Abakus
Ace of disc space!

APPM

User Manual

Table of Contents

Overview

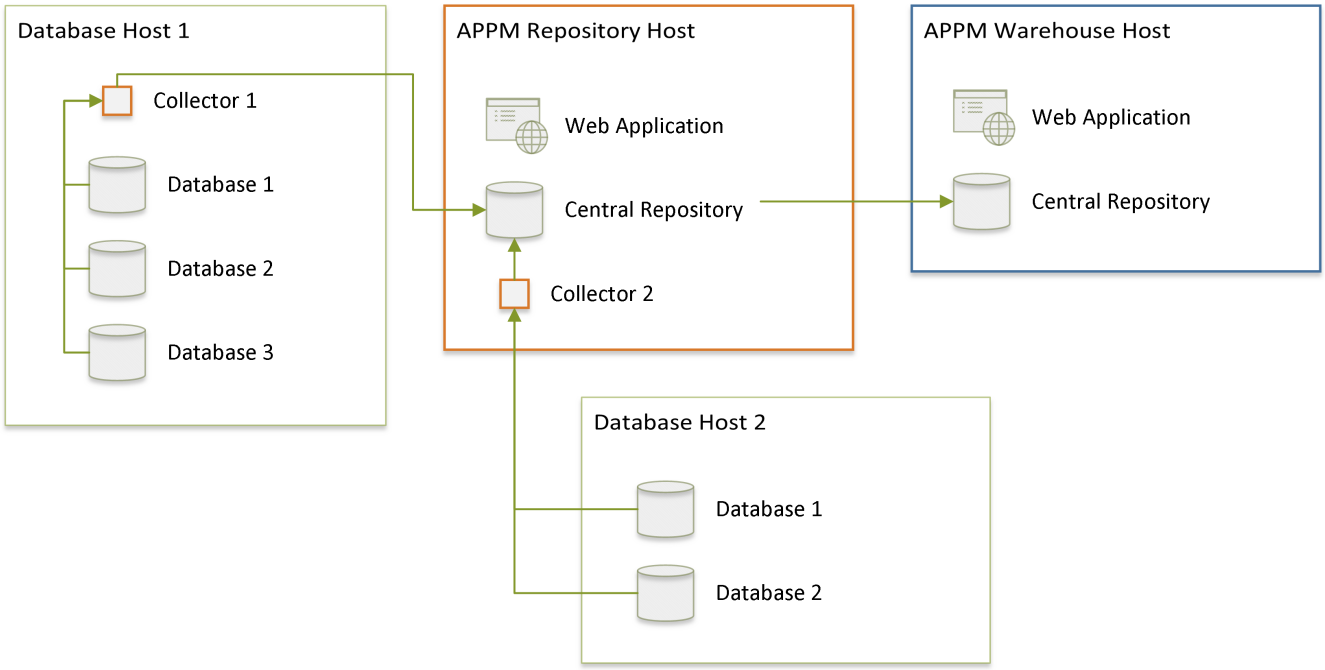
APPM collects and displays Oracle Database historical and current performance related data.

It collects performance data such as wait events and active session history. Collected data is stored in **central repository** and used to identify and analyze performance issues like resource intensive SQL statements.

One of its key features is to conveniently display information required to identify performance issues root causes. It also allows to drill down on identified performance bottlenecks.

It offers similar functionality as Oracle ASH/AWR, but it also works on Standard and Express Editions.

APPM Repository



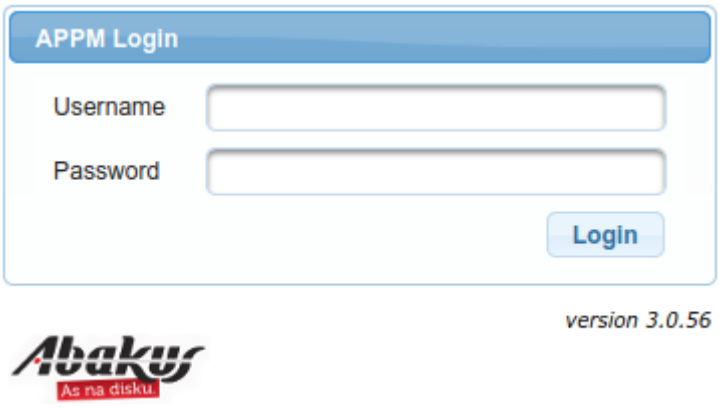
APPM Collector samples monitored database's performance statistics and stores it in APPM Central Repository. Collection of samples has minimal performance impact on monitored databases. Repository database where samples are stored does not need any Oracle license.

APPM Collector can run on the same host as database or it can collect the samples via SQL*Net.

APPM Central Repository is based on docker containers and can run on any host that can run Docker Container. Usually, this is a dedicated virtual machine. It's also possible to deploy it on Oracle Cloud.

Login

Repository Login



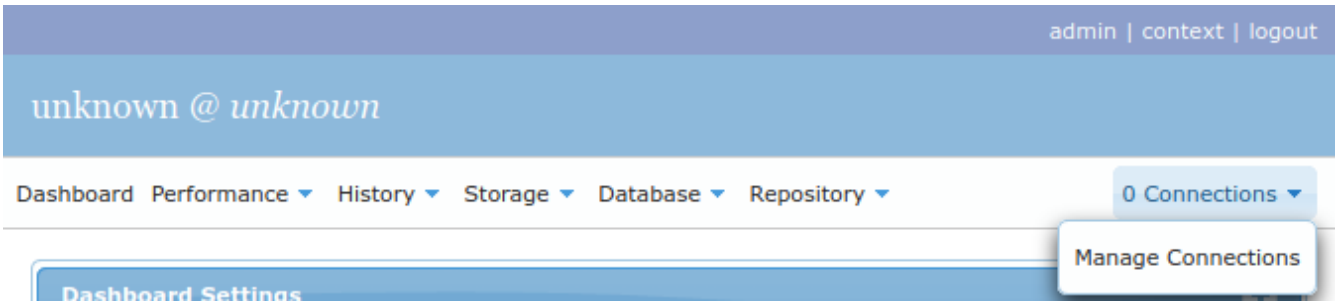
You can access web application interface at:

```
https://<hostname>/appm/
```

Default username and password after installation are:

- Username: `admin`
- Password: `change_me`

Database Login



You may connect to running Oracle instances via SQL*Net to obtain data directly from the database, rather than querying sampled data. You can do so by clicking `n Connections` → `Manage Connections` in upper right corner.

Everything under `Performance`, `Storage` and `Database` menus requires such SQL*Net connection, while all other menus refer only to samples stored in repository database.

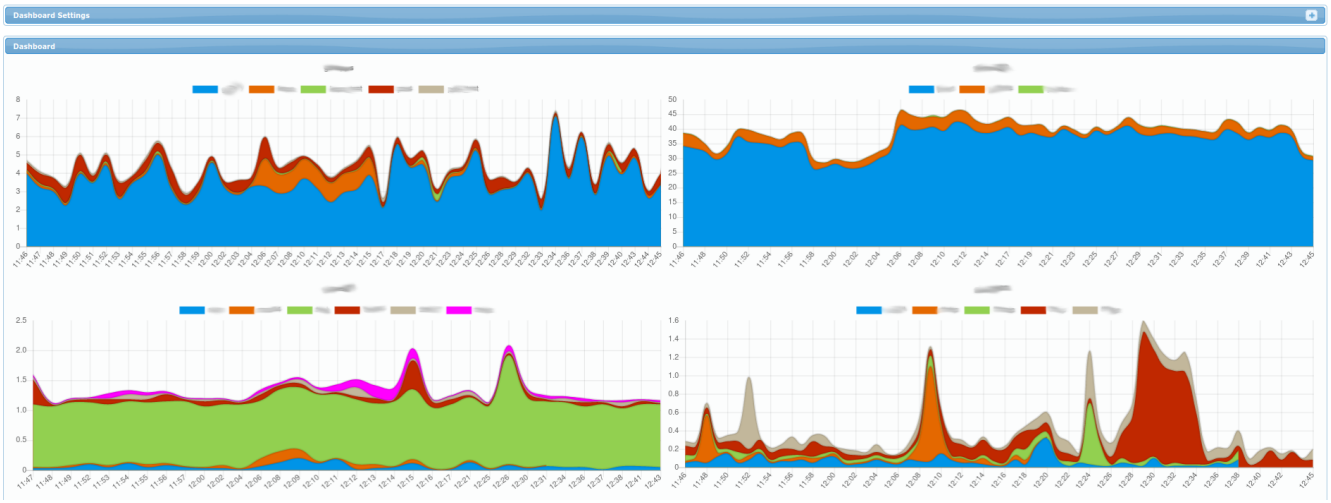
Context Fields

Note that some fields have "blue" labels, such as the ones in the screenshot above (you can open this popup by clicking `context` in the top right corner).

