**SQL Monitoring using SQL Tuning Sets**

The scope of this project is to build a SQL tracing toolkit that will help capture and store SQL workload information of PSR / QA / Dev tests.

The main objective is to

* Support PSR/Dev/QA with a toolset to capture SQLs and facilitate them to do their own first level analysis on SQL performance and trending.
* Maintain data for SQL analysis/tuning.
* Be able to track slow SQLs more efficiently and file defects for the same.

This is implemented using the SQL Tuning Set framework.

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=2)]

**What is a SQL Tuning Set?**

A SQL Tuning Set (STS) is a database object used for capturing SQL workload information. It contains:

* One or more SQL statements.
* Associated execution context such as parsing schema name and list of bind values.
* Associated execution statistics such as elapsed time, execution count, CPU time, buffer gets, disk reads, etc.
* Associated execution plans.

With the STS framework, users can capture any SQL statements of interest and store them in an STS for future tuning. Also, an STS can be transported from one instance to another.

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=3)]

**How does it work?**

The SQL Monitoring framework is implemented as a set of procedures and tables around the STS framework provided by the database. An initial setup script is to be run on the database schema to setup the required package and tables. Once setup, the API can be used to capture SQLs and save them as SQL sets.

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=4)]

**How to install?**

Login as SYSDBA and run the sqlrep\_setup.sql script

SQL> @sqlrep\_setup

SQL Monitoring Toolkit setup. ver 4.1.1

Schema Name (SQLREP) :

Password (SQLREP) :

Default Tablespace Name (FUSION\_TS\_TOOLS) :

Temporary Tablespace Name (TEMP) :

…

Setup completed sucessfully

SQL>

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=5)]

**Steps to capture SQLs**

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=6)]

**Automated creation of STS**

To turn on automated capture (To schedule automatic creation of new STS every 12 hours and capture SQLs into it), use the **sqlrep.schedule\_capture** procedure. This script will submit a DB scheduler job that will wake up at 7AM and 7PM. You can change or modify the time by passing the JOB\_REPEAT\_INTERVAL parameter.  
**Parameters**

PARSING\_SCHEMA -> Can specify multiple schemas. ex:'FUSION,ORAESS'

MODULE

ACTION

CAPTURE\_MAX\_TIME\_LIMIT -> Default 24 hrs (86400s)

CAPTURE\_TIME\_INTERVAL -> Default 1 min (180s)

JOB\_REPEAT\_INTERVAL -> Default 'FREQ=HOURLY; BYHOUR=7,19'

**Example**

SQL> exec sqlrep.schedule\_capture (PARSING\_SCHEMA =>'FUSION,ORAESS,FUSION2');

or

SQL> exec sqlrep.schedule\_capture (PARSING\_SCHEMA =>'FUSION,ORAESS,FUSION2',

job\_repeat\_interval => 'FREQ=HOURLY', capture\_max\_time\_limit => 3600);

To stop capture, call the stop\_scheduler procedure with the sql set name.

SQL> exec sqlrep.stop\_scheduler;

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=7)]

**Manual creation of STS**

How to create a new SQL set and start the capture:

SQL> exec SQLREP.Start\_Capture ('FINS\_JAN29', 'FUSION');

or

SQL> exec SQLREP.Start\_Capture ('FINS\_JAN29', 'FUSION,ORAESS');

After this, the SQLs will get captured into the STS until stopped or until the DB is shutdown.

How to stop the SQL capture process:

SQL> exec SQLREP.Stop\_Capture('FINS\_JAN29');

Manual creation of STS in RAC Environments

In RAC enabled databases, when you create an STS manually the tool will create one STS for   
every node in the RAC. For e.g, if there are 3 nodes in a RAC enabled database, and if you   
create an STS with name 'OCTOBER\_WAVE\_DRY\_RUN', the tool will create 3 STS as following:  
OCTOBER\_WAVE\_DRY\_RUN\_1,OCTOBER\_WAVE\_DRY\_RUN\_2,OCTOBER\_WAVE\_DRY\_RUN\_3

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=8)]

**Querying the captured SQL Sets**

select SQLSET\_NAME, START\_TIME, END\_TIME from SQLREP\_SQLSETS order by START\_TIME;

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=9)]

**Generating Reports**

You can generate the SQL Set report as follows:

Connect to the SQLREP user and execute the **sqlreprpt.sql** script. This script will generate an HTML report for the given SQL set. The report will be saved in your current directory as **sql\_report\_<dbname>\_<timestamp>.html**.

