0.35	156	0.00	4.47	97.11	0.00	2mgc4wm4dazsh	SQL Developer	SELECT APPENDCHILDXML (:B19 ,	1	□
0.23	17	0.01	2.90	97.82	0.00	5yv7yvjgjxu...		select TIME_WAITED_MICRO fro...	1	□
0.09	902	0.00	1.08	89.74	0.11	3c1kubcdjnppq		update sys.col_usage$ set equ	1	☑
0.06	9	0.01	0.80	101.52	0.00	7kmbrw7q8hn4g		begin prvt_ilm.ilm_cleanup;	0	□
0.03	240	0.00	0.40	123.73	0.00	1gfaj4z5hn1kf		delete from dependency$ where d	1	☑
0.03	123	0.00	0.40	99.00	2.45	8zc85a8249x81	DBMS_SCHEDULER	update obj$ set obj#=:4, type#=	1	☑
0.03	49	0.00	0.33	93.94	63.49	cvn54b7yz0s8u		select /*+ index(idl_ub1$ i_idl	1	☑
0.02	628	0.00	0.31	97.14	15.05	96g93hntrzjtr		select /*+ rule */ bucket_cnt,	1	☑
0.02	240	0.00	0.28	82.53	0.00	20vv6ttajyjzq		delete from access$ where d_obj	1	☑
0.02	83	0.00	0.24	109.64	11.49	3un99a0zwp4vd		select owner#,name,namespace,re	1	☑
0.02	30	0.00	0.22	67.48	0.00	0kkhhb2w93cx0		update seg$ set type#=:4,blocks	1	☑
0.02	9	0.00	0.19	100.08	0.00	fuws5bqghb2qh		SELECT D.COLUMN_VALUE , NVL(A.N	1	☑
0.02	9	0.00	0.21	102.44	0.00	d9vzav10pcpfh		select count(*) from dba_schedu	1	□
0.02	37	0.00	0.21	104.13	4.12	616m6uhpa2usu		select i.obj#,i.ts#,i.file#,i.b	3	☑
0.02	237	0.00	0.20	100.67	11.10	1p5grz1gs7fjq		select obj#,type#,ctime,mtime,s	2	☑
0.02	17	0.00	0.22	99.57	0.00	aykvshm7zsabd		select size_for_estimate,	1	□

SQL by CPU Time

CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.01	206	0.02	40.44	4.05	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	□
3.53	206	0.02	35.64	3.58	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	☑
0.93	1	0.93	9.42	0.95	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	□
0.47	2	0.24	4.79	0.48	99.05	0.00	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	☑
0.34	156	0.00	3.46	0.35	97.11	0.00	2mgc4wm4dazsh	SQL Developer	SELECT APPENDCHILDXML (:B19 ,	1	□
0.22	17	0.01	2.25	0.23	97.82	0.00	5yv7yvjgjxu...		select TIME_WAITED_MICRO fro...	1	□
0.08	902	0.00	0.77	0.09	89.74	0.11	3c1kubcdjnppq		update sys.col_usage$ set equ	1	☑
0.06	9	0.01	0.65	0.06	101.52	0.00	7kmbrw7q8hn4g		begin prvt_ilm.ilm_cleanup;	0	□
0.04	240	0.00	0.39	0.03	123.73	0.00	1gfaj4z5hn1kf		delete from dependency$ where d	1	☑
0.03	123	0.00	0.31	0.03	99.00	2.45	8zc85a8249x81	DBMS_SCHEDULER	update obj$ set obj#=:4, type#=	1	☑
0.02	49	0.00	0.25	0.03	93.94	63.49	cvn54b7yz0s8u		select /*+ index(idl_ub1$ i_idl	1	☑
0.02	628	0.00	0.24	0.02	97.14	15.05	96g93hntrzjtr		select /*+ rule */ bucket_cnt,	1	☑
0.02	83	0.00	0.21	0.02	109.64	11.49	3un99a0zwp4vd		select owner#,name,namespace,re	1	☑
0.02	240	0.00	0.18	0.02	82.53	0.00	20vv6ttajyjzq		delete from access$ where d_obj	1	☑
0.02	9	0.00	0.17	0.02	102.44	0.00	d9vzav10pcpfh		select count(*) from dba_schedu	1	□
0.02	37	0.00	0.17	0.02	104.13	4.12	616m6uhpa2usu		select i.obj#,i.ts#,i.file#,i.b	3	☑
0.02	17	0.00	0.17	0.02	99.57	0.00	aykvshm7zsabd		select size_for_estimate,	1	□
0.02	237	0.00	0.16	0.02	100.67	11.10	1p5grz1gs7fjq		select obj#,type#,ctime,mtime,s	2	☑
0.02	9	0.00	0.15	0.02	100.08	0.00	fuws5bqghb2qh		SELECT D.COLUMN_VALUE , NVL(A.N	1	☑
0.01	164	0.00	0.14	0.01	105.47	19.33	9mcxa5qwdxbtq		select position#,sequence#,leve	1	☑

5yv7yvjgjxugg (1-4)

```
SELECT
    time_waited_micro
FROM
    v$system_event
WHERE
    event = 'Shared IO Pool Memory'
```

```
1461717084
SELECT STATEMENT CHOOSE
  MERGE JOIN
    FIXED TABLE FULL X$KSLED
    SORT JOIN
      FIXED TABLE FULL X$KSLEI
```

History information

First Snap ID	First Snap Time	Last Active Time	Cost
2	14-JAN-2020 11:41	14-JAN-2020 11:49	1

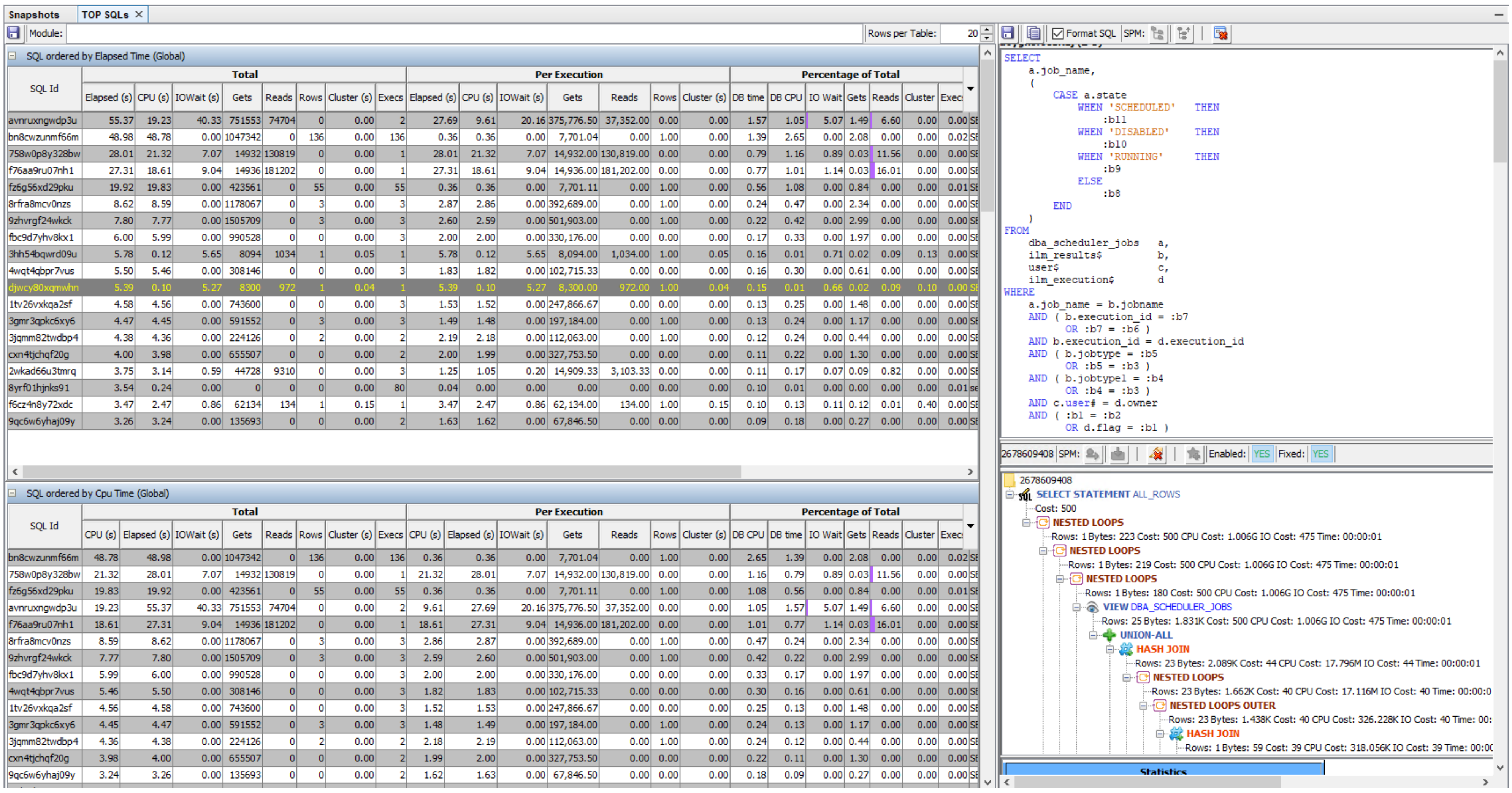
Summary

Stat name	Statement Total	Per Execution
Elapsed Time	0.228	0.013412
CPU Time	0.223	0.013118
Num of Executions	17	
Num of Rows	0	0
Disk Reads	0	0
Buffer Gets	0	0
Direct Writes	0	0
Parse Calls	0	0
User IO Wait Time	0.0	0
Cluster Wait Time	0	0
Application Wait Time	0.0	0
Concurrency Wait Time	0.0	0
Invalidations	0	0

RAC VIEW

Panel TOP SQLs RAC view opens when more nodes/instances are selected.

Note: This functionality is available only for AWR Analyzer (not for STATSPACK).



SQL ordered by Elapsed Time (Global)

SQL Id	Total								Per Execution							Percentage of Total						
	Elapsed (s)	CPU (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	Execs	Elapsed (s)	CPU (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	DB time	DB CPU	IO Wait	Gets	Reads	Cluster	Exec
avnruxngwdp3u	55.37	19.23	40.33	751553	74704	0	0.00	2	27.69	9.61	20.16	375,776.50	37,352.00	0.00	0.00	1.57	1.05	5.07	1.49	6.60	0.00	0.00
bn8cwzunmf66m	48.98	48.78	0.00	1047342	0	136	0.00	136	0.36	0.36	0.00	7,701.04	0.00	1.00	0.00	1.39	2.65	0.00	2.08	0.00	0.00	0.02
758w0p8y328bw	28.01	21.32	7.07	14932	130819	0	0.00	1	28.01	21.32	7.07	14,932.00	130,819.00	0.00	0.00	0.79	1.16	0.89	0.03	11.56	0.00	0.00
f76aa9ru07nh1	27.31	18.61	9.04	14936	181202	0	0.00	1	27.31	18.61	9.04	14,936.00	181,202.00	0.00	0.00	0.77	1.01	1.14	0.03	16.01	0.00	0.00
fz6g56xd29pku	19.92	19.83	0.00	423561	0	55	0.00	55	0.36	0.36	0.00	7,701.11	0.00	1.00	0.00	0.56	1.08	0.00	0.84	0.00	0.00	0.01
8rfra8mcv0nzs	8.62	8.59	0.00	1178067	0	3	0.00	3	2.87	2.86	0.00	392,689.00	0.00	1.00	0.00	0.24	0.47	0.00	2.34	0.00	0.00	0.00
9zhvrgf24wkck	7.80	7.77	0.00	1505709	0	3	0.00	3	2.60	2.59	0.00	501,903.00	0.00	1.00	0.00	0.22	0.42	0.00	2.99	0.00	0.00	0.00
fbc9d7yhv8kx1	6.00	5.99	0.00	990528	0	0	0.00	3	2.00	2.00	0.00	330,176.00	0.00	0.00	0.00	0.17	0.33	0.00	1.97	0.00	0.00	0.00
3hh54bqwrd09u	5.78	0.12	5.65	8094	1034	1	0.05	1	5.78	0.12	5.65	8,094.00	1,034.00	1.00	0.05	0.16	0.01	0.71	0.02	0.09	0.13	0.00
4wqt4qbpr7vus	5.50	5.46	0.00	308146	0	0	0.00	3	1.83	1.82	0.00	102,715.33	0.00	0.00	0.00	0.16	0.30	0.00	0.61	0.00	0.00	0.00
djwcy80xqmwhn	5.39	0.10	5.27	8300	972	1	0.04	1	5.39	0.10	5.27	8,300.00	972.00	1.00	0.04	0.15	0.01	0.66	0.02	0.09	0.10	0.00
1tv26vxkqa2sf	4.58	4.56	0.00	743600	0	0	0.00	3	1.53	1.52	0.00	247,866.67	0.00	0.00	0.00	0.13	0.25	0.00	1.48	0.00	0.00	0.00
3gmr3qpkc6xy6	4.47	4.45	0.00	591552	0	3	0.00	3	1.49	1.48	0.00	197,184.00	0.00	1.00	0.00	0.13	0.24	0.00	1.17	0.00	0.00	0.00
3jqmm82twdbp4	4.38	4.36	0.00	224126	0	2	0.00	2	2.19	2.18	0.00	112,063.00	0.00	1.00	0.00	0.12	0.24	0.00	0.44	0.00	0.00	0.00
cxn4tjchqf20g	4.00	3.98	0.00	655507	0	0	0.00	2	2.00	1.99	0.00	327,753.50	0.00	0.00	0.00	0.11	0.22	0.00	1.30	0.00	0.00	0.00
2wkad66u3tmrq	3.75	3.14	0.59	44728	9310	0	0.00	3	1.25	1.05	0.20	14,909.33	3,103.33	0.00	0.00	0.11	0.17	0.07	0.09	0.82	0.00	0.00
8yrf01hjnks91	3.54	0.24	0.00	0	0	0	0.00	80	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.01	0.00	0.00	0.00	0.00	0.01
f6cz4n8y72xdc	3.47	2.47	0.86	62134	134	1	0.15	1	3.47	2.47	0.86	62,134.00	134.00	1.00	0.15	0.10	0.13	0.11	0.12	0.01	0.40	0.00
9qc6w6yhaj09y	3.26	3.24	0.00	135693	0	0	0.00	2	1.63	1.62	0.00	67,846.50	0.00	0.00	0.00	0.09	0.18	0.00	0.27	0.00	0.00	0.00

SQL ordered by Cpu Time (Global)