Those *context* fields are **session scoped** and so they persist across the graphical interface among different pages.

History (based on collected samples)

Dashboard



Each chart represents a single server (or cluster of RAC nodes in more advanced setups). Each color represents amount of active sessions for a specific database on that server.

Note that on right top corner, you can click the `+` on the top right to display `Dashboard Settings` panel (it allows you to set time interval for which the charts are displayed - by default for the last hour).

History → Active Sessions

You must choose the database for which to display samples. This is done in the first column using fields:

- `Database` - for which database to display the sampled data
- `Instance` - for which instance to display sampled data (only relevant for RAC setups)
- `From` and `To` - for which period to display sampled data (it defaults to last hour)

In second column of fields, you can choose which database/period you want to compare the results of first column to:

- `Compare Database` - which database to compare results to

- `Compare Instance` - which instance to compare results to (only relevant for RAC setups)
- `Compare From` and `Compare To` - which period to compare to (it defaults to last hour)

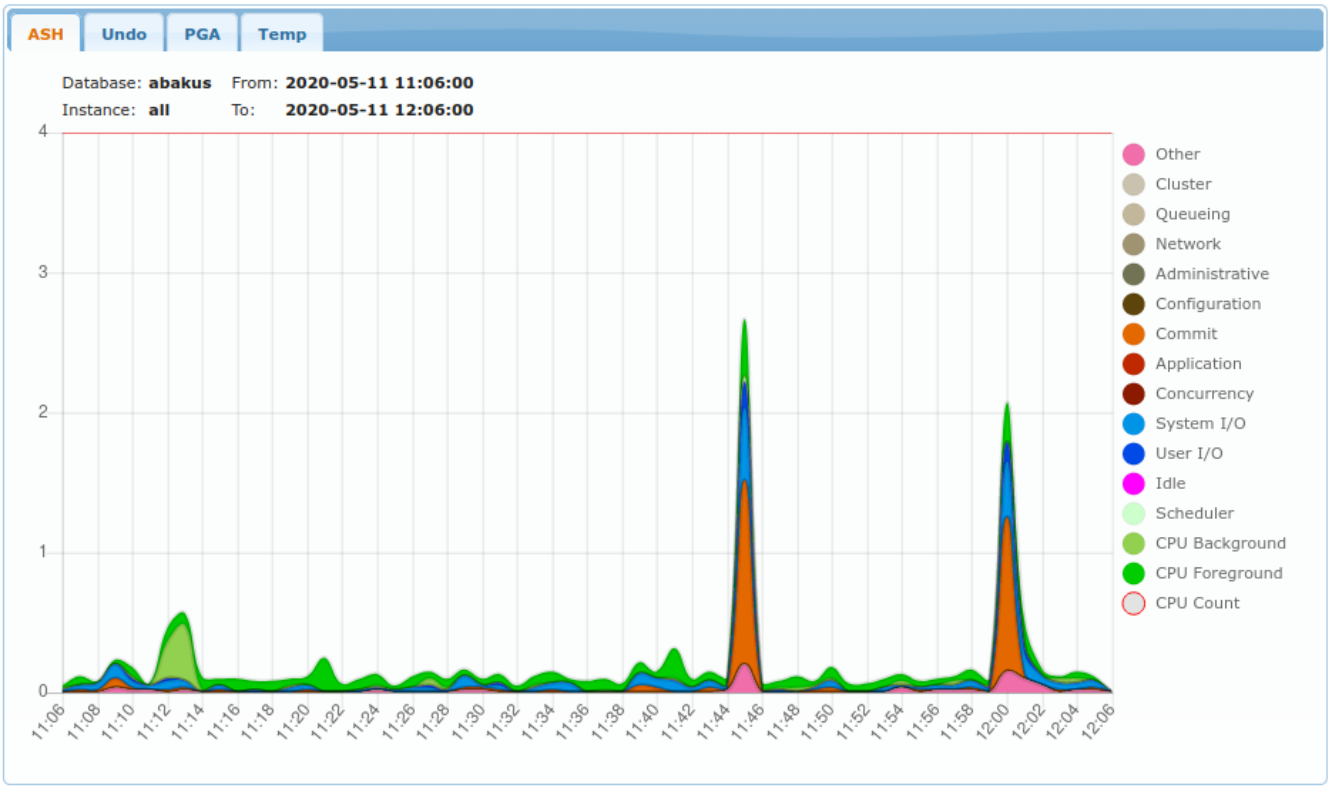
In third column of fields, you can fine-tune what/how is displayed:

- `Refresh Interval` - If enabled, the results are refreshed every `n` seconds (and timestamp forwarded).
- `Chart Group By` - Count amount of active sessions by this field (by default a `Wait Class`, but it's interesting to see other results such as `username` or `instance`; all of them refer to columns from `v$session`.
- `Chart Sample By` - Amount of average sessions are averaged by this field (by minute, hour, or day).
- `Top Table For` - What to display in `Top Table` table bellow. Among the most interesting fields (other than `Session` which is default) are
  - `SQL` - which displays most resource-consuming SQL-s
  - `PLAN` - which displays most resource-consumgin Execution Plans.
  - `SQL (per execution)` and `PLAN (per execution)` are different in that they display amount of resources consumed on average by *single* execution of SQL or it's PLAN.
- `Top Table Rows` - Amount of rows displayed in `Top Table`
- `Compare Outer Join` - Only relevant when `Compare Database` is selected; It determines whether or not to display rows that are specific to either on or the other database (`Database` and `Compare Database`).

Buttons are used to move through time more quickly - alternative would be to manually insert different dates into `From` and `To` fields.