Syntax to run report generation script:

SQL> @sqlreprpt.sql FINS\_JAN29 (replace the STS name with appropriate one.)

Report can be customized to accept a list (or) a range of sqlsets. Modify the ‘CONFIG SETTINGS’ section in the beginning of the “sqlreprpt.sql” script:

RAC Enabled environments:

In case of RAC enabled database, there will be multiple STS corresponding to each node.   
Let's say there are 3 nodes in the RAC and the names of STS are as following:  
OCTOBER\_WAVE\_DRY\_RUN\_1,OCTOBER\_WAVE\_DRY\_RUN\_2,OCTOBER\_WAVE\_DRY\_RUN\_3.  
To generate a report out of these 3 STS, you will need to run the command as follows by specifying   
Node1 STS as begin STS and node2 STS as end STS:

SQL> @sqlreprpt.sql OCTOBER\_WAVE\_DRY\_RUN\_1 OCTOBER\_WAVE\_DRY\_RUN\_3 (replace the STS name with appropriate one.)

Report can be customized to accept a list (or) a range of sqlsets. Modify the ‘CONFIG SETTINGS’ section in the beginning of the “sqlreprpt.sql” script:

Function to modify the setting is

sqlrep\_report.modify\_parameters(

parameter\_name => 'CONTEXT\_FILTER',

parameter\_value => q'[ SQLSET\_NAME = '&1' AND PARSING\_SCHEMA\_NAME IN ('FUSION') ]'

);

Parameter names supported are

CONTEXT FILTER - Supported columns are SQLSET\_NAME, PARSING\_SCHEMA\_NAME, COMMAND\_TYPE, MODULE, ACTION, SQL\_ID, SQL\_TEXT

STATISTICS FILTER - Supported columns are EXECUTIONS, ELAPSED\_TIME, CPU\_TIME, BUFFER\_GETS, DISK\_READS, DIRECT\_WRITES, ROWS\_PROCESSED, FETCHES, BUG\_ID, PRODUCT

MAX\_SQLS – Maximum number of SQLs to display

ORDER\_BY – Order in which SQLs are sorted

Look for the following section in the sqlreprpt.sql script

...

-- ------------------------------------ CONFIG SETTINGS ----------------------

-- Change these settings as per the requirement --

sqlrep\_report.modify\_parameters(

parameter\_name => 'CONTEXT\_FILTER',

parameter\_value => q'[ SQLSET\_NAME = '&1' AND PARSING\_SCHEMA\_NAME IN ('FUSION') ]'

);

sqlrep\_report.modify\_parameters(

parameter\_name => 'STATISTICS\_FILTER',

parameter\_value => q'[ ELAPSED\_TIME >= 30 OR (ELAPSED\_TIME >= 5 and EXECUTIONS >=40) OR EXECUTIONS >=50000 ]'

);

sqlrep\_report.modify\_parameters(

parameter\_name => 'MAX\_SQLS',

parameter\_value => 100

);

sqlrep\_report.modify\_parameters(

parameter\_name => 'ORDER\_BY',

parameter\_value => 'ELAPSED\_TIME DESC'

);

-- ------------------------------------------ END ---------------------------- ...

[[edit](http://psr.us.oracle.com/wiki/index.php?title=DBP_FusionApps_POCSTS&action=edit&section=10)]

**Script Download**

* Script : [sqlrep\_scripts.zip](http://psr.us.oracle.com/wiki/resources/b/bd/Sqlrep_scripts.zip)
* Report scripts : [Click the link to download](http://files.oraclecorp.com/content/MySharedFolders/PSR%20DB/Tools/PSR-STS/STS_reports)

If you are installing STS in local testing environments (PSR Lab), you can disable the automated   
scheduler by running the following command:  
  
SQL> exec sqlrep.stop\_scheduler;  
  
 You can capture STS manually using the following command:  
  
 SQL> exec sqlrep.start\_capture('FINS\_JAN29');