SQL Id	Total								Per Execution							Percentage of Total						
	CPU (s)	Elapsed (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	Execs	CPU (s)	Elapsed (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	DB CPU	DB time	IO Wait	Gets	Reads	Cluster	Exec
bn8cwzunmf66m	48.78	48.98	0.00	1047342	0	136	0.00	136	0.36	0.36	0.00	7,701.04	0.00	1.00	0.00	2.65	1.39	0.00	2.08	0.00	0.00	0.02
758w0p8y328bw	21.32	28.01	7.07	14932	130819	0	0.00	1	21.32	28.01	7.07	14,932.00	130,819.00	0.00	0.00	1.16	0.79	0.89	0.03	11.56	0.00	0.00
fz6g56xd29pku	19.83	19.92	0.00	423561	0	55	0.00	55	0.36	0.36	0.00	7,701.11	0.00	1.00	0.00	1.08	0.56	0.00	0.84	0.00	0.00	0.01
avnruxngwdp3u	19.23	55.37	40.33	751553	74704	0	0.00	2	9.61	27.69	20.16	375,776.50	37,352.00	0.00	0.00	1.05	1.57	5.07	1.49	6.60	0.00	0.00
f76aa9ru07nh1	18.61	27.31	9.04	14936	181202	0	0.00	1	18.61	27.31	9.04	14,936.00	181,202.00	0.00	0.00	1.01	0.77	1.14	0.03	16.01	0.00	0.00
8rfra8mcv0nzs	8.59	8.62	0.00	1178067	0	3	0.00	3	2.86	2.87	0.00	392,689.00	0.00	1.00	0.00	0.47	0.24	0.00	2.34	0.00	0.00	0.00
9zhvrgf24wkck	7.77	7.80	0.00	1505709	0	3	0.00	3	2.59	2.60	0.00	501,903.00	0.00	1.00	0.00	0.42	0.22	0.00	2.99	0.00	0.00	0.00
fbc9d7yhv8kx1	5.99	6.00	0.00	990528	0	0	0.00	3	2.00	2.00	0.00	330,176.00	0.00	0.00	0.00	0.33	0.17	0.00	1.97	0.00	0.00	0.00
4wqt4qbpr7vus	5.46	5.50	0.00	308146	0	0	0.00	3	1.82	1.83	0.00	102,715.33	0.00	0.00	0.00	0.30	0.16	0.00	0.61	0.00	0.00	0.00
1tv26vxkqa2sf	4.56	4.58	0.00	743600	0	0	0.00	3	1.52	1.53	0.00	247,866.67	0.00	0.00	0.00	0.25	0.13	0.00	1.48	0.00	0.00	0.00
3gmr3qpkc6xy6	4.45	4.47	0.00	591552	0	3	0.00	3	1.48	1.49	0.00	197,184.00	0.00	1.00	0.00	0.24	0.13	0.00	1.17	0.00	0.00	0.00
3jqmm82twdbp4	4.36	4.38	0.00	224126	0	2	0.00	2	2.18	2.19	0.00	112,063.00	0.00	1.00	0.00	0.24	0.12	0.00	0.44	0.00	0.00	0.00
cxn4tjchqf20g	3.98	4.00	0.00	655507	0	0	0.00	2	1.99	2.00	0.00	327,753.50	0.00	0.00	0.00	0.22	0.11	0.00	1.30	0.00	0.00	0.00
9qc6w6yhaj09y	3.24	3.26	0.00	135693	0	0	0.00	2	1.62	1.63	0.00	67,846.50	0.00	0.00	0.00	0.18	0.09	0.00	0.27	0.00	0.00	0.00

```
SELECT
    a.job_name,
    (
        CASE a.state
            WHEN 'SCHEDULED'   THEN
                :b11
            WHEN 'DISABLED'    THEN
                :b10
            WHEN 'RUNNING'     THEN
                :b9
            ELSE
                :b8
        END
    )
FROM
    dba_scheduler_jobs   a,
    ilm_results$         b,
    user$                c,
    ilm_execution$       d
WHERE
    a.job_name = b.jobname
    AND ( b.execution_id = :b7
          OR :b7 = :b6 )
    AND b.execution_id = d.execution_id
    AND ( b.jobtype = :b5
          OR :b5 = :b3 )
    AND ( b.jobtype1 = :b4
          OR :b4 = :b3 )
    AND c.user# = d.owner
    AND ( :b1 = :b2
          OR d.flag = :b1 )
```

2678609408 SPM: Enabled: YES Fixed: YES

```
2678609408
SELECT STATEMENT ALL_ROWS
 Cost: 500
 NESTED LOOPS
  Rows: 1 Bytes: 223 Cost: 500 CPU Cost: 1.006G IO Cost: 475 Time: 00:00:01
  NESTED LOOPS
   Rows: 1 Bytes: 219 Cost: 500 CPU Cost: 1.006G IO Cost: 475 Time: 00:00:01
   NESTED LOOPS
    Rows: 1 Bytes: 180 Cost: 500 CPU Cost: 1.006G IO Cost: 475 Time: 00:00:01
    VIEW DBA_SCHEDULER_JOBS
     Rows: 25 Bytes: 1.831K Cost: 500 CPU Cost: 1.006G IO Cost: 475 Time: 00:00:01
     UNION-ALL
      HASH JOIN
       Rows: 23 Bytes: 2.089K Cost: 44 CPU Cost: 17.796M IO Cost: 44 Time: 00:00:01
       NESTED LOOPS
        Rows: 23 Bytes: 1.662K Cost: 40 CPU Cost: 17.116M IO Cost: 40 Time: 00:00:0
        NESTED LOOPS OUTER
         Rows: 23 Bytes: 1.438K Cost: 40 CPU Cost: 326.228K IO Cost: 40 Time: 00:
         HASH JOIN
          Rows: 1 Bytes: 59 Cost: 39 CPU Cost: 318.056K IO Cost: 39 Time: 00:00
```

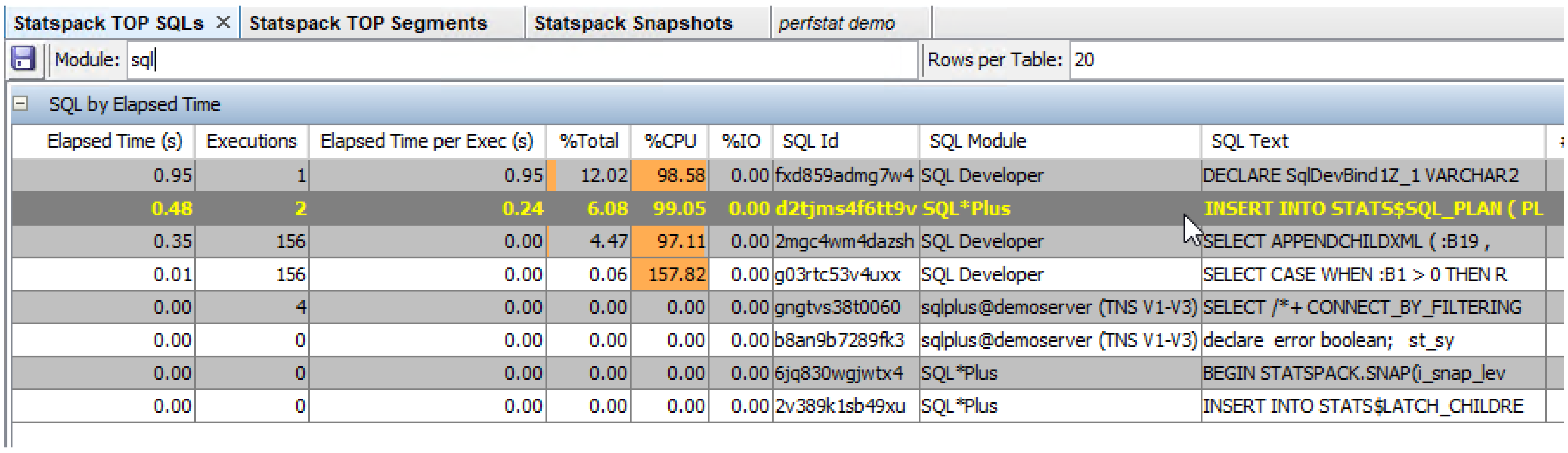
SELECTION CRITERIA

In top section there are selection criteria:

- Module
- Rows per table

  and
- SAVE button

Module –SQLs are filtered by entered text: UPPER(SQL Module) LIKE UPPER(%'entered text'%)



Module: sql

Rows per Table: 20

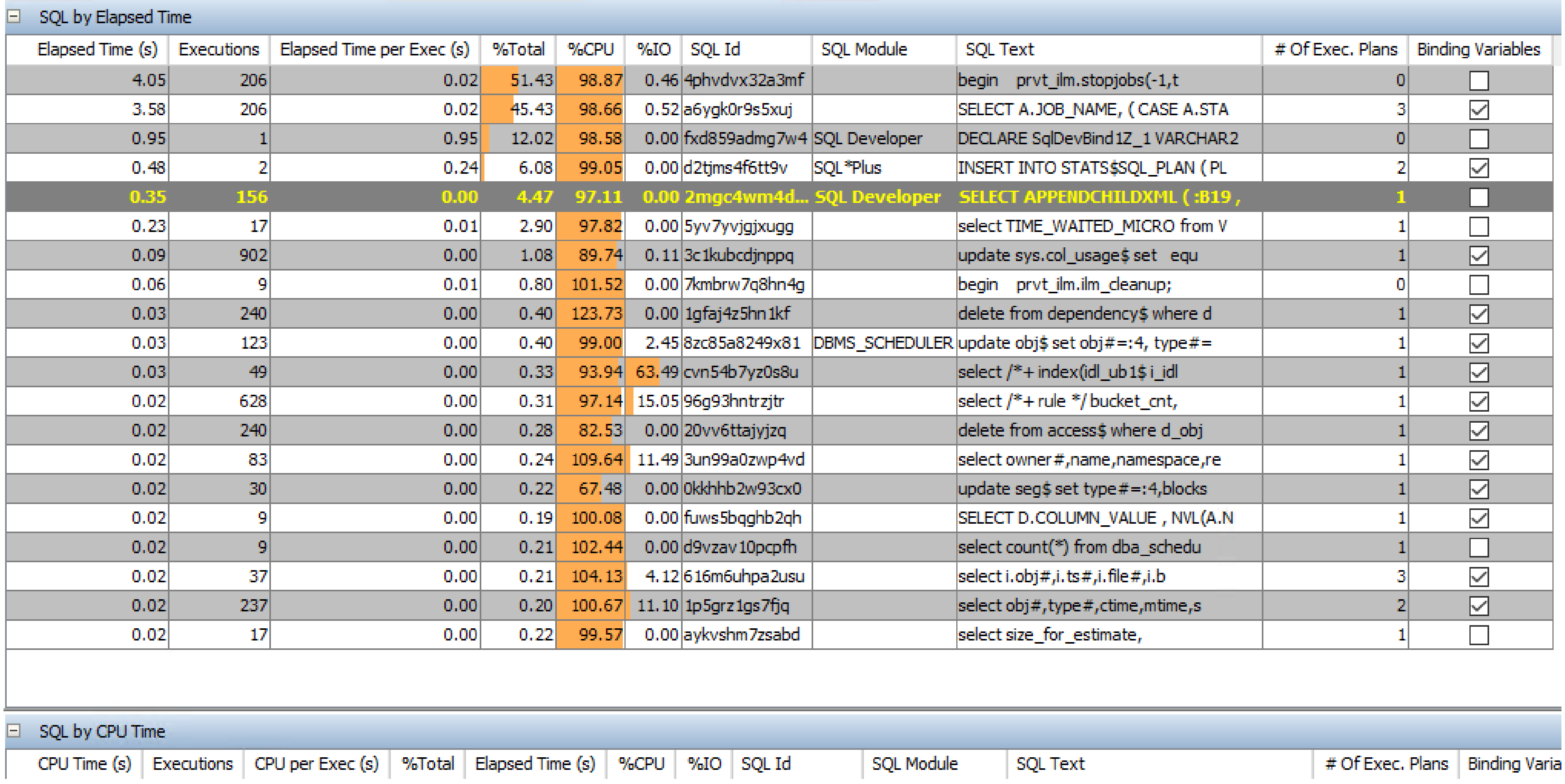
SQL by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text
0.95	1	0.95	12.02	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2
0.48	2	0.24	6.08	99.05	0.00	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL
0.35	156	0.00	4.47	97.11	0.00	2mgc4wm4dazsh	SQL Developer	SELECT APPENDCHILDXML (:B19 ,
0.01	156	0.00	0.06	157.82	0.00	g03rtc53v4uxx	SQL Developer	SELECT CASE WHEN :B1 > 0 THEN R
0.00	4	0.00	0.00	0.00	0.00	gngtvs38t0060	sqlplus@demoserver (TNS V1-V3)	SELECT /*+ CONNECT_BY_FILTERING
0.00	0	0.00	0.00	0.00	0.00	b8an9b7289fk3	sqlplus@demoserver (TNS V1-V3)	declare error boolean; st_sy
0.00	0	0.00	0.00	0.00	0.00	6jq830wgjwtx4	SQL*Plus	BEGIN STATSPACK.SNAP(i_snap_lev
0.00	0	0.00	0.00	0.00	0.00	2v389k1sb49xu	SQL*Plus	INSERT INTO STATS$LATCH_CHILDRE

Note: This criterion is very useful when optimizing processes. When name of online or batch process is entered in the field "Module" system will identify all SQLs in this process.

MAIN PANEL

Main panel displays all statistics in separate tables.



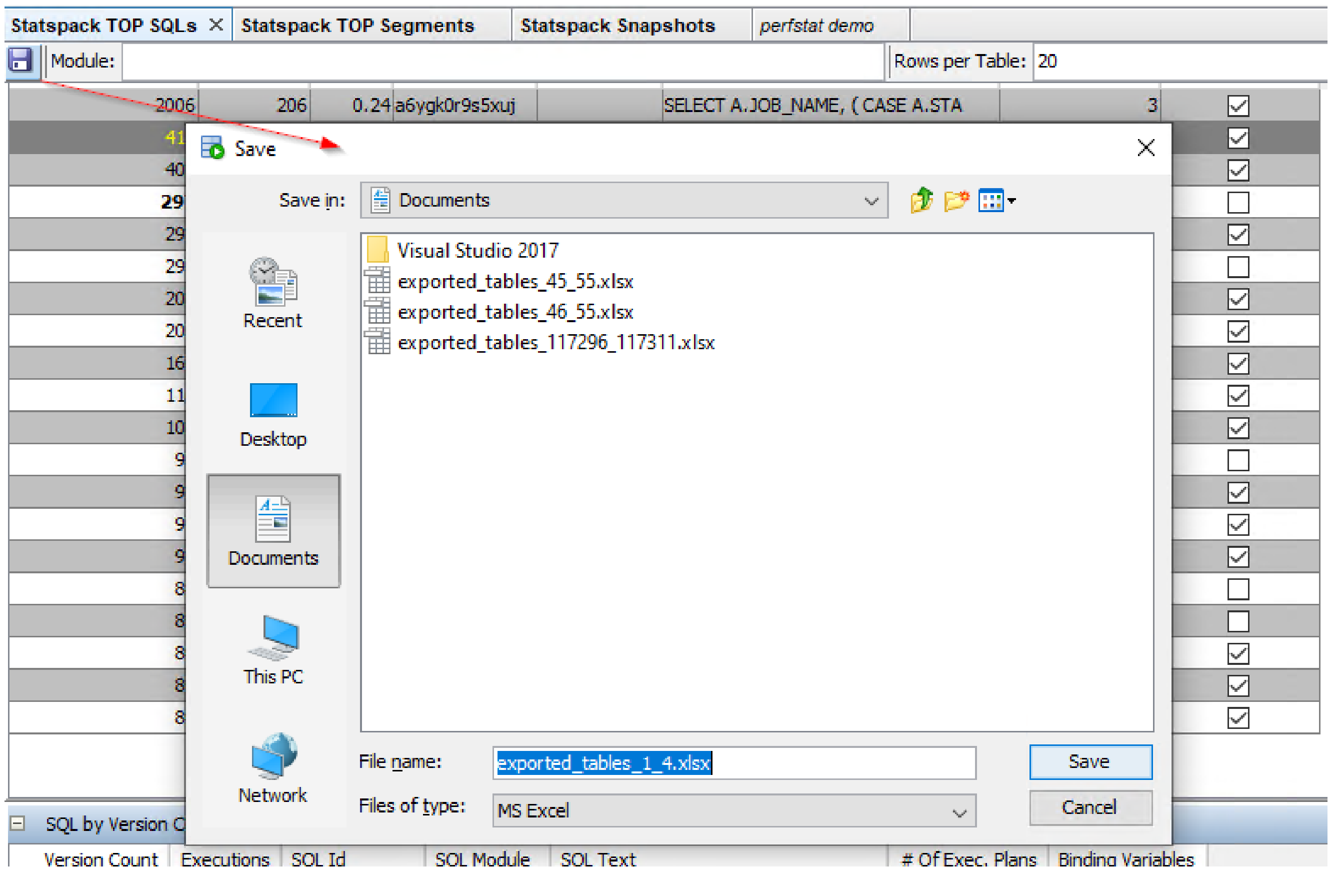
⊟ SQL by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.05	206	0.02	51.43	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	□
3.58	206	0.02	45.43	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	☑
0.95	1	0.95	12.02	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	□
0.48	2	0.24	6.08	99.05	0.00	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	☑
0.35	156	0.00	4.47	97.11	0.00	2mgc4wm4d...	SQL Developer	SELECT APPENDCHILDXML (:B19 ,	1	□
0.23	17	0.01	2.90	97.82	0.00	5yv7yvjgjxugg		select TIME_WAITED_MICRO from V	1	□
0.09	902	0.00	1.08	89.74	0.11	3c1kubcdjnppq		update sys.col_usage$ set equ	1	☑
0.06	9	0.01	0.80	101.52	0.00	7kmbrw7q8hn4g		begin prvt_ilm.ilm_cleanup;	0	□
0.03	240	0.00	0.40	123.73	0.00	1gfaj4z5hn1kf		delete from dependency$ where d	1	☑
0.03	123	0.00	0.40	99.00	2.45	8zc85a8249x81	DBMS_SCHEDULER	update obj$ set obj#=:4, type#=	1	☑
0.03	49	0.00	0.33	93.94	63.49	cvn54b7yz0s8u		select /*+ index(idl_ub1$ i_idl	1	☑
0.02	628	0.00	0.31	97.14	15.05	96g93hntrzjtr		select /*+ rule */ bucket_cnt,	1	☑
0.02	240	0.00	0.28	82.53	0.00	20vv6ttajyjzq		delete from access$ where d_obj	1	☑
0.02	83	0.00	0.24	109.64	11.49	3un99a0zwp4vd		select owner#,name,namespace,re	1	☑
0.02	30	0.00	0.22	67.48	0.00	0kkhhb2w93cx0		update seg$ set type#=:4,blocks	1	☑
0.02	9	0.00	0.19	100.08	0.00	fuws5bqghb2qh		SELECT D.COLUMN_VALUE , NVL(A.N	1	☑
0.02	9	0.00	0.21	102.44	0.00	d9vzav10pcpfh		select count(*) from dba_schedu	1	□
0.02	37	0.00	0.21	104.13	4.12	616m6uhpa2usu		select i.obj#,i.ts#,i.file#,i.b	3	☑
0.02	237	0.00	0.20	100.67	11.10	1p5grz1gs7fjq		select obj#,type#,ctime,mtime,s	2	☑
0.02	17	0.00	0.22	99.57	0.00	aykvshm7zsabd		select size_for_estimate,	1	□