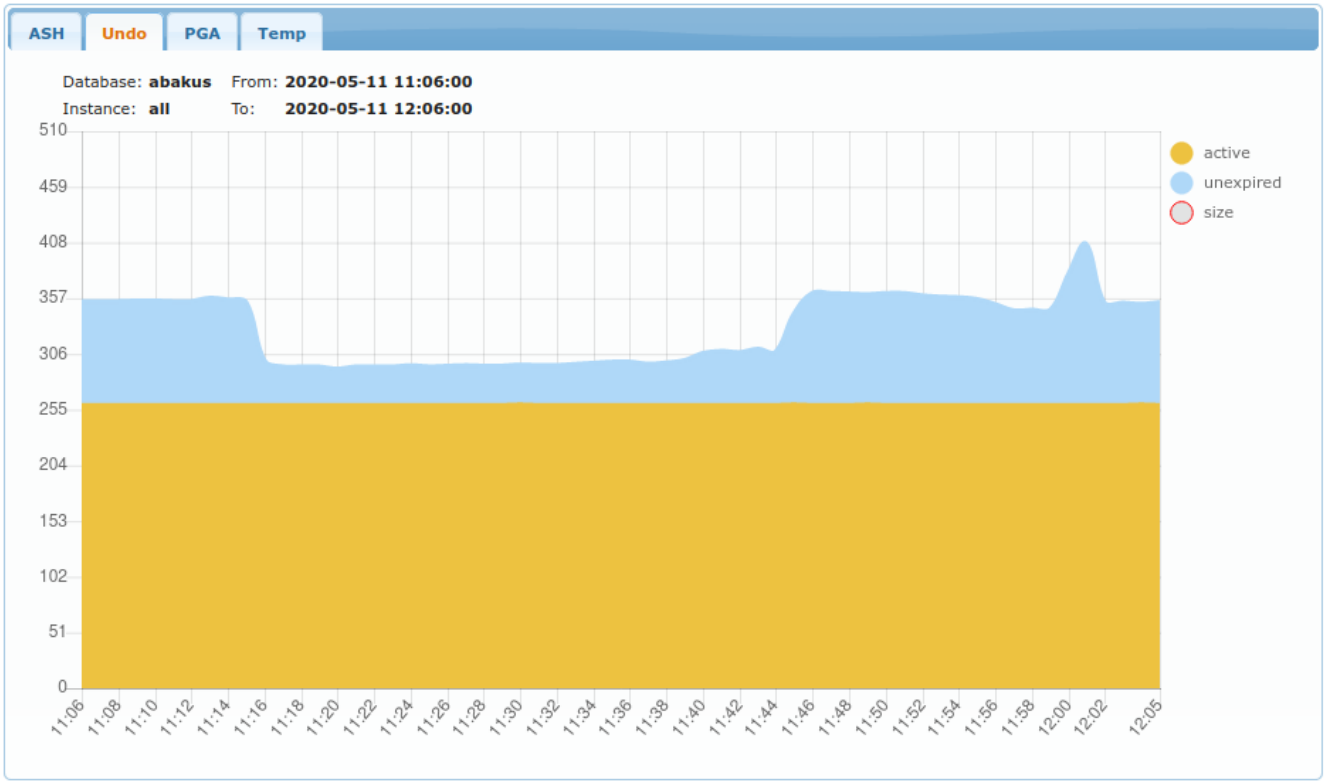
Charts

ASH



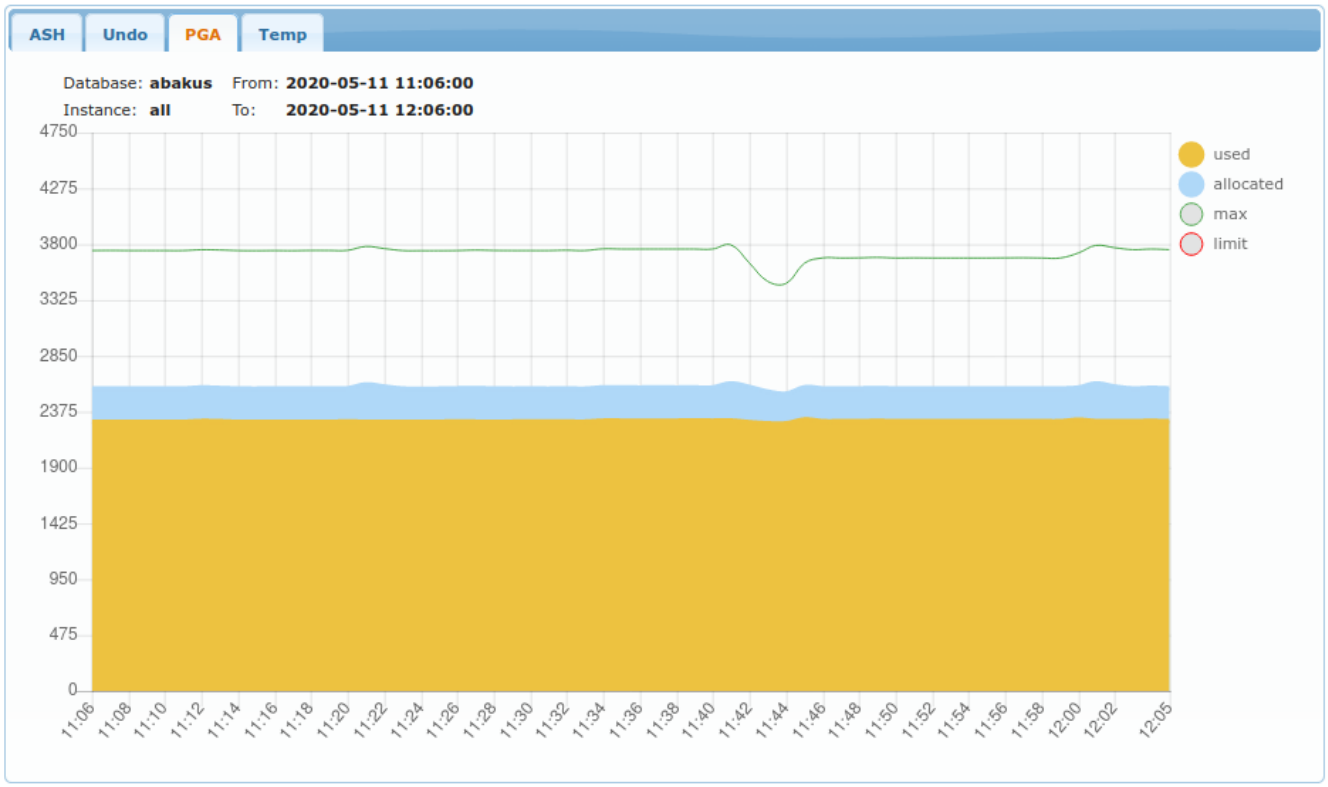
Legend displays what was selected in `Chart Group By` column. On Y axis is the number of concurrently active sessions. Data comes from samples of `v$session` view.

Undo



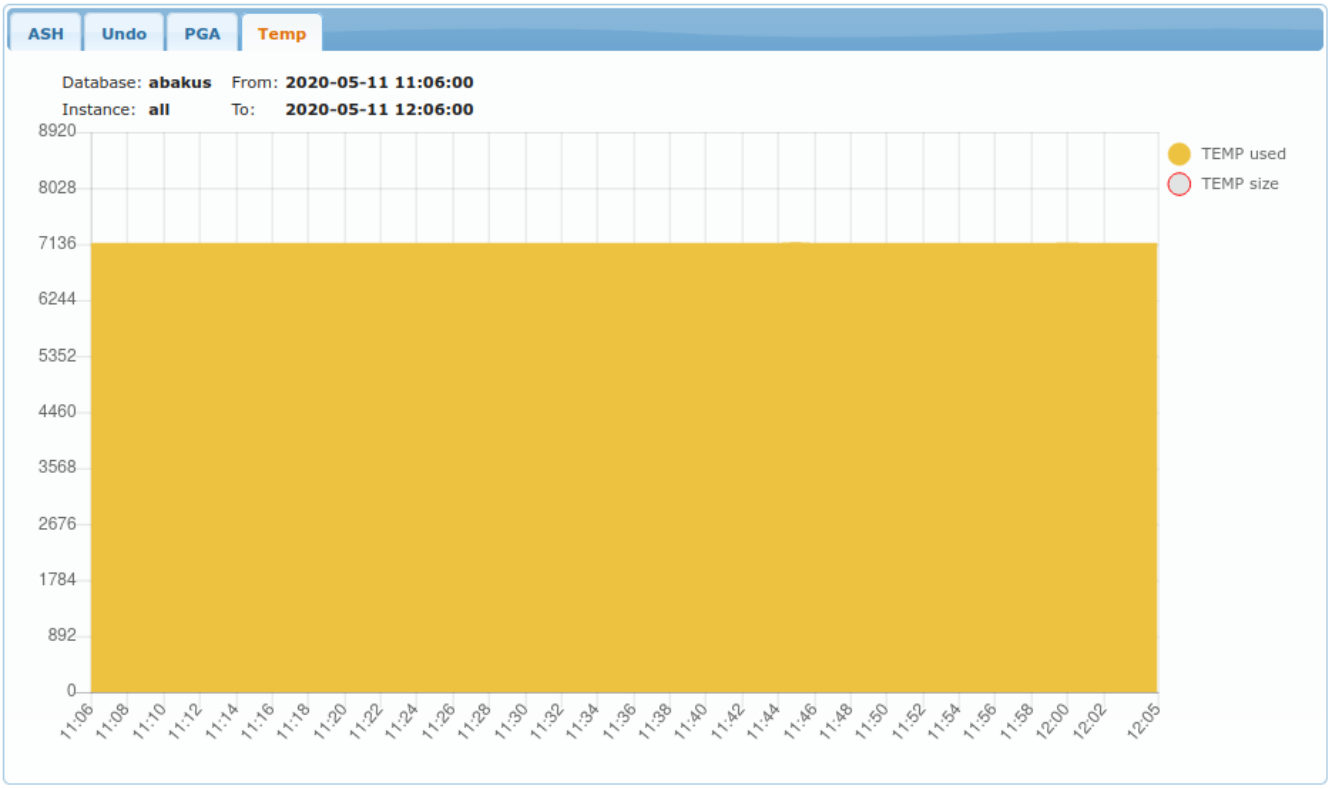
Displays amount of UNDO used in MB (megabytes). Data comes from samples of `v$transaction`.

PGA



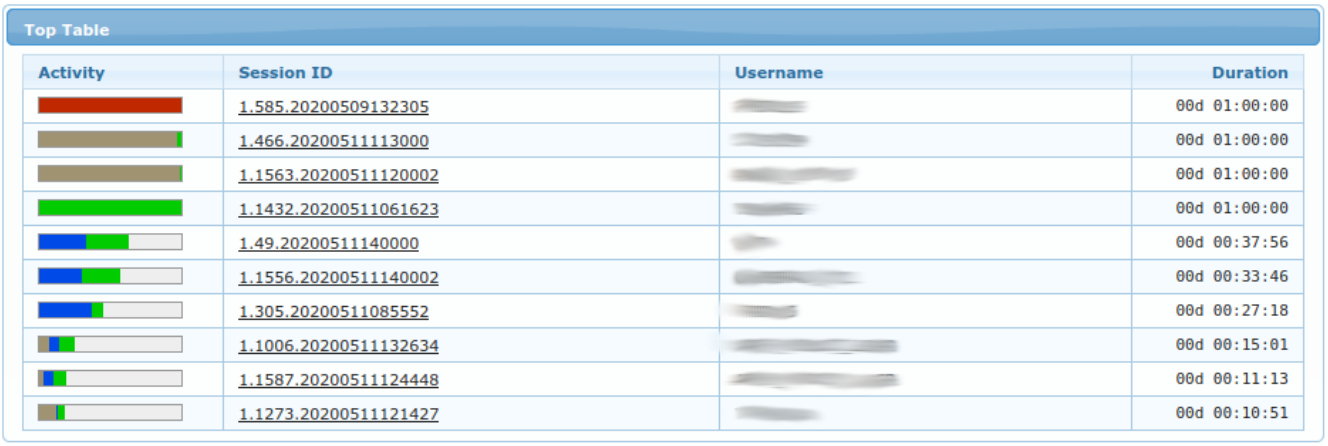
Displays amount of PGA used in MB (megabytes). Data comes from samples of `v$process`.

Temp



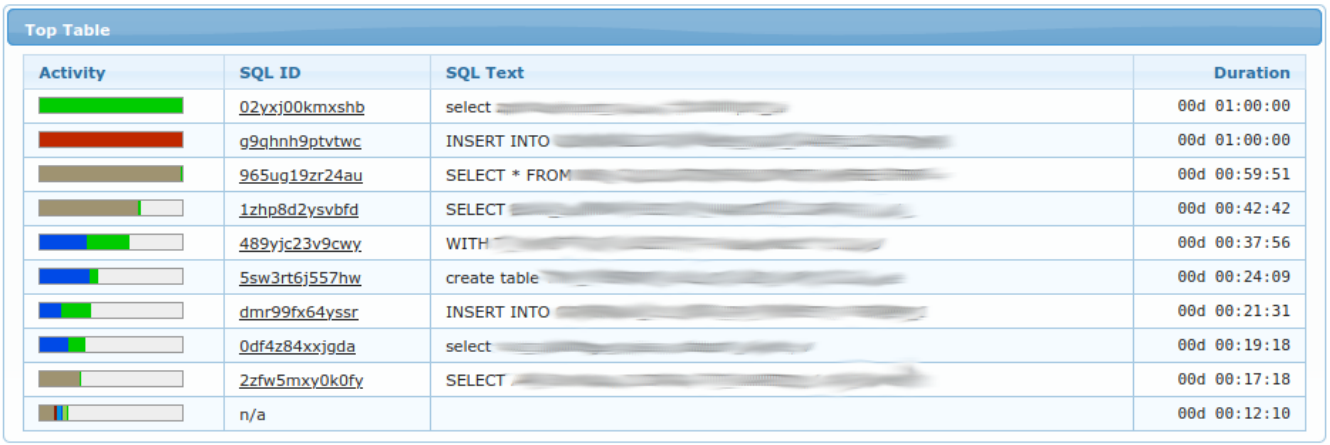
Displays amount of TEMP space usage (by temp tablespaces).

Top table



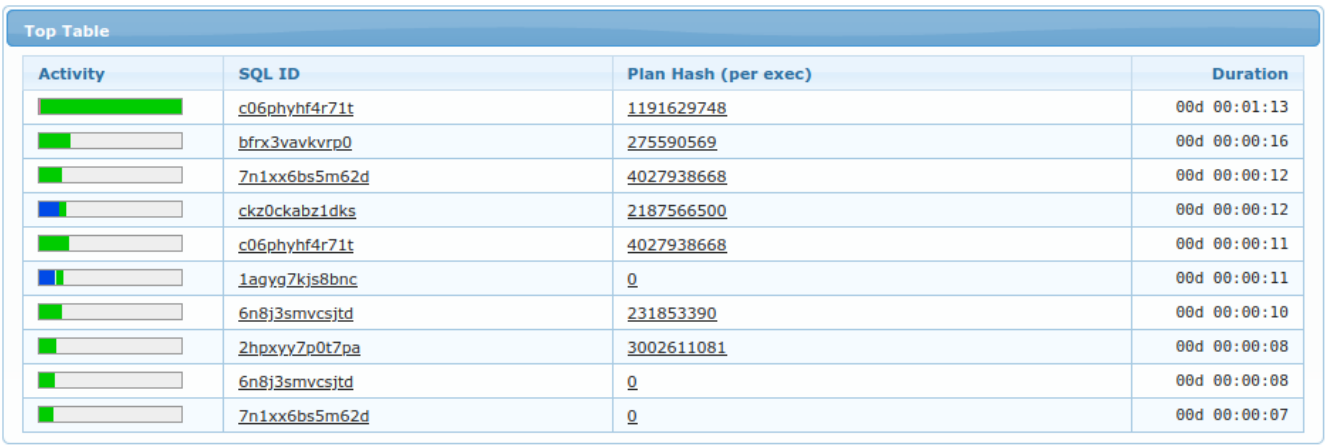
Top Table

Activity	Session ID	Username	Duration
	1.585.20200509132305		00d 01:00:00
	1.466.20200511113000		00d 01:00:00
	1.1563.20200511120002		00d 01:00:00
	1.1432.20200511061623		00d 01:00:00
	1.49.20200511140000		00d 00:37:56
	1.1556.20200511140002		00d 00:33:46
	1.305.20200511085552		00d 00:27:18
	1.1006.20200511132634		00d 00:15:01
	1.1587.20200511124448		00d 00:11:13
	1.1273.20200511121427		00d 00:10:51



Top Table

Activity	SQL ID	SQL Text	Duration
	02yxj00kmxshb	select	00d 01:00:00
	g9qhnh9ptvtwc	INSERT INTO	00d 01:00:00
	965ug19zr24au	SELECT * FROM	00d 00:59:51
	1zhp8d2ysvbfd	SELECT	00d 00:42:42
	489yjc23v9cwy	WITH	00d 00:37:56
	5sw3rt6j557hw	create table	00d 00:24:09
	dmr99fx64yssr	INSERT INTO	00d 00:21:31
	0df4z84xxjgda	select	00d 00:19:18
	2zfw5mxy0k0fy	SELECT	00d 00:17:18
	n/a		00d 00:12:10



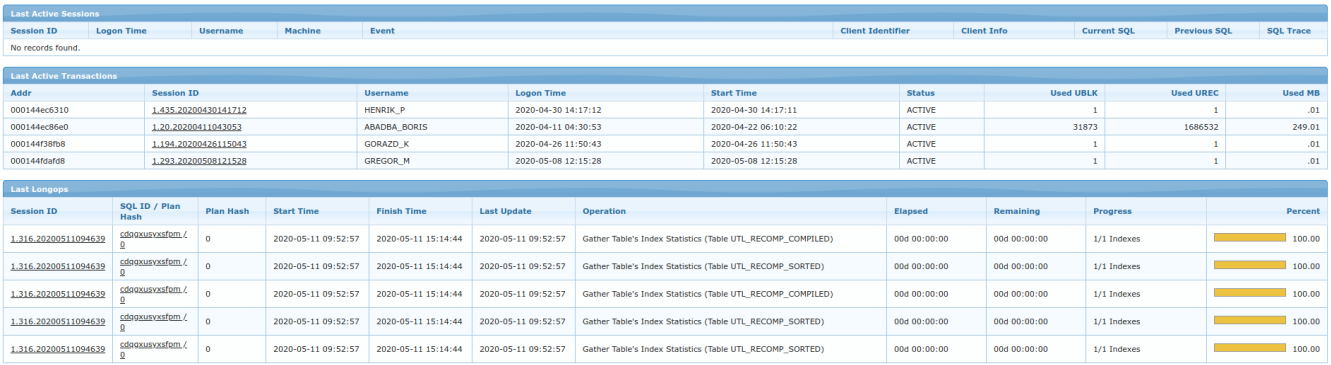
Top Table

Activity	SQL ID	Plan Hash (per exec)	Duration
	c06phyhf4r71t	1191629748	00d 00:01:13
	bfrx3vavkvrp0	275590569	00d 00:00:16
	7n1xx6bs5m62d	4027938668	00d 00:00:12
	ckz0ckabz1dks	2187566500	00d 00:00:12
	c06phyhf4r71t	4027938668	00d 00:00:11
	1agyg7kjs8bnc	0	00d 00:00:11
	6n8j3smvcsjtd	231853390	00d 00:00:10
	2hpxyy7p0t7pa	3002611081	00d 00:00:08
	6n8j3smvcsjtd	0	00d 00:00:08
	7n1xx6bs5m62d	0	00d 00:00:07

Top table displays what was selected in `Top Table For` field. Examples above are for `Sessions`, `SQL` and `PLAN`. You can click on specific sql_id or specific plan_hash_value to see details of selected SQL or Plan.

`Activity` column displays how many times this object was seen in regard to the object displayed in first row. `Duration` column displays amount of times (=seconds) such object was seen as ACTIVE in `v$session`.

Last



Last Active Sessions

Session ID	Logon Time	Username	Machine	Event	Client Identifier	Client Info	Current SQL	Previous SQL	SQL Trace
No records found.									