⊟ SQL by CPU Time

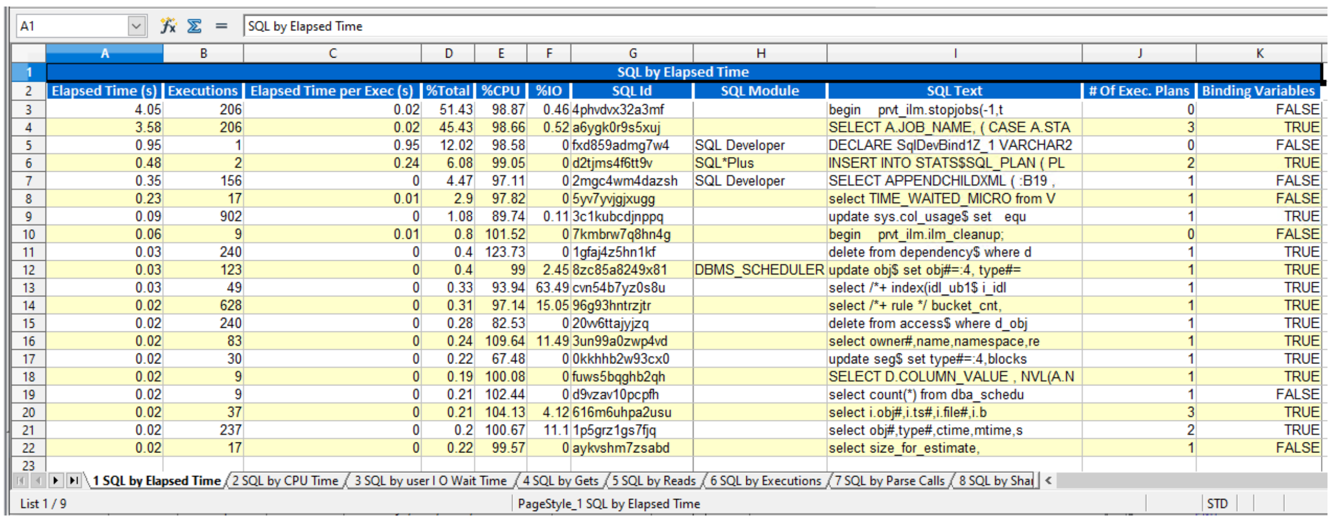
CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Varia

In the left top corner there is a button for exporting all tables into EXCEL .xlsx file.



Name of file is generated automatically. It contains identification number of the snapshot's interval snap_from and snap_to (1 and 4). It can be altered by user.

Generated EXCEL file has particular SQL statistics stored in separate tabs named by type of statistics.



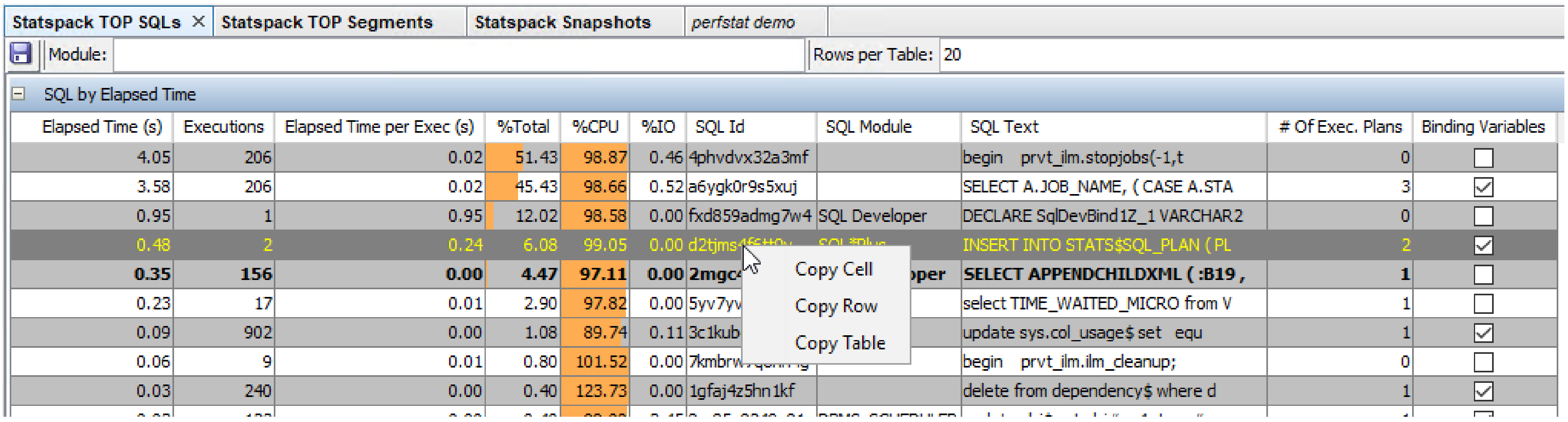
SQL by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.05	206	0.02	51.43	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	FALSE
3.58	206	0.02	45.43	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	TRUE
0.95	1	0.95	12.02	98.58	0	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	FALSE
0.48	2	0.24	6.08	99.05	0	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	TRUE
0.35	156	0	4.47	97.11	0	2mgc4wm4dazsh	SQL Developer	SELECT APPENDCHILDXML (:B19 ,	1	FALSE
0.23	17	0.01	2.9	97.82	0	5yv7yvjgjxugg		select TIME_WAITED_MICRO from V	1	FALSE
0.09	902	0	1.08	89.74	0.11	3c1kubcdjnppq		update sys.col_usage$ set equ	1	TRUE
0.06	9	0.01	0.8	101.52	0	7kmbrw7q8hn4g		begin prvt_ilm.ilm_cleanup;	0	FALSE
0.03	240	0	0.4	123.73	0	1gfaj4z5hn1kf		delete from dependency$ where d	1	TRUE
0.03	123	0	0.4	99	2.45	8zc85a8249x81	DBMS_SCHEDULER	update obj$ set obj#=:4, type#=	1	TRUE
0.03	49	0	0.33	93.94	63.49	cvn54b7yz0s8u		select /*+ index(idl_ub1$ i_idl	1	TRUE
0.02	628	0	0.31	97.14	15.05	96g93hntrzjtr		select /*+ rule */ bucket_cnt,	1	TRUE
0.02	240	0	0.28	82.53	0	20w6ttajyjzq		delete from access$ where d_obj	1	TRUE
0.02	83	0	0.24	109.64	11.49	3un99a0zwp4vd		select owner#,name,namespace,re	1	TRUE
0.02	30	0	0.22	67.48	0	0kkhhb2w93cx0		update seg$ set type#=:4,blocks	1	TRUE
0.02	9	0	0.19	100.08	0	fuws5bqghb2qh		SELECT D.COLUMN_VALUE , NVL(A.N	1	TRUE
0.02	9	0	0.21	102.44	0	d9vzav10pcpfh		select count(*) from dba_schedu	1	FALSE
0.02	37	0	0.21	104.13	4.12	616m6uhpa2usu		select i.obj#,i.ts#,i.file#,i.b	3	TRUE
0.02	237	0	0.2	100.67	11.1	1p5grz1gs7fjq		select obj#,type#,ctime,mtime,s	2	TRUE
0.02	17	0	0.22	99.57	0	aykvshm7zsabd		select size_for_estimate,	1	FALSE

COPY OF DATA INTO CLIPBOARD

Content of the cell, row or whole table can be copied into clipboard by selecting required object and clicking with right mouse button. Menu appears:

- Copy Cell
- Copy Row
- Copy Table



Module:

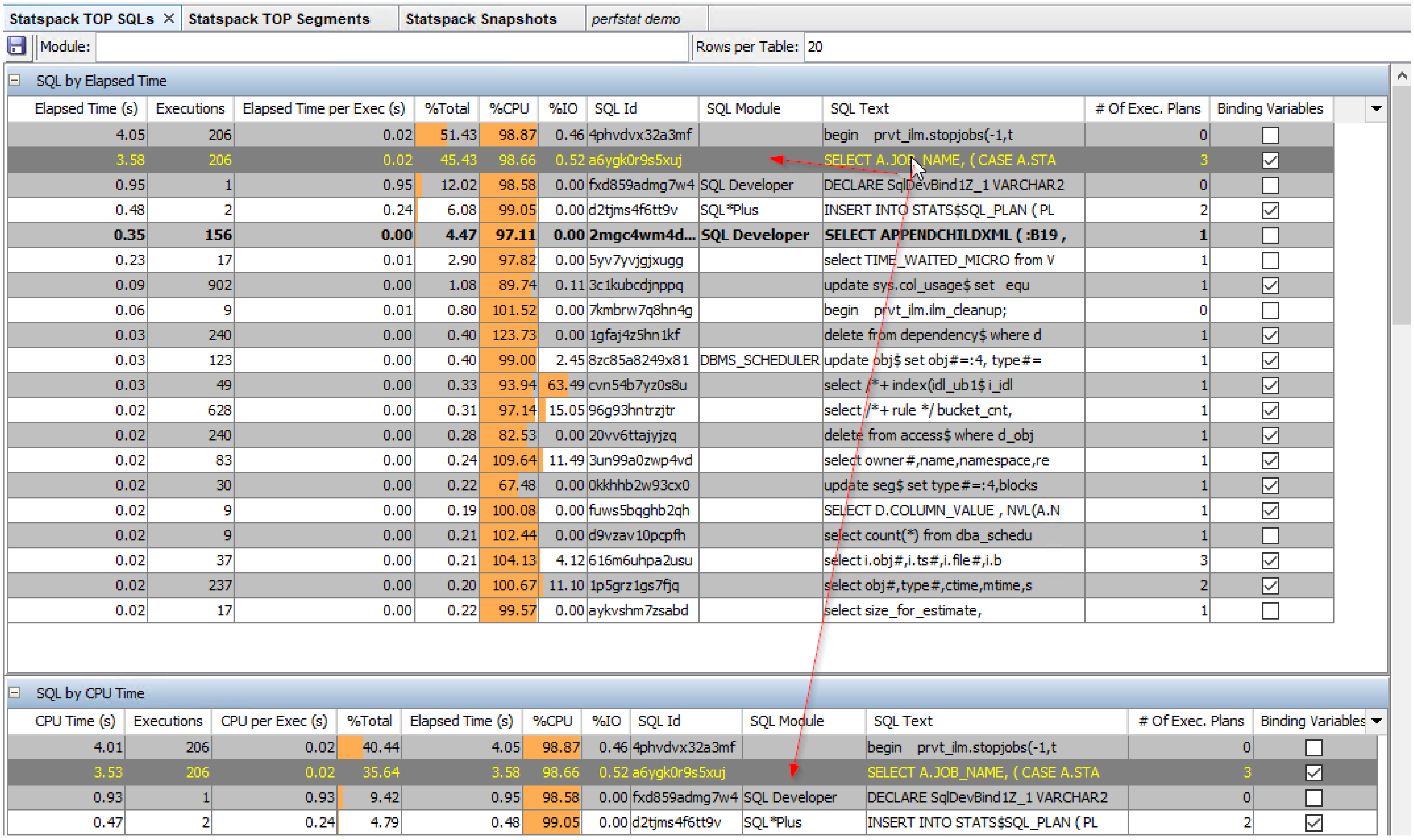
Rows per Table: 20

SQL by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.05	206	0.02	51.43	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	□
3.58	206	0.02	45.43	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	☑
0.95	1	0.95	12.02	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	□
0.48	2	0.24	6.08	99.05	0.00	d2tjms	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	☑
0.35	**156**	**0.00**	**4.47**	**97.11**	**0.00**	**2mgc**	**per**	**SELECT APPENDCHILDXML (:B19 ,**	**1**	□
0.23	17	0.01	2.90	97.82	0.00	5yv7yv		select TIME_WAITED_MICRO from V	1	□
0.09	902	0.00	1.08	89.74	0.11	3c1kub		update sys.col_usage$ set equ	1	☑
0.06	9	0.01	0.80	101.52	0.00	7kmbrw		begin prvt_ilm.ilm_cleanup;	0	□
0.03	240	0.00	0.40	123.73	0.00	1gfaj4z5hn1kf		delete from dependency$ where d	1	☑

VISUAL CONNECTION OF THE TABLES

While moving mouse pointer over the table, the rows with the same SQL Id are selected automatically in the other sections which allows better intelligibility.



Module:

Rows per Table: 20

SQL by Elapsed Time

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.05	206	0.02	51.43	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	□
3.58	206	0.02	45.43	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	☑
0.95	1	0.95	12.02	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	□
0.48	2	0.24	6.08	99.05	0.00	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	☑
0.35	**156**	**0.00**	**4.47**	**97.11**	**0.00**	**2mgc4wm4d...**	**SQL Developer**	**SELECT APPENDCHILDXML (:B19 ,**	**1**	□
0.23	17	0.01	2.90	97.82	0.00	5yv7yvjgjxugg		select TIME_WAITED_MICRO from V	1	□
0.09	902	0.00	1.08	89.74	0.11	3c1kubcdjnppq		update sys.col_usage$ set equ	1	☑
0.06	9	0.01	0.80	101.52	0.00	7kmbrw7q8hn4g		begin prvt_ilm.ilm_cleanup;	0	□
0.03	240	0.00	0.40	123.73	0.00	1gfaj4z5hn1kf		delete from dependency$ where d	1	☑
0.03	123	0.00	0.40	99.00	2.45	8zc85a8249x81	DBMS_SCHEDULER	update obj$ set obj#=:4, type#=	1	☑
0.03	49	0.00	0.33	93.94	63.49	cvn54b7yz0s8u		select /*+ index(idl_ub1$ i_idl	1	☑
0.02	628	0.00	0.31	97.14	15.05	96g93hntrzjtr		select /*+ rule */ bucket_cnt,	1	☑
0.02	240	0.00	0.28	82.53	0.00	20vv6ttajyjzq		delete from access$ where d_obj	1	☑
0.02	83	0.00	0.24	109.64	11.49	3un99a0zwp4vd		select owner#,name,namespace,re	1	☑
0.02	30	0.00	0.22	67.48	0.00	0kkhhb2w93cx0		update seg$ set type#=:4,blocks	1	☑
0.02	9	0.00	0.19	100.08	0.00	fuws5bqghb2qh		SELECT D.COLUMN_VALUE , NVL(A.N	1	☑
0.02	9	0.00	0.21	102.44	0.00	d9vzav10pcpfh		select count(*) from dba_schedu	1	□
0.02	37	0.00	0.21	104.13	4.12	616m6uhpa2usu		select i.obj#,i.ts#,i.file#,i.b	3	☑
0.02	237	0.00	0.20	100.67	11.10	1p5grz1gs7fjq		select obj#,type#,ctime,mtime,s	2	☑
0.02	17	0.00	0.22	99.57	0.00	aykvshm7zsabd		select size_for_estimate,	1	□

SQL by CPU Time

CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text	# Of Exec. Plans	Binding Variables
4.01	206	0.02	40.44	4.05	98.87	0.46	4phvdvx32a3mf		begin prvt_ilm.stopjobs(-1,t	0	□
3.53	206	0.02	35.64	3.58	98.66	0.52	a6ygk0r9s5xuj		SELECT A.JOB_NAME, (CASE A.STA	3	☑
0.93	1	0.93	9.42	0.95	98.58	0.00	fxd859admg7w4	SQL Developer	DECLARE SqlDevBind1Z_1 VARCHAR2	0	□
0.47	2	0.24	4.79	0.48	99.05	0.00	d2tjms4f6tt9v	SQL*Plus	INSERT INTO STATS$SQL_PLAN (PL	2	☑

SQL DETAILS

After click on any row details of selected SQL displays on the right side. These details are constrained by used level of STATSPACK / AWR.