Last Active Transactions

Addr	Session ID	Username	Logon Time	Start Time	Status	Used UBLK	Used UREC	Used MB
000144ec6310	1.435.20200430141712	HENRIK_P	2020-04-30 14:17:12	2020-04-30 14:17:11	ACTIVE	1	1	.01
000144ec86e0	1.20.20200411043053	ABADBA_BORIS	2020-04-11 04:30:53	2020-04-22 06:10:22	ACTIVE	31873	1686532	249.01
000144f38fb8	1.194.20200426115043	GORAZD_K	2020-04-26 11:50:43	2020-04-26 11:50:43	ACTIVE	1	1	.01
000144fdafd8	1.293.20200508121528	GREGOR_M	2020-05-08 12:15:28	2020-05-08 12:15:28	ACTIVE	1	1	.01

Last Longops

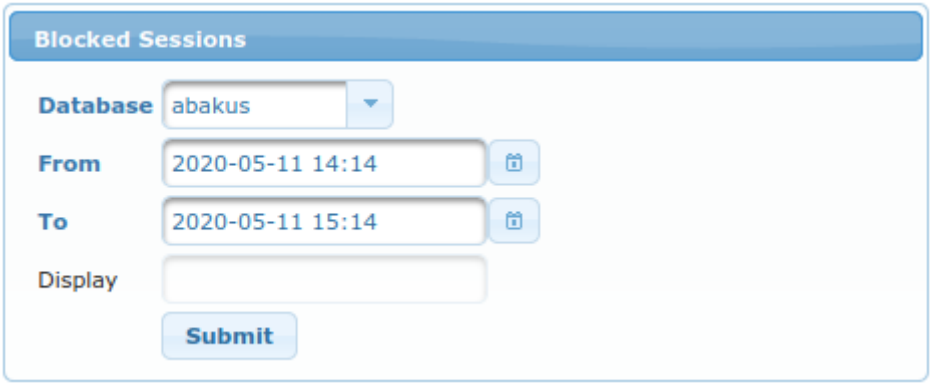
Session ID	SQL ID / Plan Hash	Plan Hash	Start Time	Finish Time	Last Update	Operation	Elapsed	Remaining	Progress	Percent
1.316.20200511094639	cdqgxusvxsfqm / 0	0	2020-05-11 09:52:57	2020-05-11 15:14:44	2020-05-11 09:52:57	Gather Table's Index Statistics (Table UTL_RECOMP_COMPILED)	00d 00:00:00	00d 00:00:00	1/1 Indexes	100.00
1.316.20200511094639	cdqgxusvxsfqm / 0	0	2020-05-11 09:52:57	2020-05-11 15:14:44	2020-05-11 09:52:57	Gather Table's Index Statistics (Table UTL_RECOMP_SORTED)	00d 00:00:00	00d 00:00:00	1/1 Indexes	100.00
1.316.20200511094639	cdqgxusvxsfqm / 0	0	2020-05-11 09:52:57	2020-05-11 15:14:44	2020-05-11 09:52:57	Gather Table's Index Statistics (Table UTL_RECOMP_COMPILED)	00d 00:00:00	00d 00:00:00	1/1 Indexes	100.00
1.316.20200511094639	cdqgxusvxsfqm / 0	0	2020-05-11 09:52:57	2020-05-11 15:14:44	2020-05-11 09:52:57	Gather Table's Index Statistics (Table UTL_RECOMP_SORTED)	00d 00:00:00	00d 00:00:00	1/1 Indexes	100.00
1.316.20200511094639	cdqgxusvxsfqm / 0	0	2020-05-11 09:52:57	2020-05-11 15:14:44	2020-05-11 09:52:57	Gather Table's Index Statistics (Table UTL_RECOMP_SORTED)	00d 00:00:00	00d 00:00:00	1/1 Indexes	100.00

This section displays last non-aggreggated samples which were collected at time less or equal to `To` field.

@todo=[insert links] For definition of each column please refer to:

- Sessions
- Transactions
- Longops

History → Blocked Sessions



Form allows to select which `Database` to analyze in which period (`From` and `To`).

Chart

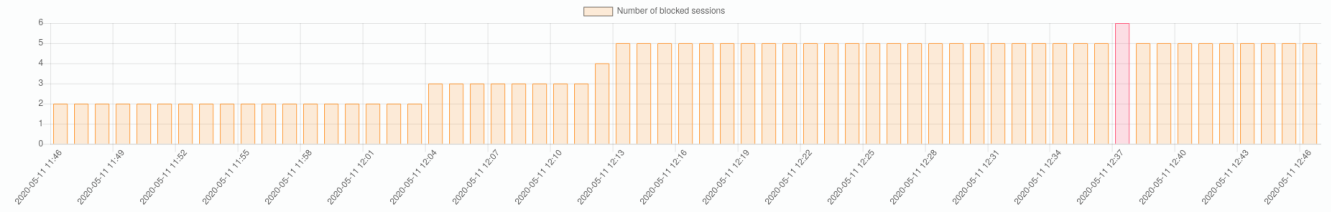
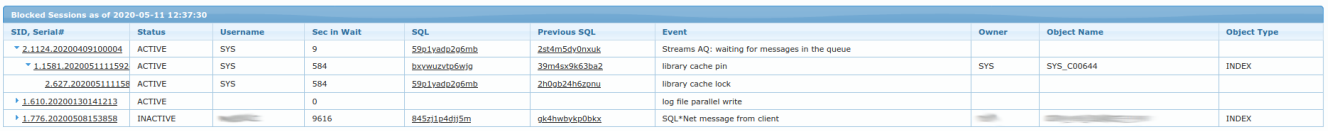


Chart displays amount of blocked sessions in selected period. One of the bars (the highest one) is of red colour - this means that the table bellow the chart refers to the same point-in-time as the red bar.

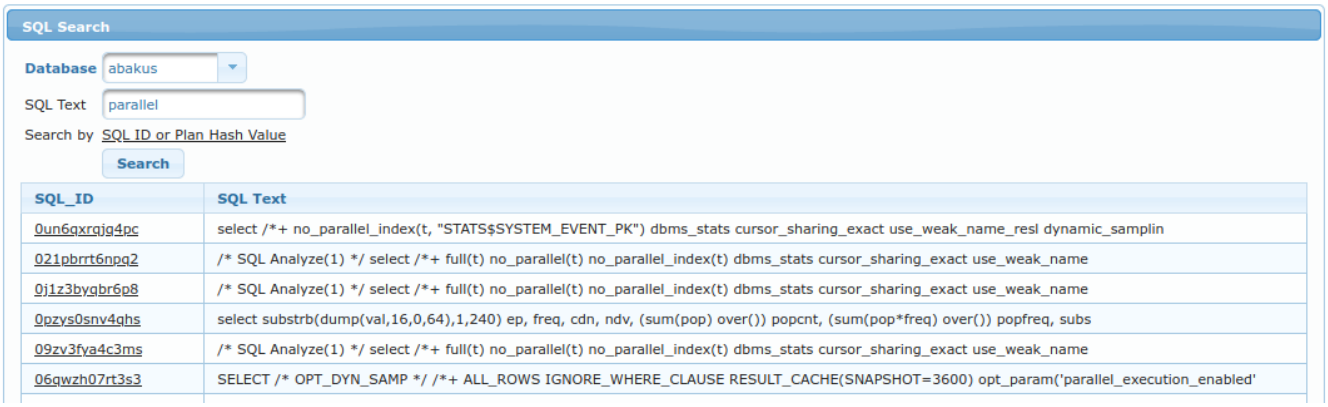
Tree Table of Blockers and Waiters



Treetable displays who was blocking who. The columns are as follows:

- `SID` - Session ID (composed of <inst_id>.<sid>.<logon_time>)
- `Status` - Status of the session as reported by `v$session.status`
- `Username` - Username for this session
- `Sec in Wait` - Seconds in wait as reported by `v$session.seconds_in_wait`
- `SQL` - SQL that is currently being executed. You can click on it to see more details.
- `Previous SQL` - SQL that was executed before the current one. You can click on it to see more details.
- `Event` - Event which is being waited on.
- `Owner` - Schema owning the object that was locked and being waited on.
- `Object Name` - Name of the object that was locked and being waited on.
- `Object Type` - Type of the object that was locked and being waited on.

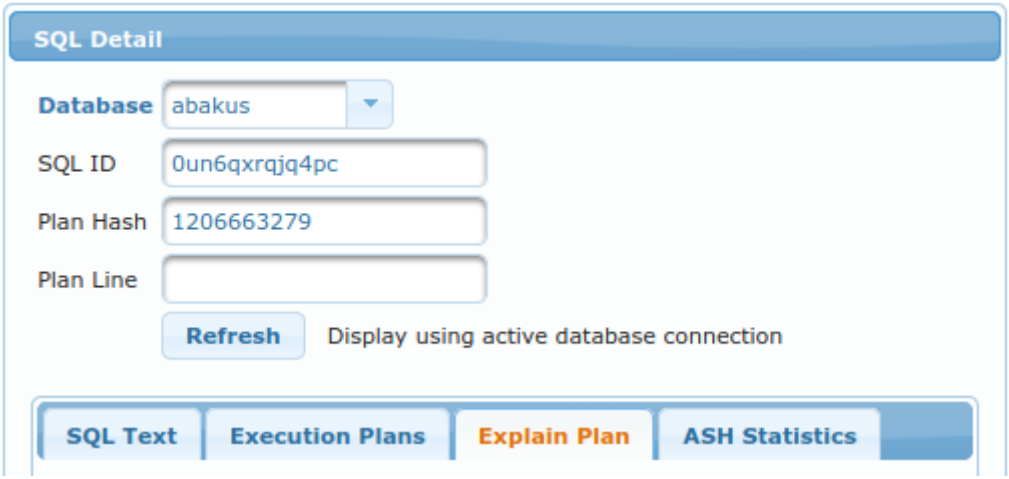
History → SQL Statements



Allows you to find any SQL Statement whose sql_id was found among samples. You can either search by it's `SQL Text` or by it's SQL ID (or plan hash value).

You can click on any sql_id found in the table to obtain more details which are explained in the next section.

SQL Detail



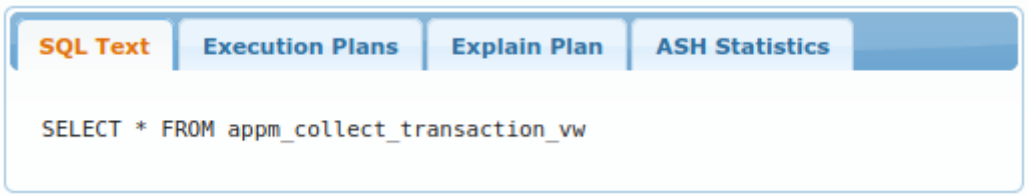
This view can be accessed by clicking on sql_id or plan_hash_value while being on any page under the `History` menu.

In order to display details for specific SQL, this form displays:

- `Database` - in which database to look for specified sql_id
- `SQL ID` - for which sql_id to look for
- `Plan Hash` - (optional) display specific plan for this sql_id
- `Plan Line` - (optional) highligh specific line in plan table output.

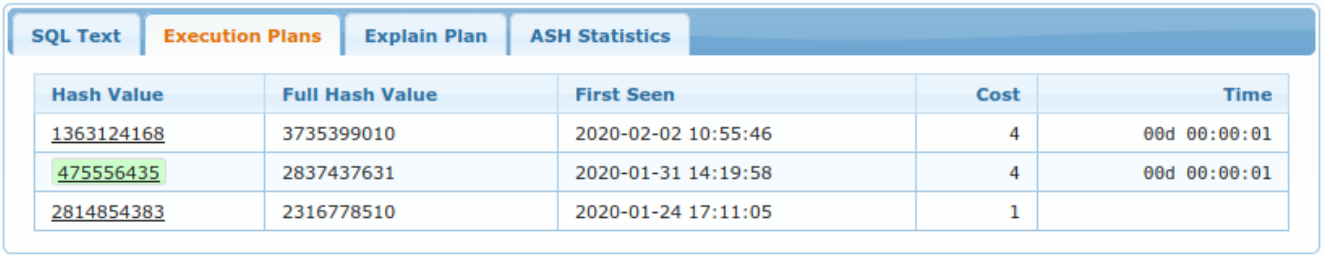
Following tabs are available:

SQL Text



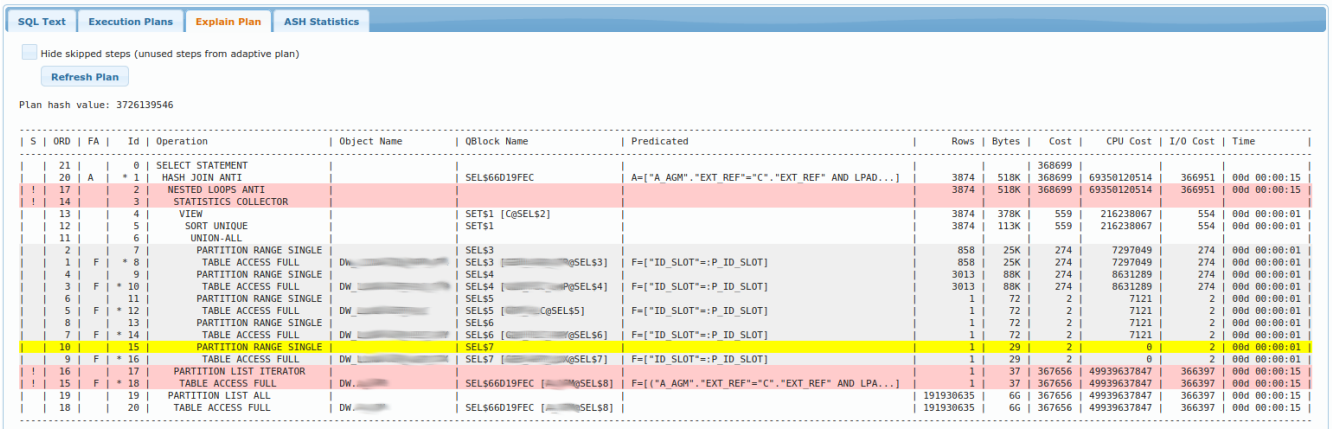
It simply displays SQL Text as seen in `v$sql.sql_fulltext`.

Execution Plans



It displays list of all plan_hash_values, which were seen while executing this specific SQL. You can select any of them - currently selected one is colored green like the one on the screenshot.

Explain Plan



S	ORD	FA	Id	Operation	Object Name	QBlock Name	Predicated	Rows	Bytes	Cost	CPU Cost	I/O Cost	Time
	21		0	SELECT STATEMENT						368699			
	20	A	* 1	HASH JOIN ANTI		SEL$66D19FEC	A=["A_AGM"."EXT_REF"="C"."EXT_REF" AND LPAD...]	3874	518K	368699	69350120514	366951	00d 00:00:15
!	17		2	NESTED LOOPS ANTI				3874	518K	368699	69350120514	366951	00d 00:00:15
!	14		3	STATISTICS COLLECTOR									
	13		4	VIEW		SET$1 [C@SEL$2]		3874	378K	559	216238067	554	00d 00:00:01
	12		5	SORT UNIQUE		SET$1		3874	113K	559	216238067	554	00d 00:00:01
	11		6	UNION-ALL									
	2		7	PARTITION RANGE SINGLE		SEL$3		858	25K	274	7297049	274	00d 00:00:01
	1	F	* 8	TABLE ACCESS FULL	DW[redacted]	SEL$3 [redacted]@SEL$3]	F=["ID_SLOT"=:P_ID_SLOT]	858	25K	274	7297049	274	00d 00:00:01
	4		9	PARTITION RANGE SINGLE		SEL$4		3013	88K	274	8631289	274	00d 00:00:01
	3	F	* 10	TABLE ACCESS FULL	DW[redacted]	SEL$4 [redacted]P@SEL$4]	F=["ID_SLOT"=:P_ID_SLOT]	3013	88K	274	8631289	274	00d 00:00:01
	6		11	PARTITION RANGE SINGLE		SEL$5		1	72	2	7121	2	00d 00:00:01
	5	F	* 12	TABLE ACCESS FULL	DW[redacted]	SEL$5 [redacted]C@SEL$5]	F=["ID_SLOT"=:P_ID_SLOT]	1	72	2	7121	2	00d 00:00:01
	8		13	PARTITION RANGE SINGLE		SEL$6		1	72	2	7121	2	00d 00:00:01
	7	F	* 14	TABLE ACCESS FULL	DW[redacted]	SEL$6 [redacted]@SEL$6]	F=["ID_SLOT"=:P_ID_SLOT]	1	72	2	7121	2	00d 00:00:01
	10		15	PARTITION RANGE SINGLE		SEL$7		1	29	2	0	2	00d 00:00:01
	9	F	* 16	TABLE ACCESS FULL	DW[redacted]	SEL$7 [redacted]@SEL$7]	F=["ID_SLOT"=:P_ID_SLOT]	1	29	2	0	2	00d 00:00:01
!	16		17	PARTITION LIST ITERATOR				1	37	367656	49939637847	366397	00d 00:00:15
!	15	F	* 18	TABLE ACCESS FULL	DW.[redacted]	SEL$66D19FEC [A[redacted]M@SEL$8]	F=[("A_AGM"."EXT_REF"="C"."EXT_REF" AND LPA...]	1	37	367656	49939637847	366397	00d 00:00:15
	19		19	PARTITION LIST ALL				191930635	6G	367656	49939637847	366397	00d 00:00:15
	18		20	TABLE ACCESS FULL	DW.[redacted]	SEL$66D19FEC [A[redacted]SEL$8]		191930635	6G	367656	49939637847	366397	00d 00:00:15

It displays table similar to what `dbms_xplan.display()` shows;