SQL Details panel consist of these sections:

- Panel bar
- SQL text
- Data for every execution plan (PLAN_HASH_VALUE):
    - SPM panel - only in Analyze AWR
    - execution plan
    - historical information
    - bind variables - variables from runtime
- Summary for all execution plans

PANEL BAR

Panel bar contain:

- Export of all details into PDF
- Copy - copy SQL text into clipboard
- Checkbox for SQL formatting

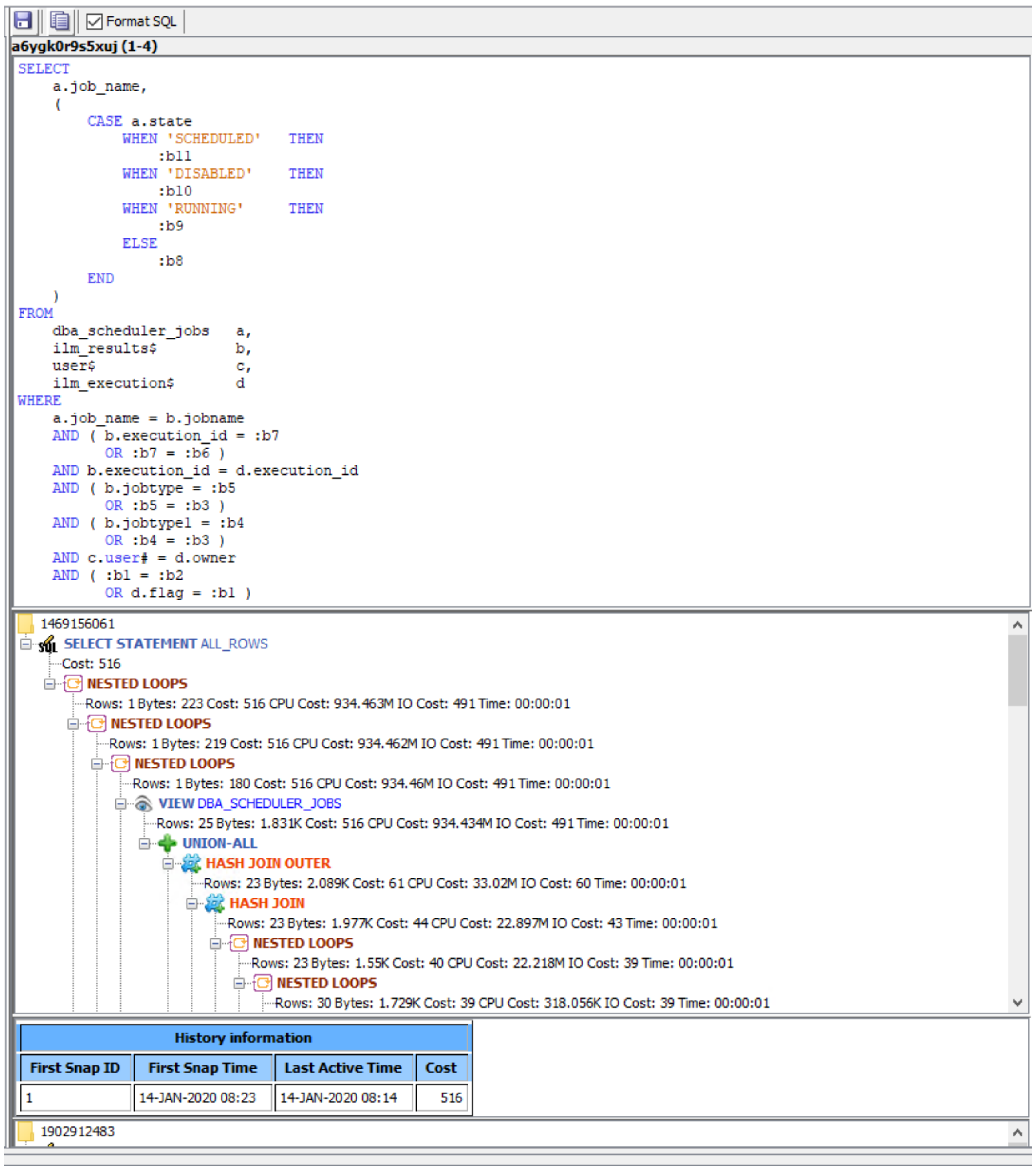
Export example

PDF

sqlDetails_a6ygk0r9
s5xuj_1_4.pdf

SQL TEXT

SQL Text displays the text of SQLquery either formatted or in original form depending on value of "Format SQL" checkbox in panel bar.



a6ygk0r9s5xuj (1-4)

```
SELECT
    a.job_name,
    (
        CASE a.state
            WHEN 'SCHEDULED'   THEN
                :b11
            WHEN 'DISABLED'    THEN
                :b10
            WHEN 'RUNNING'     THEN
                :b9
            ELSE
                :b8
        END
    )
FROM
    dba_scheduler_jobs   a,
    ilm_results$         b,
    user$                c,
    ilm_execution$       d
WHERE
    a.job_name = b.jobname
    AND ( b.execution_id = :b7
          OR :b7 = :b6 )
    AND b.execution_id = d.execution_id
    AND ( b.jobtype = :b5
          OR :b5 = :b3 )
    AND ( b.jobtype1 = :b4
          OR :b4 = :b3 )
    AND c.user# = d.owner
    AND ( :b1 = :b2
          OR d.flag = :b1 )
```

```
1469156061
SELECT STATEMENT ALL_ROWS
 Cost: 516
  NESTED LOOPS
   Rows: 1 Bytes: 223 Cost: 516 CPU Cost: 934.463M IO Cost: 491 Time: 00:00:01
    NESTED LOOPS
     Rows: 1 Bytes: 219 Cost: 516 CPU Cost: 934.462M IO Cost: 491 Time: 00:00:01
      NESTED LOOPS
       Rows: 1 Bytes: 180 Cost: 516 CPU Cost: 934.46M IO Cost: 491 Time: 00:00:01
        VIEW DBA_SCHEDULER_JOBS
         Rows: 25 Bytes: 1.831K Cost: 516 CPU Cost: 934.434M IO Cost: 491 Time: 00:00:01
          UNION-ALL
           HASH JOIN OUTER
            Rows: 23 Bytes: 2.089K Cost: 61 CPU Cost: 33.02M IO Cost: 60 Time: 00:00:01
             HASH JOIN
              Rows: 23 Bytes: 1.977K Cost: 44 CPU Cost: 22.897M IO Cost: 43 Time: 00:00:01
               NESTED LOOPS
                Rows: 23 Bytes: 1.55K Cost: 40 CPU Cost: 22.218M IO Cost: 39 Time: 00:00:01
                 NESTED LOOPS
                  Rows: 30 Bytes: 1.729K Cost: 39 CPU Cost: 318.056K IO Cost: 39 Time: 00:00:01
```

History information			
First Snap ID	First Snap Time	Last Active Time	Cost
1	14-JAN-2020 08:23	14-JAN-2020 08:14	516

1902912483

SPM TOOL

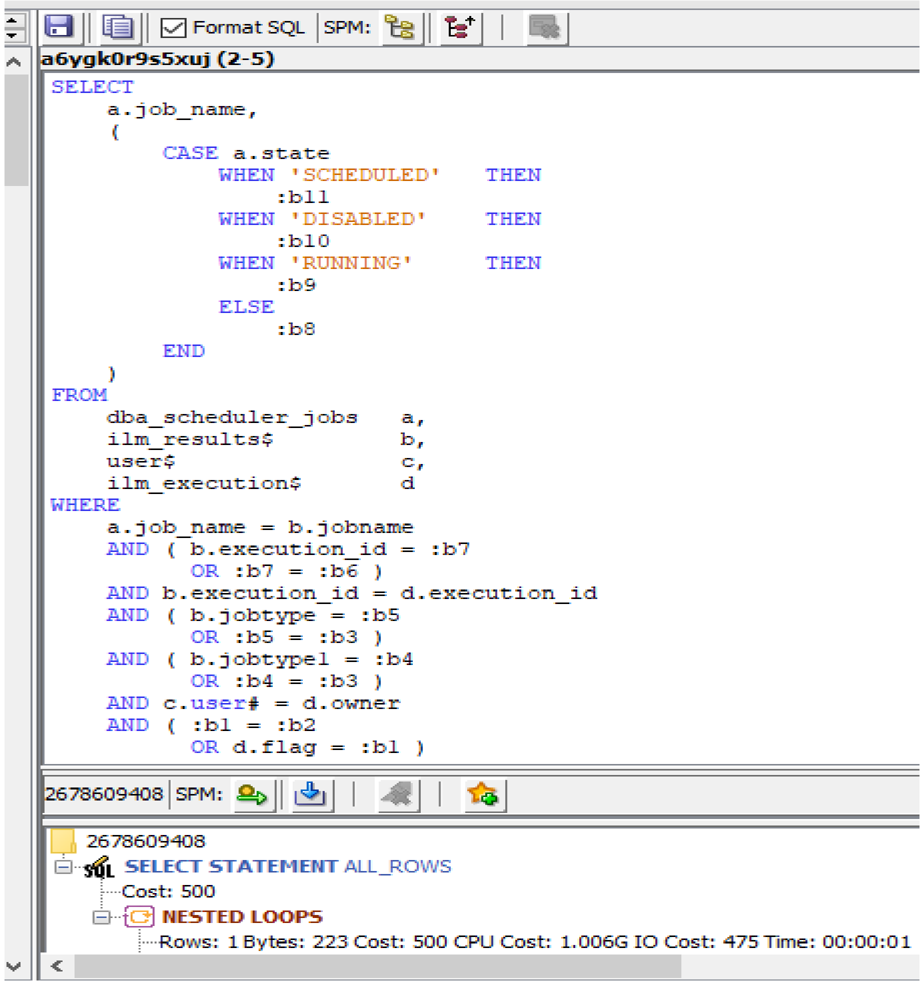
AWR Analyzer contains also tools for work with SPM.

Briefly about SPM:

Performance of any database depends mainly on the execution of the query. Oracle optimizer is not always perfect in selecting the best execution plan without user's intervention. Execution plan can change from many reasons including: optimizer statistics regulations, changes of the parameters of optimizer or schemes definitions and metadata. Optimizer's inability to guarantee best execution plan led some users to freeze the SQL execution plans or lock the optimizer statistics.

But this approach prevent them from using the new features of optimizer which could improve performance of the SQL queries. Ideal solution would be keeping actual execution plan during the changes of environment and changes of the plan would be allowed only in case of improvement of the performance.

SPM provides user with such frame and allows him fully manage SQL execution plans. By using SPM optimizer automatically manage execution plans and guarantee to use only verified plans. When new execution plan is found for SQL, it will not be used until is verified by database whether has the same or better performance than actual plan.



a6ygk0r9s5xuj (2-5)

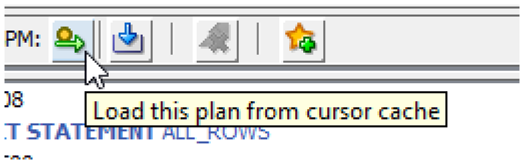
```
SELECT
     a.job_name,
     (
          CASE a.state
               WHEN 'SCHEDULED'    THEN
                    :b11
               WHEN 'DISABLED'     THEN
                    :b10
               WHEN 'RUNNING'      THEN
                    :b9
               ELSE
                    :b8
          END
     )
FROM
     dba_scheduler_jobs    a,
     ilm_results$          b,
     user$                 c,
     ilm_execution$        d
WHERE
     a.job_name = b.jobname
     AND ( b.execution_id = :b7
        OR :b7 = :b6 )
     AND b.execution_id = d.execution_id
     AND ( b.jobtype = :b5
        OR :b5 = :b3 )
     AND ( b.jobtype1 = :b4
        OR :b4 = :b3 )
     AND c.user# = d.owner
     AND ( :b1 = :b2
        OR d.flag = :b1 )
```

2678609408 SPM:

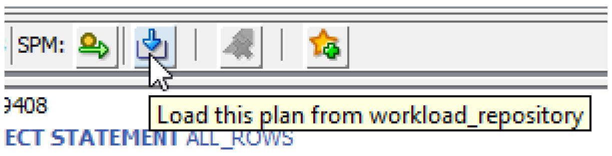
```
2678609408
⊟ SELECT STATEMENT ALL_ROWS
   Cost: 500
   ⊟ NESTED LOOPS
      Rows: 1 Bytes: 223 Cost: 500 CPU Cost: 1.006G IO Cost: 475 Time: 00:00:01
```

SPM functionality

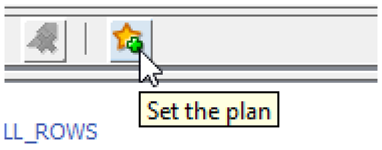
Plans may be loaded from cursor cache



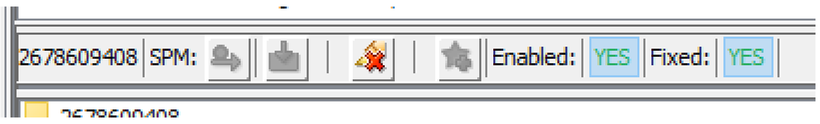
Or from workload repository



Setting the plan by one click

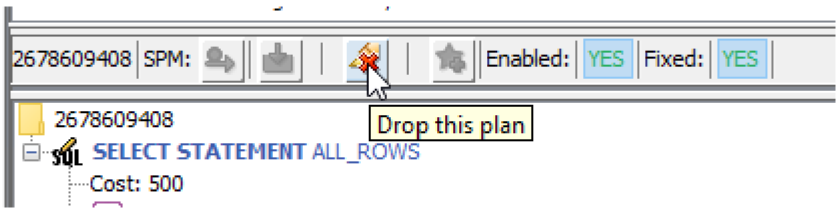


When plan is set - new control buttons and information fields are displayed. These are used to enable plan or to set it's parameters.



2678609408 SPM: Enabled: YES Fixed: YES

Plan can be removed by clicking on button "Drop this plan"

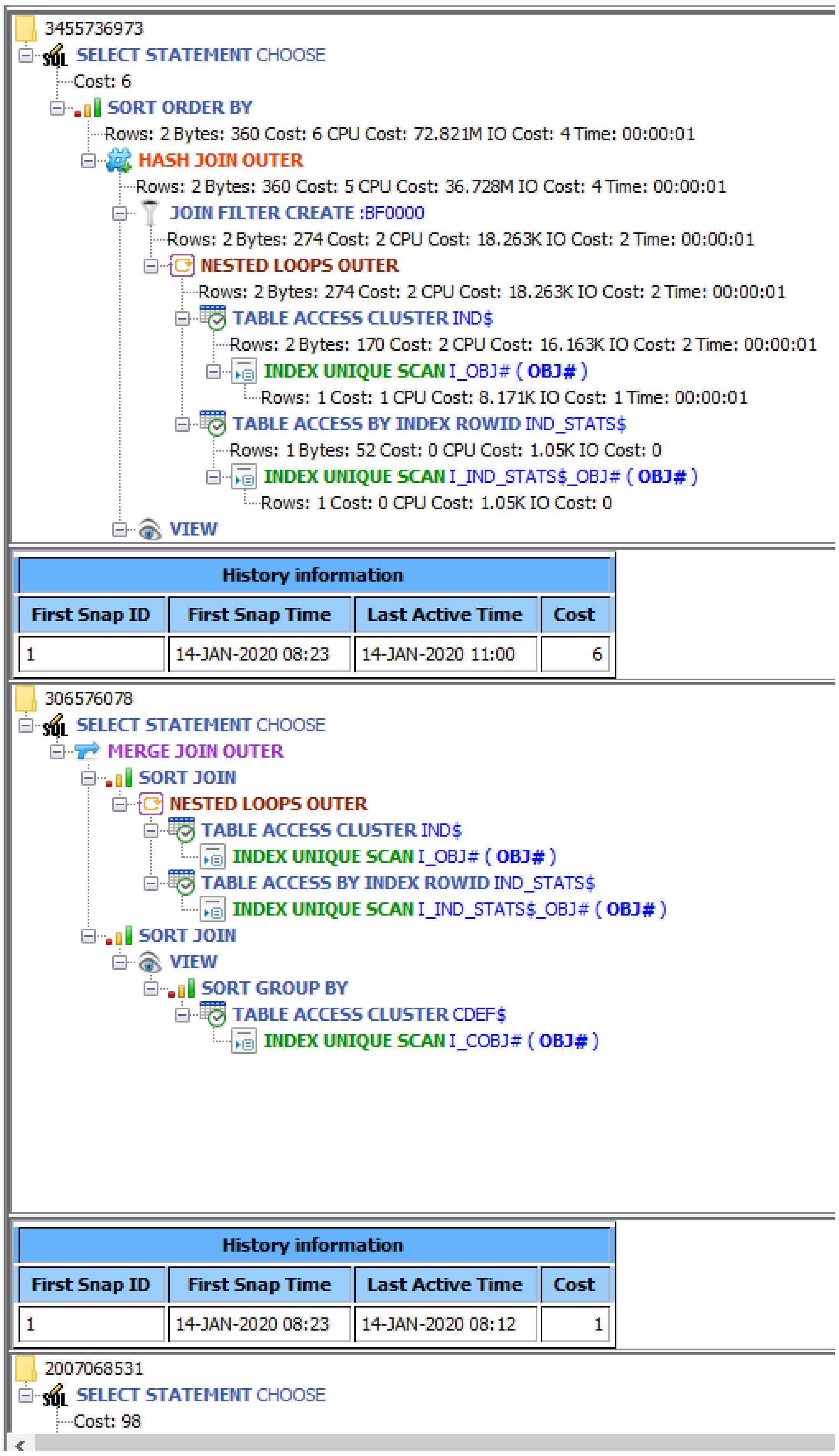


2678609408 SPM: Enabled: YES Fixed: YES

```
2678609408
SELECT STATEMENT ALL_ROWS
  Cost: 500
```

EXECUTION PLAN

In details of SQL query all plans with costs are displayed by hash value in the tree.
Root of the tree shows PLAN_HASH_VALUE number.



```
3455736973
SELECT STATEMENT CHOOSE
  Cost: 6
  SORT ORDER BY
    Rows: 2 Bytes: 360 Cost: 6 CPU Cost: 72.821M IO Cost: 4 Time: 00:00:01
    HASH JOIN OUTER
      Rows: 2 Bytes: 360 Cost: 5 CPU Cost: 36.728M IO Cost: 4 Time: 00:00:01
      JOIN FILTER CREATE :BF0000
        Rows: 2 Bytes: 274 Cost: 2 CPU Cost: 18.263K IO Cost: 2 Time: 00:00:01
        NESTED LOOPS OUTER
          Rows: 2 Bytes: 274 Cost: 2 CPU Cost: 18.263K IO Cost: 2 Time: 00:00:01
          TABLE ACCESS CLUSTER IND$
            Rows: 2 Bytes: 170 Cost: 2 CPU Cost: 16.163K IO Cost: 2 Time: 00:00:01
            INDEX UNIQUE SCAN I_OBJ# ( OBJ# )
              Rows: 1 Cost: 1 CPU Cost: 8.171K IO Cost: 1 Time: 00:00:01
          TABLE ACCESS BY INDEX ROWID IND_STATS$
            Rows: 1 Bytes: 52 Cost: 0 CPU Cost: 1.05K IO Cost: 0
            INDEX UNIQUE SCAN I_IND_STATS$_OBJ# ( OBJ# )
              Rows: 1 Cost: 0 CPU Cost: 1.05K IO Cost: 0
      VIEW
```

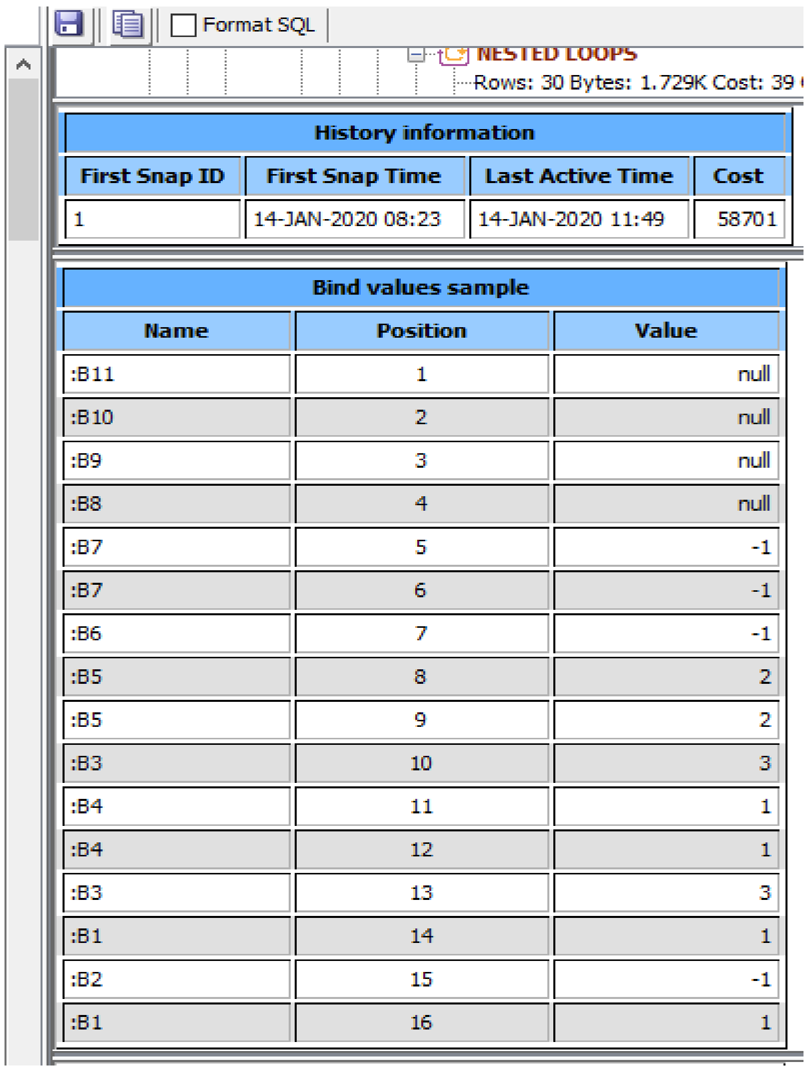
History information			
First Snap ID	First Snap Time	Last Active Time	Cost
1	14-JAN-2020 08:23	14-JAN-2020 11:00	6

```
306576078
SELECT STATEMENT CHOOSE
  MERGE JOIN OUTER
    SORT JOIN
      NESTED LOOPS OUTER
        TABLE ACCESS CLUSTER IND$
          INDEX UNIQUE SCAN I_OBJ# ( OBJ# )
        TABLE ACCESS BY INDEX ROWID IND_STATS$
          INDEX UNIQUE SCAN I_IND_STATS$_OBJ# ( OBJ# )
    SORT JOIN
      VIEW
        SORT GROUP BY
          TABLE ACCESS CLUSTER CDEF$
            INDEX UNIQUE SCAN I_COBJ# ( OBJ# )
```

History information			
First Snap ID	First Snap Time	Last Active Time	Cost
1	14-JAN-2020 08:23	14-JAN-2020 08:12	1

```
2007068531
SELECT STATEMENT CHOOSE
  Cost: 98
```

BIND VARIABLES

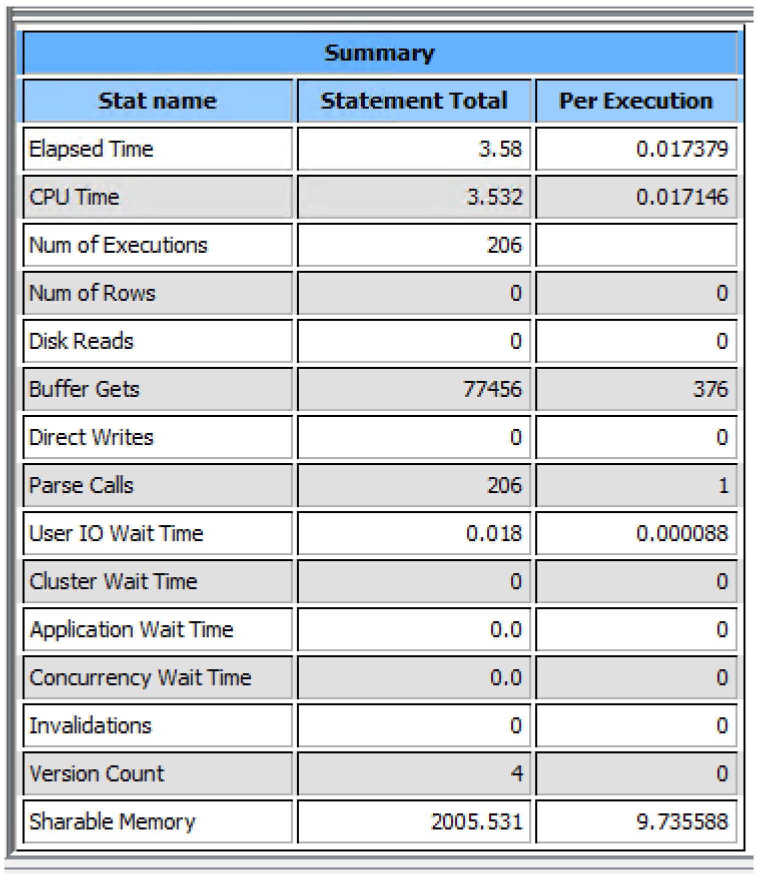


NESTED LOOPS
Rows: 30 Bytes: 1.729K Cost: 39

History information			
First Snap ID	First Snap Time	Last Active Time	Cost
1	14-JAN-2020 08:23	14-JAN-2020 11:49	58701

Bind values sample		
Name	Position	Value
:B11	1	null
:B10	2	null
:B9	3	null
:B8	4	null
:B7	5	-1
:B7	6	-1
:B6	7	-1
:B5	8	2
:B5	9	2
:B3	10	3
:B4	11	1
:B4	12	1
:B3	13	3
:B1	14	1
:B2	15	-1
:B1	16	1

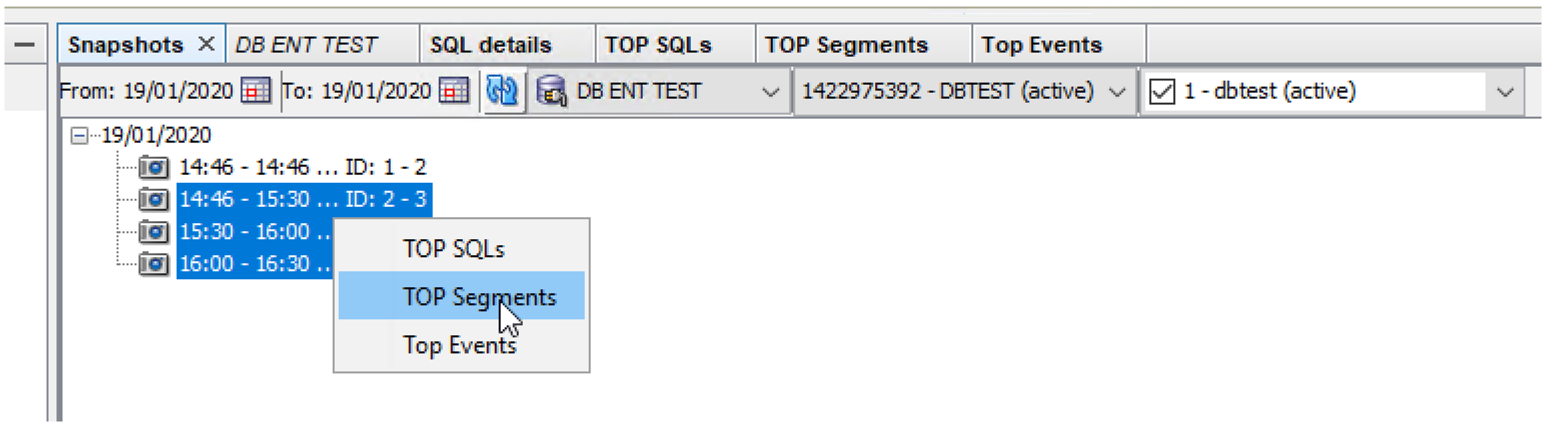
SQL SUMMARY DATA



Summary		
Stat name	Statement Total	Per Execution
Elapsed Time	3.58	0.017379
CPU Time	3.532	0.017146
Num of Executions	206	
Num of Rows	0	0
Disk Reads	0	0
Buffer Gets	77456	376
Direct Writes	0	0
Parse Calls	206	1
User IO Wait Time	0.018	0.000088
Cluster Wait Time	0	0
Application Wait Time	0.0	0
Concurrency Wait Time	0.0	0
Invalidations	0	0
Version Count	4	0
Sharable Memory	2005.531	9.735588

TOP SEGMENTS

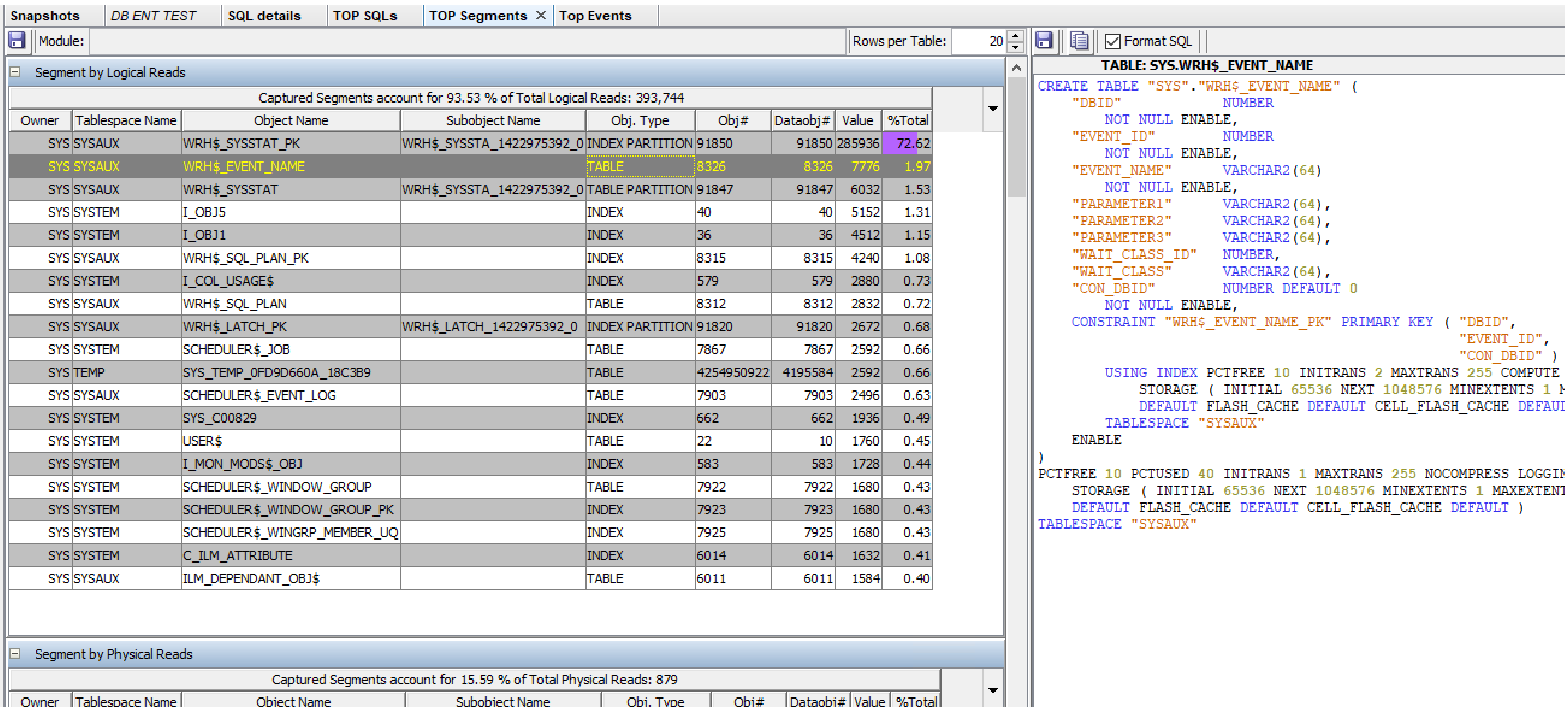
Panel "TOP Segments" is opened from right-click context menu.