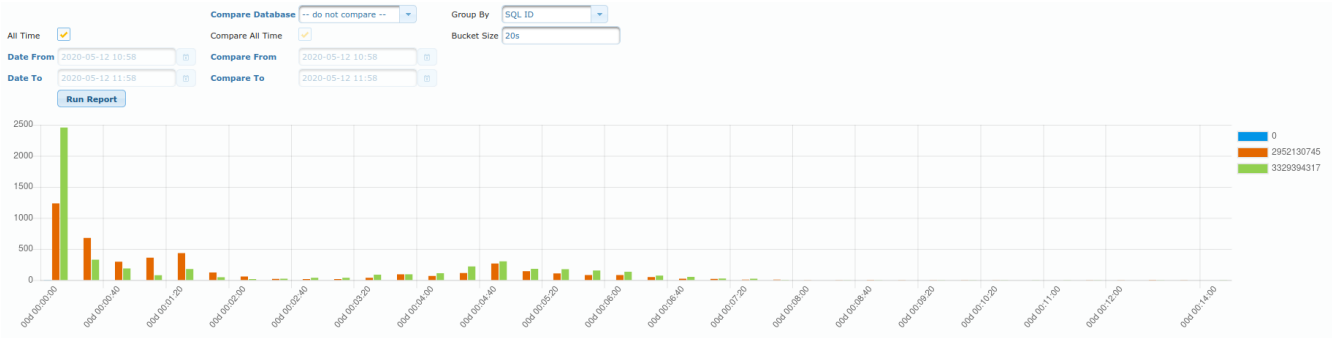
Yellow line is the line number as specified by `Plan Line` (to highlight currently executing line of query when it is known - or it can be used to highlight arbitrary plan line).

Gray lines are those that were already executed (although they can be executed again).

Red lines are steps that were not used due to *adaptive plan*. You may choose to hide those steps entirely by selecting `Hide skipped steps`.

White lines lines are yet to be executed (will be executed after the yellow line).

ASH Statistics



SQL ID	Plan Hash Value	Elapsed Time	Executions
bvwsmu96ty8g7	0	00d 00:00:00	1
bvwsmu96ty8g7	2952130745	00d 00:00:00	1235
bvwsmu96ty8g7	3329394317	00d 00:00:00	2457
bvwsmu96ty8g7	2952130745	00d 00:00:20	679
bvwsmu96ty8g7	3329394317	00d 00:00:20	327
bvwsmu96ty8g7	2952130745	00d 00:00:40	297
bvwsmu96ty8g7	3329394317	00d 00:00:40	186
bvwsmu96ty8g7	2952130745	00d 00:01:00	361
bvwsmu96ty8g7	3329394317	00d 00:01:00	78
bvwsmu96ty8g7	2952130745	00d 00:01:20	433

This view should provide insight into which plan (for specific sql) have performed better (faster) than the others. Of course, same plan can perform faster/slower at different times. And so, this view shows histogram of how fast was the plan executed.

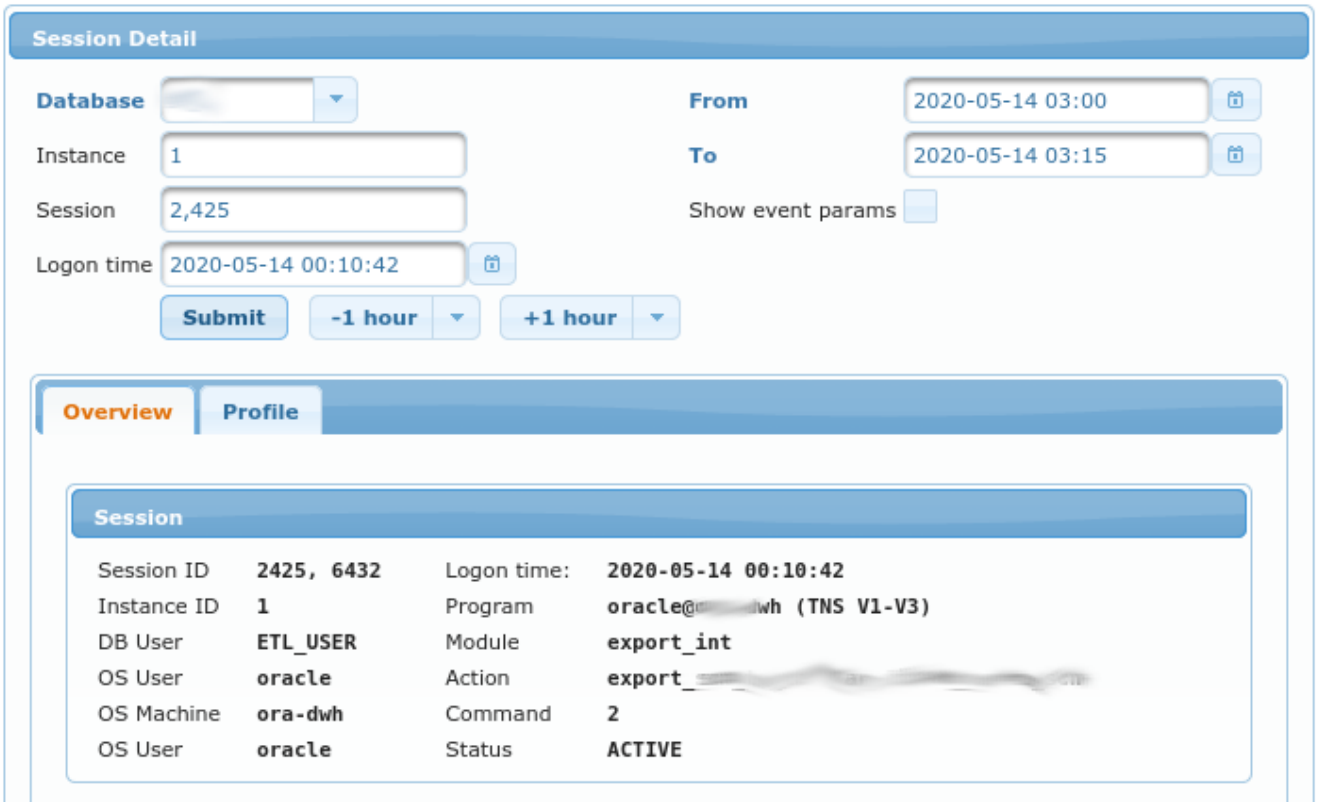
e.g. in the chart above, "green" plan was executed 2500 times in under 20 seconds.

Table displays the same data in tabular format.

You can also utility `Compare` fields in order to compare execution times across different databases or

different periods.

Session Detail



This view displays various info about specific session. Each session is uniquely identified by (fields of this form):

- `Database` - which database it refers to
- `Instance` - instance on which the session was logged on (`v$session.inst_id`)
- `Session` - session id (this is unique only at specific point in time for specific instance). It refers to `v$session.sid`.
- `Logon time` - when the session logged on. It refers to `v$session.logon_time`

Additional parameters to this view are:

- `From` - when the period for report of this view begins
- `To` - when the period for report of this view ends
- `Show event params` - whether or not to display event parameters in `Profile` tab.

Overview

Session

This view display values from `v$session` view. Fields are documented in official Oracle documentation.

Transactions

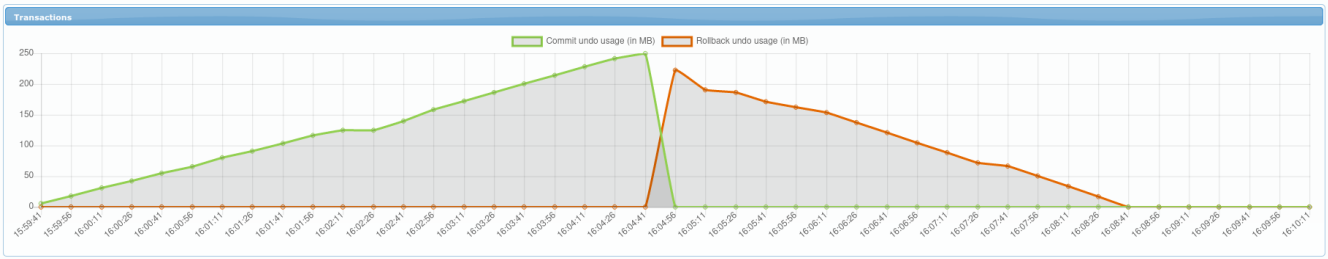


Chart displays amount of UNDO used. Green line shows amount of currently uncommited data. Orange line shows amount of UNDO used by transaction that is being rolled back.