From: 19/01/2020 To: 19/01/2020 DB ENT TEST 1422975392 - DBTEST (active) ☑ 1 - dbtest (active)

- 19/01/2020
  - 14:46 - 14:46 … ID: 1 - 2
  - 14:46 - 15:30 … ID: 2 - 3
  - 15:30 - 16:00 ..
  - 16:00 - 16:30 ..

Main panel for TOP Segments



Module:

Rows per Table: 20

☑ Format SQL

Segment by Logical Reads

Captured Segments account for 93.53 % of Total Logical Reads: 393,744

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Obj#	Dataobj#	Value	%Total
SYS	SYSAUX	WRH$_SYSSTAT_PK	WRH$_SYSSTA_1422975392_0	INDEX PARTITION	91850	91850	285936	72.62
SYS	SYSAUX	WRH$_EVENT_NAME		TABLE	8326	8326	7776	1.97
SYS	SYSAUX	WRH$_SYSSTAT	WRH$_SYSSTA_1422975392_0	TABLE PARTITION	91847	91847	6032	1.53
SYS	SYSTEM	I_OBJ5		INDEX	40	40	5152	1.31
SYS	SYSTEM	I_OBJ1		INDEX	36	36	4512	1.15
SYS	SYSAUX	WRH$_SQL_PLAN_PK		INDEX	8315	8315	4240	1.08
SYS	SYSTEM	I_COL_USAGE$		INDEX	579	579	2880	0.73
SYS	SYSAUX	WRH$_SQL_PLAN		TABLE	8312	8312	2832	0.72
SYS	SYSAUX	WRH$_LATCH_PK	WRH$_LATCH_1422975392_0	INDEX PARTITION	91820	91820	2672	0.68
SYS	SYSTEM	SCHEDULER$_JOB		TABLE	7867	7867	2592	0.66
SYS	TEMP	SYS_TEMP_0FD9D660A_18C3B9		TABLE	4254950922	4195584	2592	0.66
SYS	SYSAUX	SCHEDULER$_EVENT_LOG		TABLE	7903	7903	2496	0.63
SYS	SYSTEM	SYS_C00829		INDEX	662	662	1936	0.49
SYS	SYSTEM	USER$		TABLE	22	10	1760	0.45
SYS	SYSTEM	I_MON_MODS$_OBJ		INDEX	583	583	1728	0.44
SYS	SYSTEM	SCHEDULER$_WINDOW_GROUP		TABLE	7922	7922	1680	0.43
SYS	SYSTEM	SCHEDULER$_WINDOW_GROUP_PK		INDEX	7923	7923	1680	0.43
SYS	SYSTEM	SCHEDULER$_WINGRP_MEMBER_UQ		INDEX	7925	7925	1680	0.43
SYS	SYSTEM	C_ILM_ATTRIBUTE		INDEX	6014	6014	1632	0.41
SYS	SYSAUX	ILM_DEPENDANT_OBJ$		TABLE	6011	6011	1584	0.40

Segment by Physical Reads

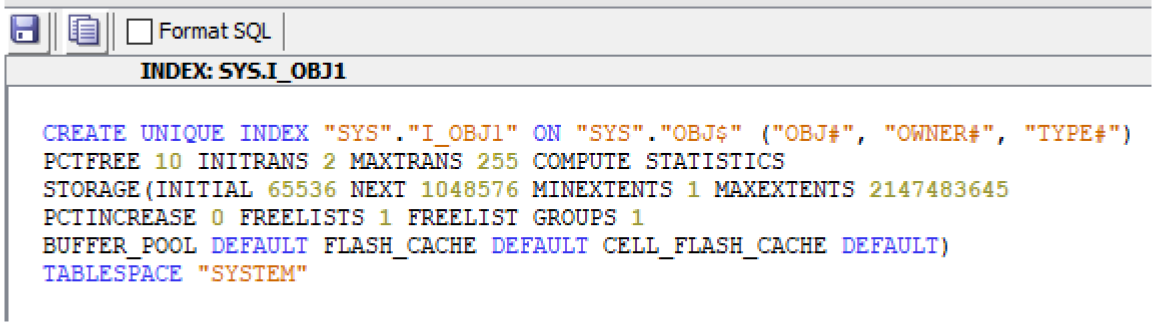
Captured Segments account for 15.59 % of Total Physical Reads: 879

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Obj#	Dataobj#	Value	%Total

TABLE: SYS.WRH$_EVENT_NAME

```
CREATE TABLE "SYS"."WRH$_EVENT_NAME" (
    "DBID"             NUMBER
        NOT NULL ENABLE,
    "EVENT_ID"         NUMBER
        NOT NULL ENABLE,
    "EVENT_NAME"       VARCHAR2(64)
        NOT NULL ENABLE,
    "PARAMETER1"       VARCHAR2(64),
    "PARAMETER2"       VARCHAR2(64),
    "PARAMETER3"       VARCHAR2(64),
    "WAIT_CLASS_ID"    NUMBER,
    "WAIT_CLASS"       VARCHAR2(64),
    "CON_DBID"         NUMBER DEFAULT 0
        NOT NULL ENABLE,
    CONSTRAINT "WRH$_EVENT_NAME_PK" PRIMARY KEY ( "DBID",
                                                  "EVENT_ID",
                                                  "CON_DBID" )
        USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
            STORAGE ( INITIAL 65536 NEXT 1048576 MINEXTENTS 1 M
            DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAUL
        TABLESPACE "SYSAUX"
    ENABLE
)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGIN
    STORAGE ( INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENT
    DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT )
TABLESPACE "SYSAUX"
```

After some row is selected, DDL of selected object is displayed in the detail frame on the right side of the screen



□ Format SQL

INDEX: SYS.I_OBJ1

```
CREATE UNIQUE INDEX "SYS"."I_OBJ1" ON "SYS"."OBJ$" ("OBJ#", "OWNER#", "TYPE#")
PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "SYSTEM"
```

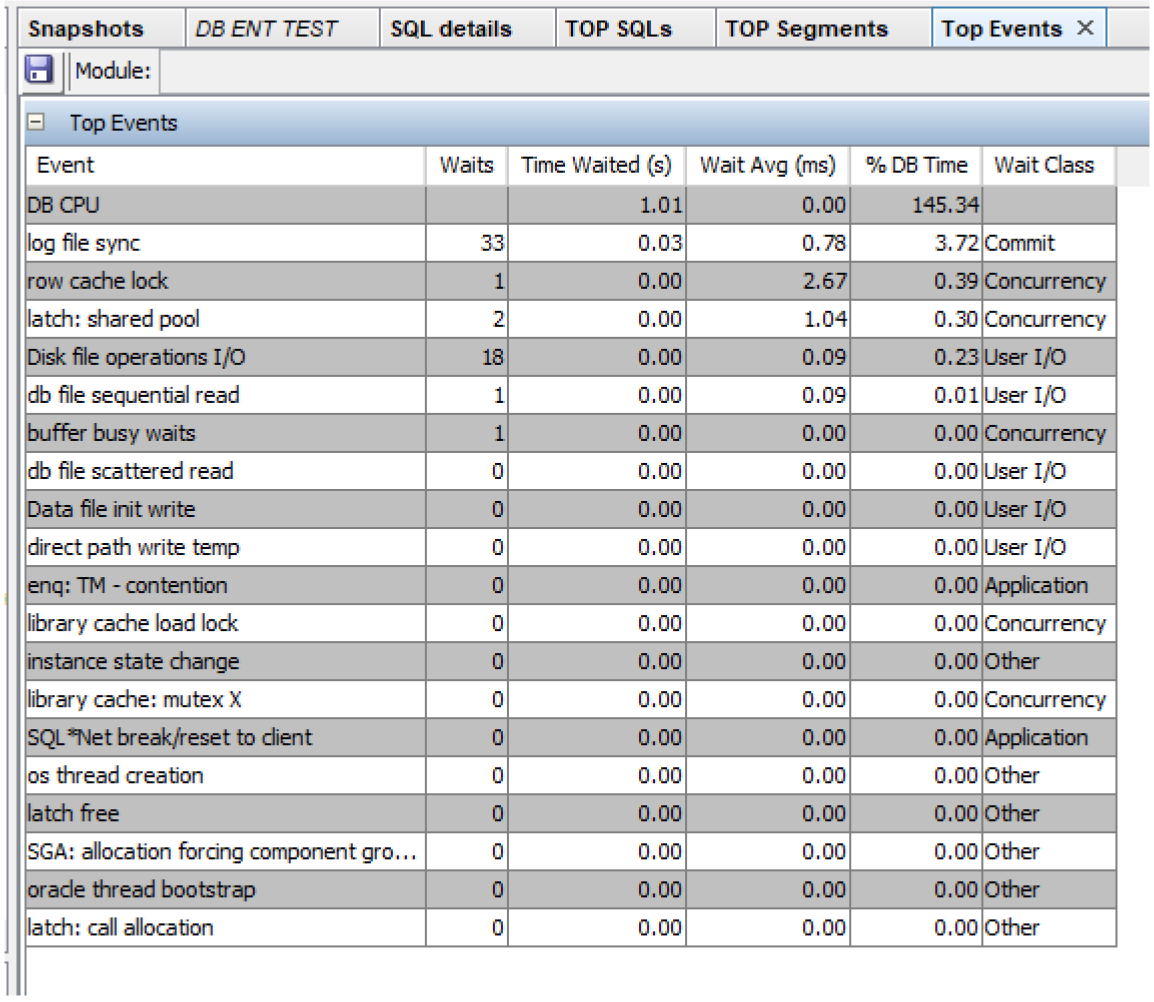
It is possible to save DDL as PDF by clicking "Save" button.

segmentDetails_SYS-I_OBJ5-INDEX.pdf

exported_tables_2_5.xlsx

TOP EVENTS

Main panel "TOP Events" with data within the time interval snap_from and snap_to:



Event	Waits	Time Waited (s)	Wait Avg (ms)	% DB Time	Wait Class
DB CPU		1.01	0.00	145.34	
log file sync	33	0.03	0.78	3.72	Commit
row cache lock	1	0.00	2.67	0.39	Concurrency
latch: shared pool	2	0.00	1.04	0.30	Concurrency
Disk file operations I/O	18	0.00	0.09	0.23	User I/O
db file sequential read	1	0.00	0.09	0.01	User I/O
buffer busy waits	1	0.00	0.00	0.00	Concurrency
db file scattered read	0	0.00	0.00	0.00	User I/O
Data file init write	0	0.00	0.00	0.00	User I/O
direct path write temp	0	0.00	0.00	0.00	User I/O
enq: TM - contention	0	0.00	0.00	0.00	Application
library cache load lock	0	0.00	0.00	0.00	Concurrency
instance state change	0	0.00	0.00	0.00	Other
library cache: mutex X	0	0.00	0.00	0.00	Concurrency
SQL*Net break/reset to client	0	0.00	0.00	0.00	Application
os thread creation	0	0.00	0.00	0.00	Other
latch free	0	0.00	0.00	0.00	Other
SGA: allocation forcing component gro…	0	0.00	0.00	0.00	Other
oracle thread bootstrap	0	0.00	0.00	0.00	Other
latch: call allocation	0	0.00	0.00	0.00	Other

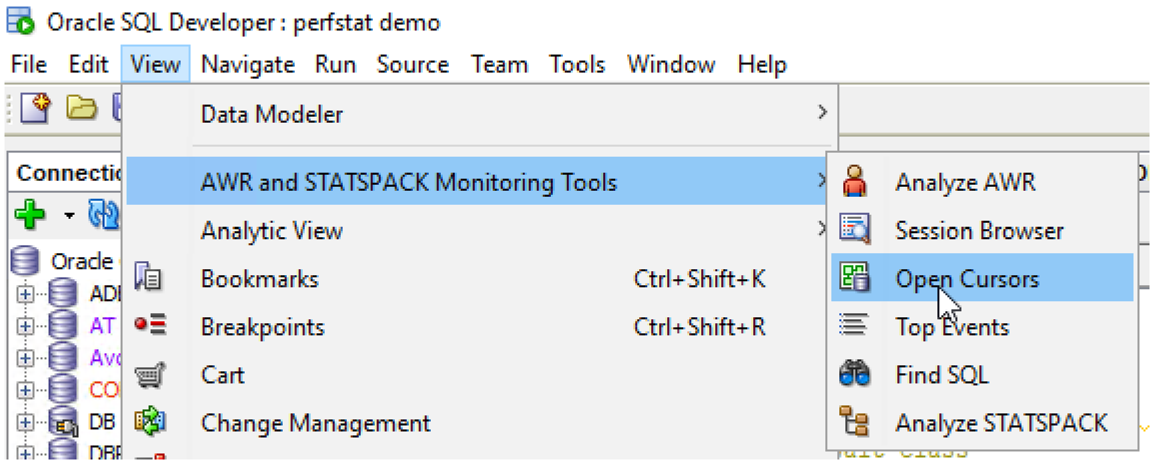
Note: Top Events panel doesn't contain detail SQLs.

OPEN CURSORS

Functionality of Open Cursors allows user to identify the cause of error „ORA-01000: maximum open cursors exceeded". This error is usually solved by increasing parameter OPEN_CURSOR without investigating the real cause which can lead to increased consumption of the system resources.

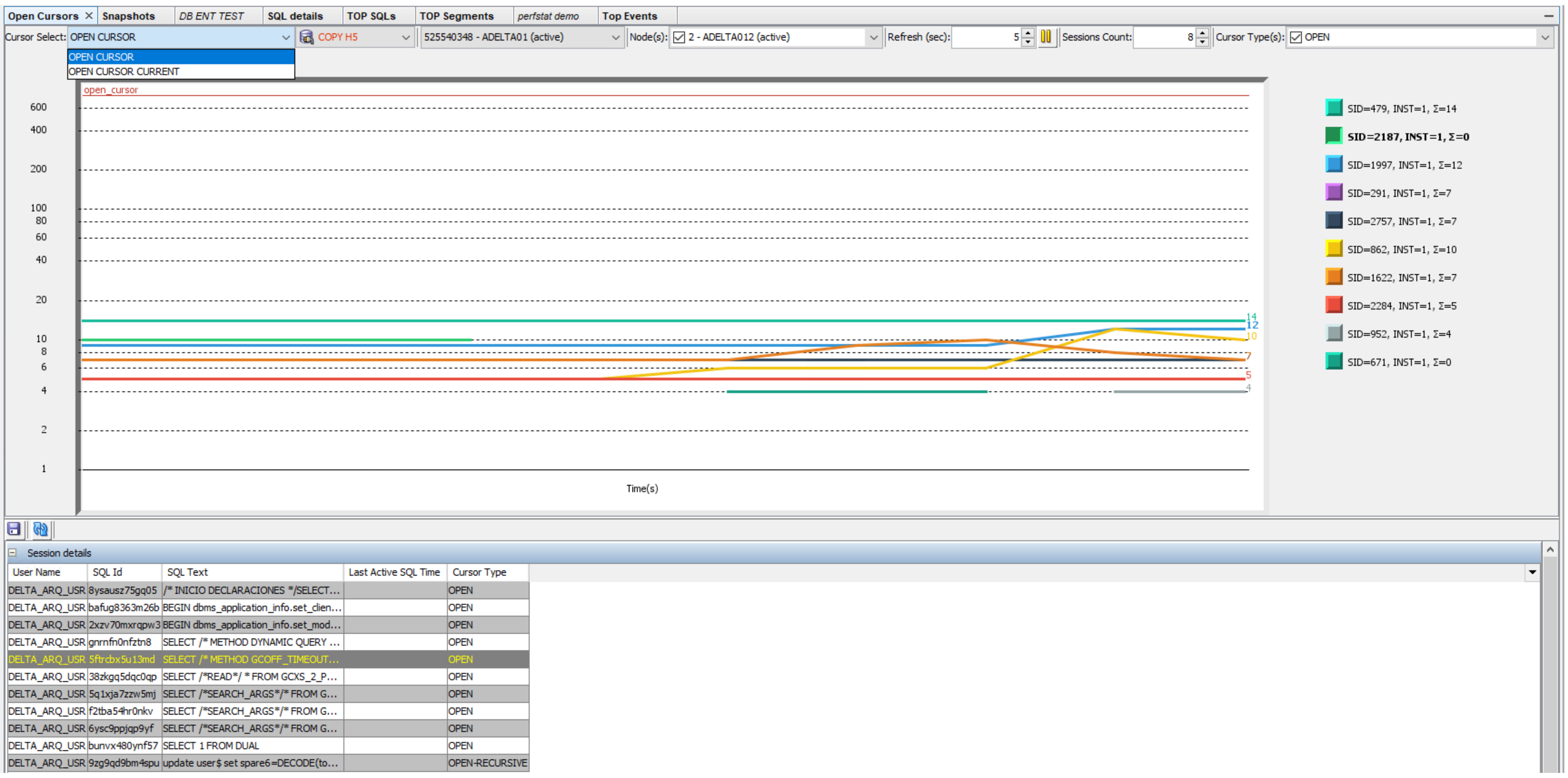
This error is often caused by keeping cursors opened even if are not used any more (e.g. in JAVA). With OPEN CURSORS feature is possible to identify sessions which have most opened cursors.

Open cursors panel is opened from "View"-> "AWR and STATSPACK Monitoring Tools" -> "Open Cursors" menu



MAIN PANEL

Panel displays TOP Sessions by number of opened cursors in the graphical form on the logarithmic scale where sessions are displayed in different colors. In the graph there is also displayed value of parameter OPEN_CURSOR. Reaching this value will raise the error ORA-01000. The graph legend contains identification data like session ID and instance ID. Details about the running SQL queries of the particular session can be displayed by clicking the button of the corresponding color.



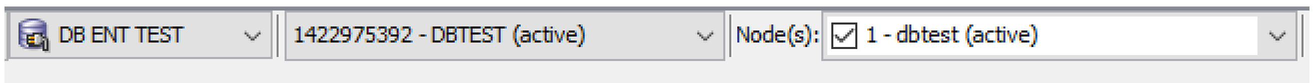
Cursor Select: OPEN CURSOR | COPY H5 | 525540348 - ADELTA01 (active) | Node(s): ☑ 2 - ADELTA012 (active) | Refresh (sec): 5 | Sessions Count: 8 | Cursor Type(s): ☑ OPEN

Session details

User Name	SQL Id	SQL Text	Last Active SQL Time	Cursor Type
DELTA_ARQ_USR	8ysausz75gq05	/* INICIO DECLARACIONES */SELECT...		OPEN
DELTA_ARQ_USR	bafug8363m26b	BEGIN dbms_application_info.set_clien...		OPEN
DELTA_ARQ_USR	2xzv70mxrqpw3	BEGIN dbms_application_info.set_mod...		OPEN
DELTA_ARQ_USR	gnrnfn0nfztn8	SELECT /* METHOD DYNAMIC QUERY ...		OPEN
DELTA_ARQ_USR	5ftrcbx5u13md	SELECT /* METHOD GCOFF_TIMEOUT...		OPEN
DELTA_ARQ_USR	38zkgq5dqc0qp	SELECT /*READ*/ * FROM GCXS_2_P...		OPEN
DELTA_ARQ_USR	5q1xja7zzw5mj	SELECT /*SEARCH_ARGS*/* FROM G...		OPEN
DELTA_ARQ_USR	f2tba54hr0nkv	SELECT /*SEARCH_ARGS*/* FROM G...		OPEN
DELTA_ARQ_USR	6ysc9ppjqp9yf	SELECT /*SEARCH_ARGS*/* FROM G...		OPEN
DELTA_ARQ_USR	bunvx480ynf57	SELECT 1 FROM DUAL		OPEN
DELTA_ARQ_USR	9zg9qd9bm4spu	update user$ set spare6=DECODE(to...		OPEN-RECURSIVE

SELECTION CRITERIA

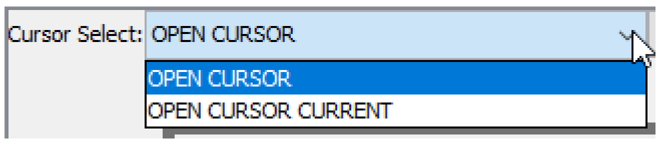
Top bar provides user with the selection criteria for the database connection (OPEN CURSORS monitoring is real-time online process therefore it is not working with the imported AWR/STATSPACK data).



DB ENT TEST | 1422975392 - DBTEST (active) | Node(s): ☑ 1 - dbtest (active)

Selection criteria allows working with the one (single instance or one instance of RAC) or more nodes of database RAC.

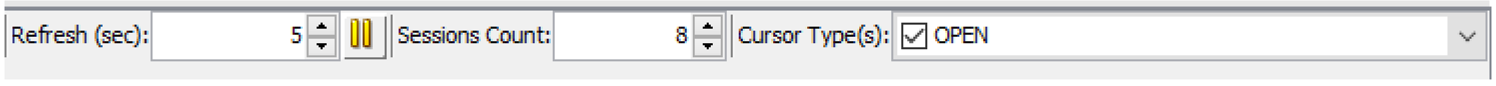
On the left side of the bar there is a combo-box for selection of the type of Cursor Select



Cursor Select: OPEN CURSOR

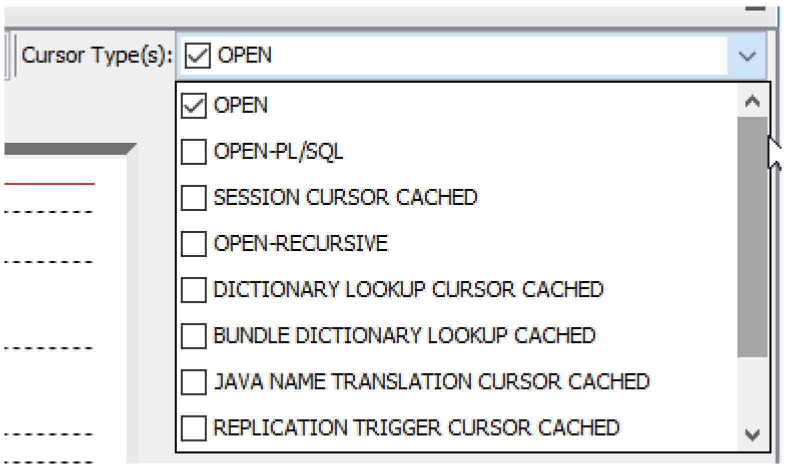
- OPEN CURSOR
- OPEN CURSOR CURRENT

On the right side of the bar there are more criteria for Refresh (refresh rate in seconds) with "Pause" button, Sessions Count (number of monitored sessions) and Cursor Type(s).



Refresh (sec): 5 | Sessions Count: 8 | Cursor Type(s): ☑ OPEN

Types of Cursors:

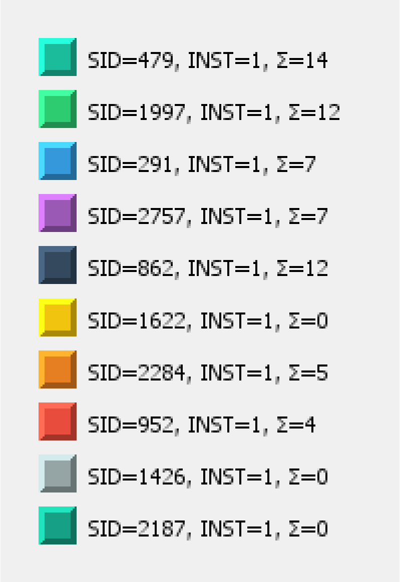


Cursor Type(s): ☑ OPEN

- ☑ OPEN
- ☐ OPEN-PL/SQL
- ☐ SESSION CURSOR CACHED
- ☐ OPEN-RECURSIVE
- ☐ DICTIONARY LOOKUP CURSOR CACHED
- ☐ BUNDLE DICTIONARY LOOKUP CACHED
- ☐ JAVA NAME TRANSLATION CURSOR CACHED
- ☐ REPLICATION TRIGGER CURSOR CACHED

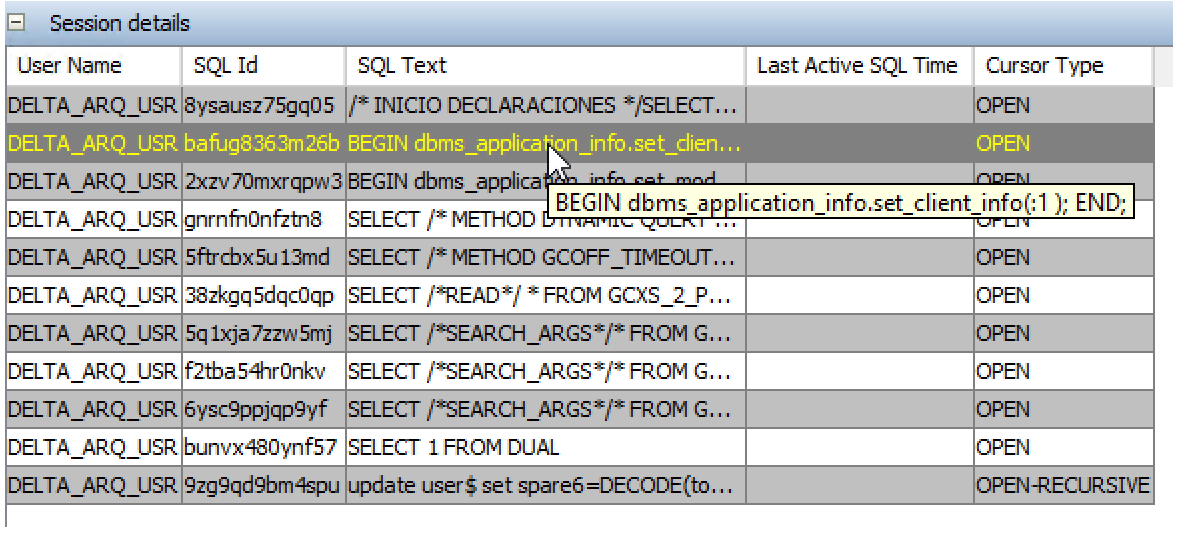
PANEL TOP SESSIONS

Graph legend example:



- 🟩 SID=479, INST=1, Σ=14
- 🟩 SID=1997, INST=1, Σ=12
- 🟦 SID=291, INST=1, Σ=7
- 🟪 SID=2757, INST=1, Σ=7
- ⬛ SID=862, INST=1, Σ=12
- 🟨 SID=1622, INST=1, Σ=0
- 🟧 SID=2284, INST=1, Σ=5
- 🟥 SID=952, INST=1, Σ=4
- ⬜ SID=1426, INST=1, Σ=0
- 🟩 SID=2187, INST=1, Σ=0

When the one of the buttons in the graph legend is clicked, bottom section is populated with the list of all running SQL queries within the selected session. Queries can be exported and details of the SQL can be displayed after double-click (SQL text is also displayed as a tooltip when the mouse pointer is moved over the selected row).

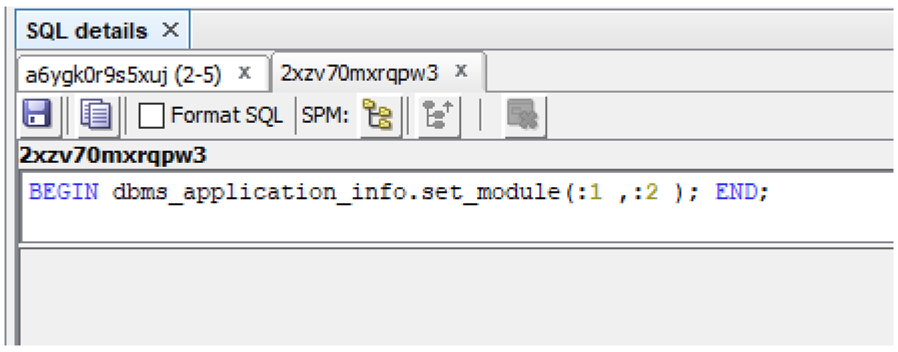


Session details

User Name	SQL Id	SQL Text	Last Active SQL Time	Cursor Type
DELTA_ARQ_USR	8ysausz75gq05	/* INICIO DECLARACIONES */SELECT…		OPEN
DELTA_ARQ_USR	bafug8363m26b	BEGIN dbms_application_info.set_clien…		OPEN
DELTA_ARQ_USR	2xzv70mxrqpw3	BEGIN dbms_application_info.set_mod…		OPEN
DELTA_ARQ_USR	gnrnfn0nfztn8	SELECT /* METHOD DYNAMIC QUERY …		OPEN
DELTA_ARQ_USR	5ftrcbx5u13md	SELECT /* METHOD GCOFF_TIMEOUT…		OPEN
DELTA_ARQ_USR	38zkgq5dqc0qp	SELECT /*READ*/ * FROM GCXS_2_P…		OPEN
DELTA_ARQ_USR	5q1xja7zzw5mj	SELECT /*SEARCH_ARGS*/* FROM G…		OPEN
DELTA_ARQ_USR	f2tba54hr0nkv	SELECT /*SEARCH_ARGS*/* FROM G…		OPEN
DELTA_ARQ_USR	6ysc9ppjqp9yf	SELECT /*SEARCH_ARGS*/* FROM G…		OPEN
DELTA_ARQ_USR	bunvx480ynf57	SELECT 1 FROM DUAL		OPEN
DELTA_ARQ_USR	9zg9qd9bm4spu	update user$ set spare6=DECODE(to…		OPEN-RECURSIVE

BEGIN dbms_application_info.set_client_info(:1); END;

Detail on double-click:



2xzv70mxrqpw3

```
BEGIN dbms_application_info.set_module(:1 ,:2 ); END;
```

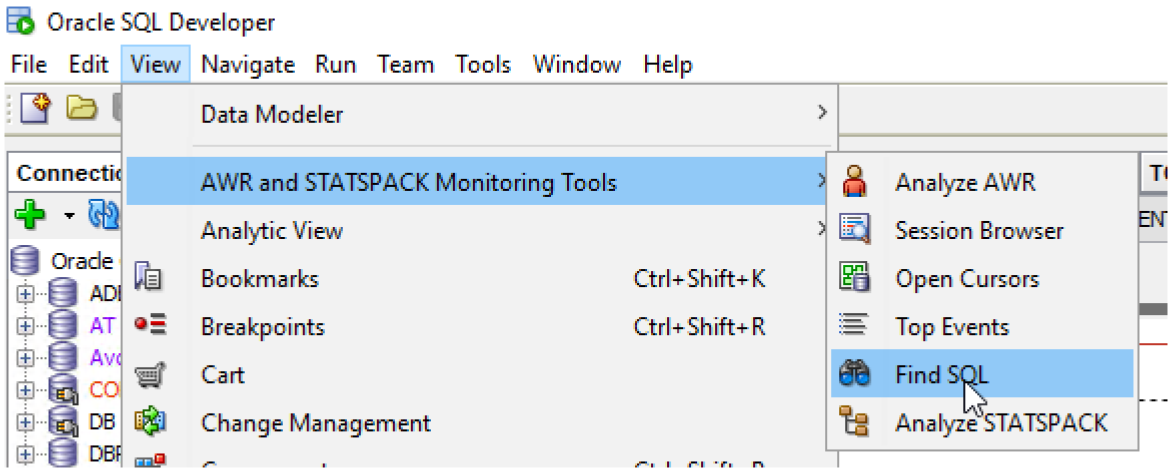
Note: Problematic opened cursors can be detected by monitoring the session with the increasing number of open cursors with the repeating SQL_ID in the details. That indicates repetitive opening of the cursor with the same SQL query. Good manner is to export all SQLs of the monitored session and compare manually because it is possible that there are more same SQLs with different SQL_ID (or the "Hard Parse" problem when the variables are not sent as parameters but are hardcoded in the SQL query).

FIND SQL

Tool for finding particular SQL query by various criteria (Module, SQL text, Execution Plan hash value).

Panel is designed for searching by criteria so non-selective criteria can cause performance problems.

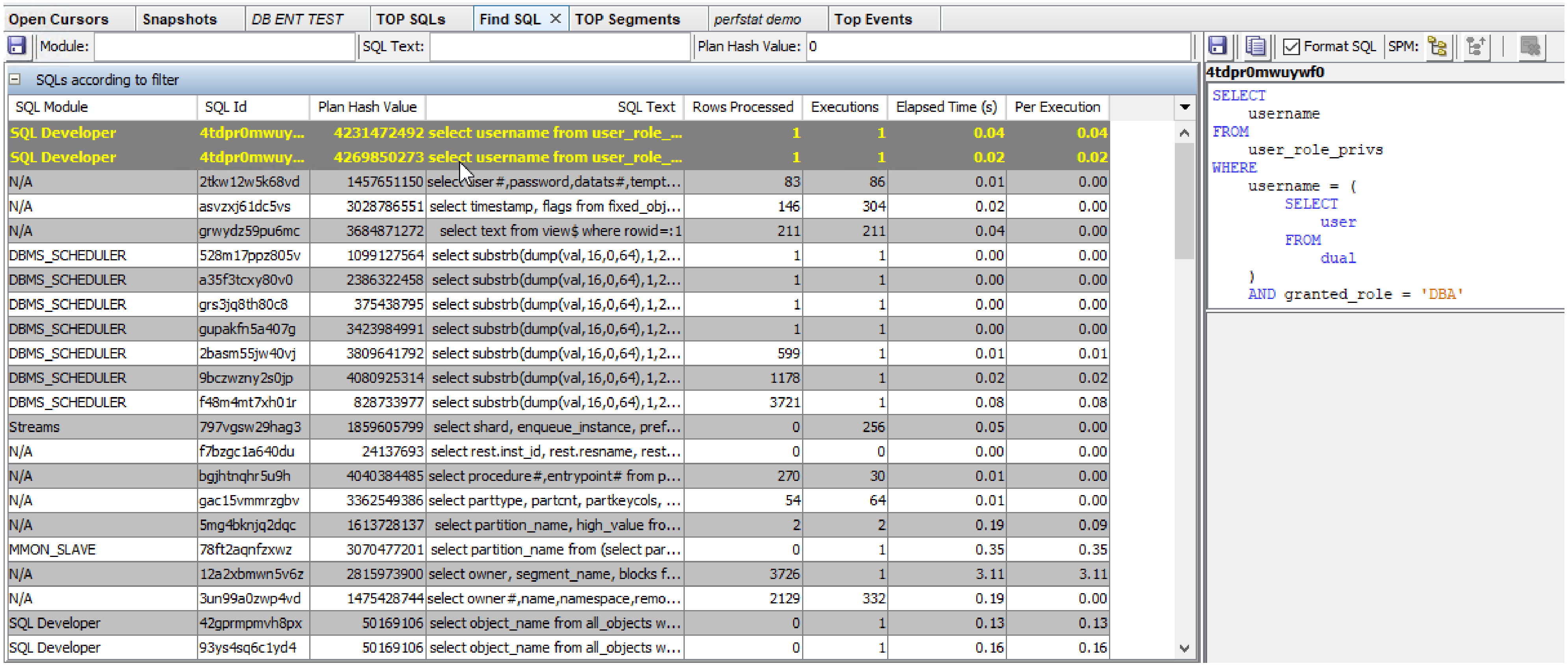
Panel Find SQL is opened from "View" -> "AWR and STATSPACK Monitoring Tools" -> "Find SQL" menu



Panel displays basic data about found SQL queries:

- Number of processed rows
- Number of executions
- Elapsed time
- Average time per execution

Details of the SQL query are displayed after click on the selected row. SQL text can be formatted and its plans can be uploaded with SPM functionality (described above).



Module: SQL Text: Plan Hash Value: 0

SQLs according to filter

SQL Module	SQL Id	Plan Hash Value	SQL Text	Rows Processed	Executions	Elapsed Time (s)	Per Execution
SQL Developer	4tdpr0mwuy...	4231472492	select username from user_role_...	1	1	0.04	0.04
SQL Developer	4tdpr0mwuy...	4269850273	select username from user_role_...	1	1	0.02	0.02
N/A	2tkw12w5k68vd	1457651150	select user#,password,datats#,tempt...	83	86	0.01	0.00
N/A	asvzxj61dc5vs	3028786551	select timestamp, flags from fixed_obj...	146	304	0.02	0.00
N/A	grwydz59pu6mc	3684871272	select text from view$ where rowid=:1	211	211	0.04	0.00
DBMS_SCHEDULER	528m17ppz805v	1099127564	select substrb(dump(val,16,0,64),1,2...	1	1	0.00	0.00
DBMS_SCHEDULER	a35f3tcxy80v0	2386322458	select substrb(dump(val,16,0,64),1,2...	1	1	0.00	0.00
DBMS_SCHEDULER	grs3jq8th80c8	375438795	select substrb(dump(val,16,0,64),1,2...	1	1	0.00	0.00
DBMS_SCHEDULER	gupakfn5a407g	3423984991	select substrb(dump(val,16,0,64),1,2...	1	1	0.00	0.00
DBMS_SCHEDULER	2basm55jw40vj	3809641792	select substrb(dump(val,16,0,64),1,2...	599	1	0.01	0.01
DBMS_SCHEDULER	9bczwzny2s0jp	4080925314	select substrb(dump(val,16,0,64),1,2...	1178	1	0.02	0.02
DBMS_SCHEDULER	f48m4mt7xh01r	828733977	select substrb(dump(val,16,0,64),1,2...	3721	1	0.08	0.08
Streams	797vgsw29hag3	1859605799	select shard, enqueue_instance, pref...	0	256	0.05	0.00
N/A	f7bzgc1a640du	24137693	select rest.inst_id, rest.resname, rest...	0	0	0.00	0.00
N/A	bgjhtnqhr5u9h	4040384485	select procedure#,entrypoint# from p...	270	30	0.01	0.00
N/A	gac15vmmrzgbv	3362549386	select parttype, partcnt, partkeycols, ...	54	64	0.01	0.00
N/A	5mg4bknjq2dqc	1613728137	select partition_name, high_value fro...	2	2	0.19	0.09
MMON_SLAVE	78ft2aqnfzxwz	3070477201	select partition_name from (select par...	0	1	0.35	0.35
N/A	12a2xbmwn5v6z	2815973900	select owner, segment_name, blocks f...	3726	1	3.11	3.11
N/A	3un99a0zwp4vd	1475428744	select owner#,name,namespace,remo...	2129	332	0.19	0.00
SQL Developer	42gprmpmvh8px	50169106	select object_name from all_objects w...	0	1	0.13	0.13
SQL Developer	93ys4sq6c1yd4	50169106	select object_name from all_objects w...	0	1	0.16	0.16

4tdpr0mwuywf0

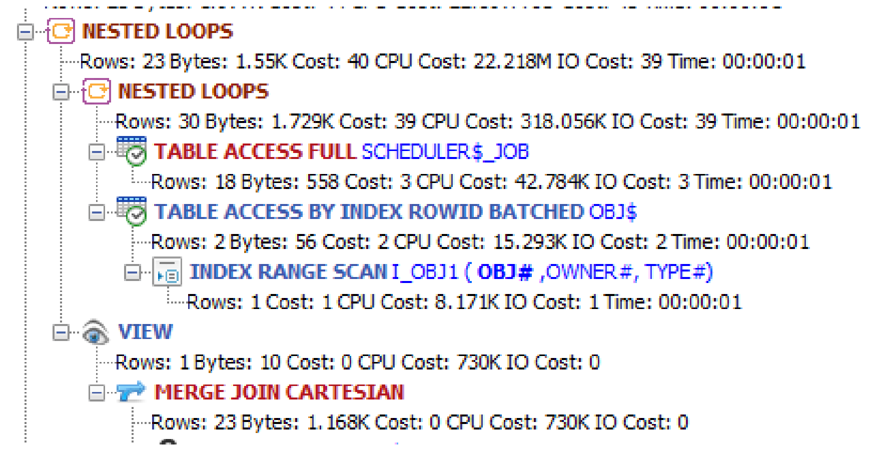
```
SELECT
    username
FROM
    user_role_privs
WHERE
    username = (
        SELECT
            user
        FROM
            dual
    )
    AND granted_role = 'DBA'
```

APPENDIX

EXAMPLES OF EXECUTION PLANS AND THE WAY HOW TO INTERPRET IT

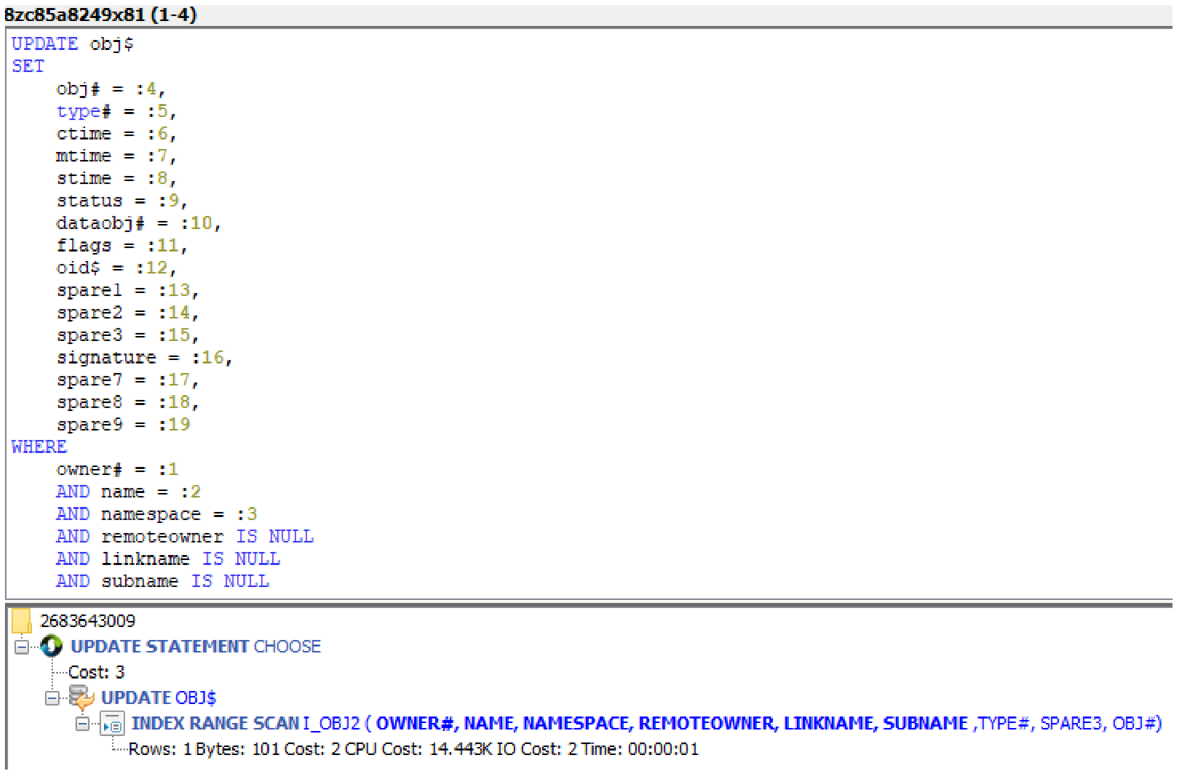
In the Execution Plan are the details of all steps highlighted with different colors and also columns used for RANGE SCAN are highlighted.

Example 1 - in the row with INDEX RANGE SCAN is highlighted only first column which means that range scan used only first column of the index. This feature helps user to identify the problems related to the use of the wrong index or the wrong index structure.



```
NESTED LOOPS
  Rows: 23 Bytes: 1.55K Cost: 40 CPU Cost: 22.218M IO Cost: 39 Time: 00:00:01
  NESTED LOOPS
    Rows: 30 Bytes: 1.729K Cost: 39 CPU Cost: 318.056K IO Cost: 39 Time: 00:00:01
    TABLE ACCESS FULL SCHEDULER$_JOB
      Rows: 18 Bytes: 558 Cost: 3 CPU Cost: 42.784K IO Cost: 3 Time: 00:00:01
    TABLE ACCESS BY INDEX ROWID BATCHED OBJ$
      Rows: 2 Bytes: 56 Cost: 2 CPU Cost: 15.293K IO Cost: 2 Time: 00:00:01
      INDEX RANGE SCAN I_OBJ1 ( OBJ# ,OWNER#, TYPE#)
        Rows: 1 Cost: 1 CPU Cost: 8.171K IO Cost: 1 Time: 00:00:01
  VIEW
    Rows: 1 Bytes: 10 Cost: 0 CPU Cost: 730K IO Cost: 0
    MERGE JOIN CARTESIAN
      Rows: 23 Bytes: 1.168K Cost: 0 CPU Cost: 730K IO Cost: 0
```

Example 2 illustrates how many columns in the index were used:

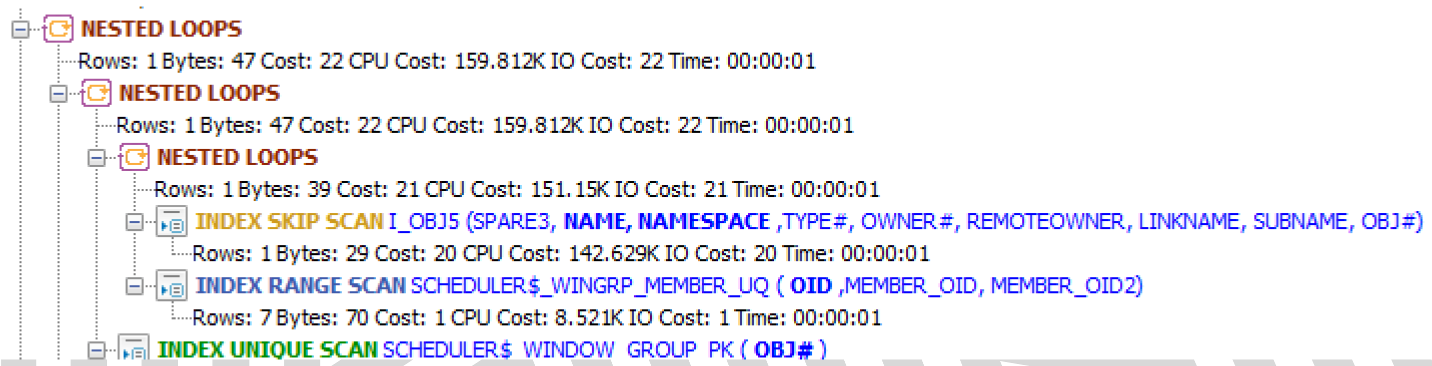


8zc85a8249x81 (1-4)

```
UPDATE obj$
SET
    obj# = :4,
    type# = :5,
    ctime = :6,
    mtime = :7,
    stime = :8,
    status = :9,
    dataobj# = :10,
    flags = :11,
    oid$ = :12,
    spare1 = :13,
    spare2 = :14,
    spare3 = :15,
    signature = :16,
    spare7 = :17,
    spare8 = :18,
    spare9 = :19
WHERE
    owner# = :1
    AND name = :2
    AND namespace = :3
    AND remoteowner IS NULL
    AND linkname IS NULL
    AND subname IS NULL
```

```
2683643009
UPDATE STATEMENT CHOOSE
  Cost: 3
  UPDATE OBJ$
    INDEX RANGE SCAN I_OBJ2 ( OWNER#, NAME, NAMESPACE, REMOTEOWNER, LINKNAME, SUBNAME ,TYPE#, SPARE3, OBJ#)
      Rows: 1 Bytes: 101 Cost: 2 CPU Cost: 14.443K IO Cost: 2 Time: 00:00:01
```

Example 3 illustrates which columns were used by INDEX SKIP SCAN



```
NESTED LOOPS
  Rows: 1 Bytes: 47 Cost: 22 CPU Cost: 159.812K IO Cost: 22 Time: 00:00:01
  NESTED LOOPS
    Rows: 1 Bytes: 47 Cost: 22 CPU Cost: 159.812K IO Cost: 22 Time: 00:00:01
    NESTED LOOPS
      Rows: 1 Bytes: 39 Cost: 21 CPU Cost: 151.15K IO Cost: 21 Time: 00:00:01
      INDEX SKIP SCAN I_OBJ5 (SPARE3, NAME, NAMESPACE ,TYPE#, OWNER#, REMOTEOWNER, LINKNAME, SUBNAME, OBJ#)
        Rows: 1 Bytes: 29 Cost: 20 CPU Cost: 142.629K IO Cost: 20 Time: 00:00:01
      INDEX RANGE SCAN SCHEDULER$_WINGRP_MEMBER_UQ ( OID ,MEMBER_OID, MEMBER_OID2)
        Rows: 7 Bytes: 70 Cost: 1 CPU Cost: 8.521K IO Cost: 1 Time: 00:00:01
    INDEX UNIQUE SCAN SCHEDULER$_WINDOW_GROUP_PK ( OBJ# )
```