Process

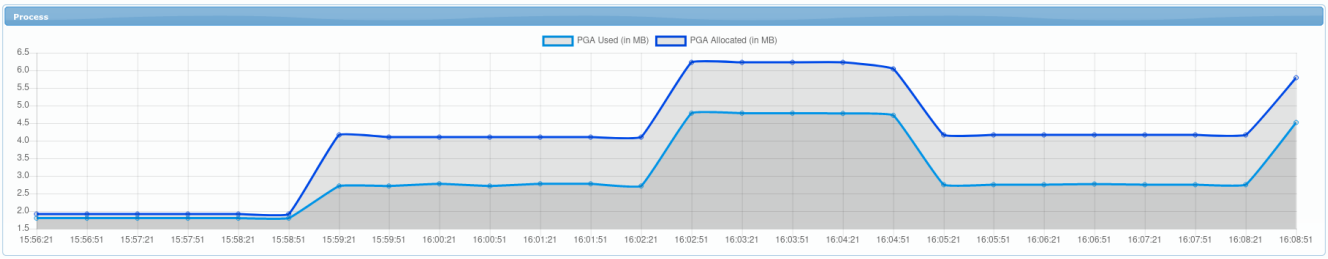
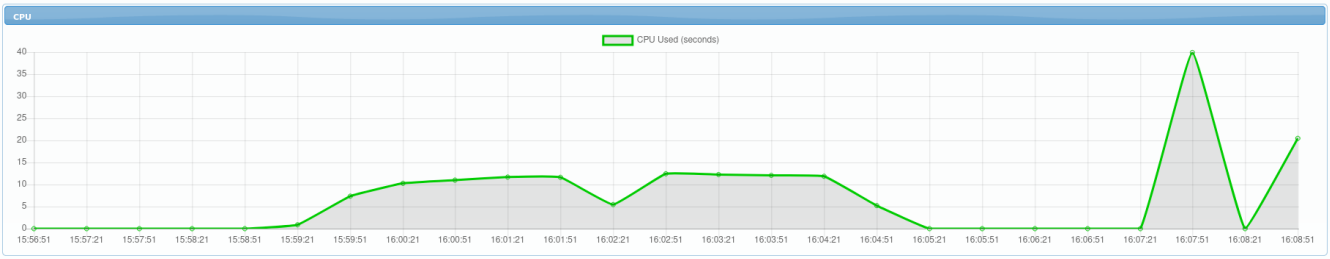


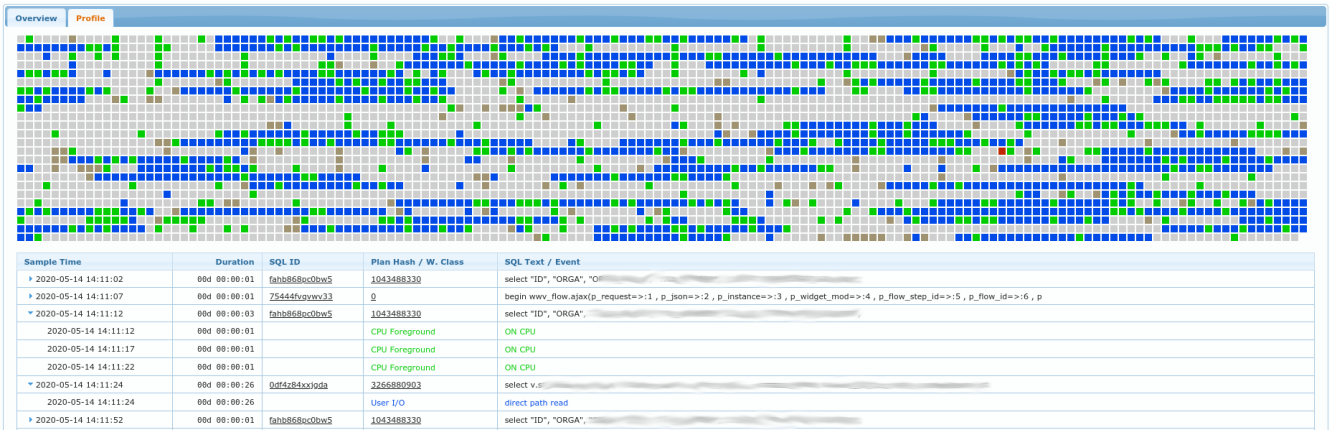
Chart displays PGA memory usage for this session's process. Darker line shows allocated usage (in MB) and ligher line shows actually used PGA (in MB).

CPU



CPU Usage for this sessions's process. This chart is available only for databases version 18c and higher, where `v$process.cpu_usage` is available.

Profile



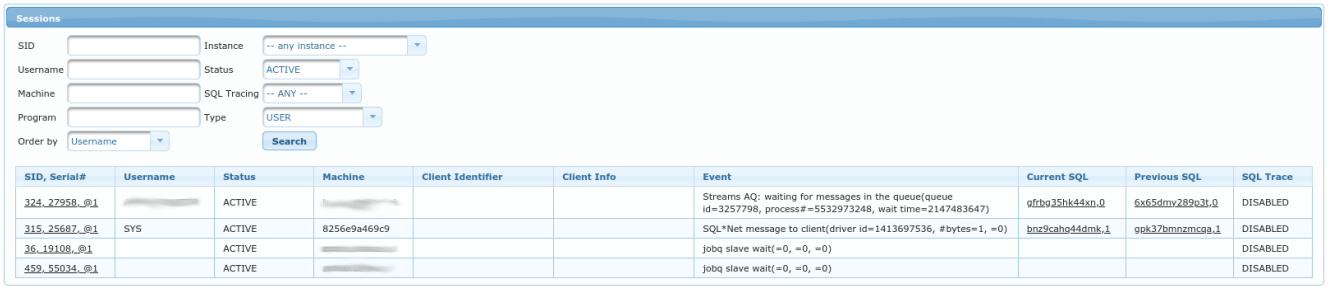
If selected period is ⇐ 2h, then chart with a lot of small squares is drawn. Each sqare represents 1 second of what session was doing. Gray squares represent idle times while all other colors represent on of the "Wait Class" colors. You can hover over any of the squares in order to obtain tooltip information what the session was waintin on on exactly that moment (which wait class and which event exactly).

Bellow the chart is table with contents somewhat similar to what SQL Trace would provide. It shows list of SQLs being executed by the session and list of events that were waited on for each of those SQLs. Color of event matches the color for a wait class.

You can get even more detailed info (added event parameters) in this table if you set checkbox `Show event params` to true.

Performance (using active connection)

Performance → Sessions



This view allows searching through `v$session` directly on the database that we're connected to.

If you simply click `Search` without changing any criteria it will show all currently `ACTIVE` sessions. You can of course search the sessions using provided form:

- `SID` - by session ID (`v$session.sid`)
- `Username` - by username (`v$session.username`)
- `Machine` - by machine (`v$session.machine`)
- `Program` - by program (`v$sessionn.program`)
- `Instance` - by instance id (`v$session.inst_id`)
- `Status` - whether to only search among currently [in]active sessions.
- `SQL Tracing` - whether to only search the sessions that have SQL Tracing enabled or disabled.
- `Type` - by type (`v$session.type`)

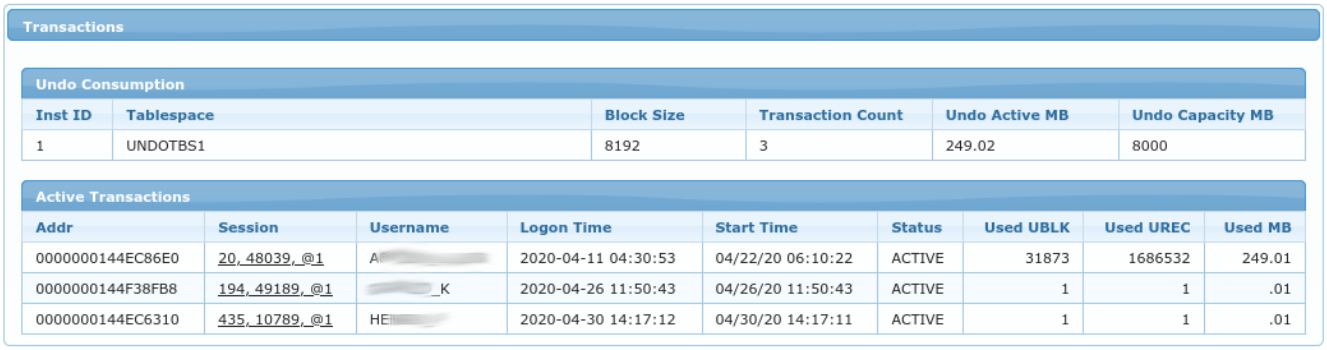
The table bellow displays results of your query. It contains columns from `v$session` - those columns are explained in official Oracle documentation at:

https://docs.oracle.com/en/database/oracle/oracle-database/19/refrn/V-SESSION.html

Note that columns `SID, Serial#`, `Current SQL` and `Previous SQL` contain links to session or SQL details.

@todo: link to `Session Detail`...

Performance → Transactions



Transactions

Undo Consumption

Inst ID	Tablespace	Block Size	Transaction Count	Undo Active MB	Undo Capacity MB
1	UNDOTBS1	8192	3	249.02	8000

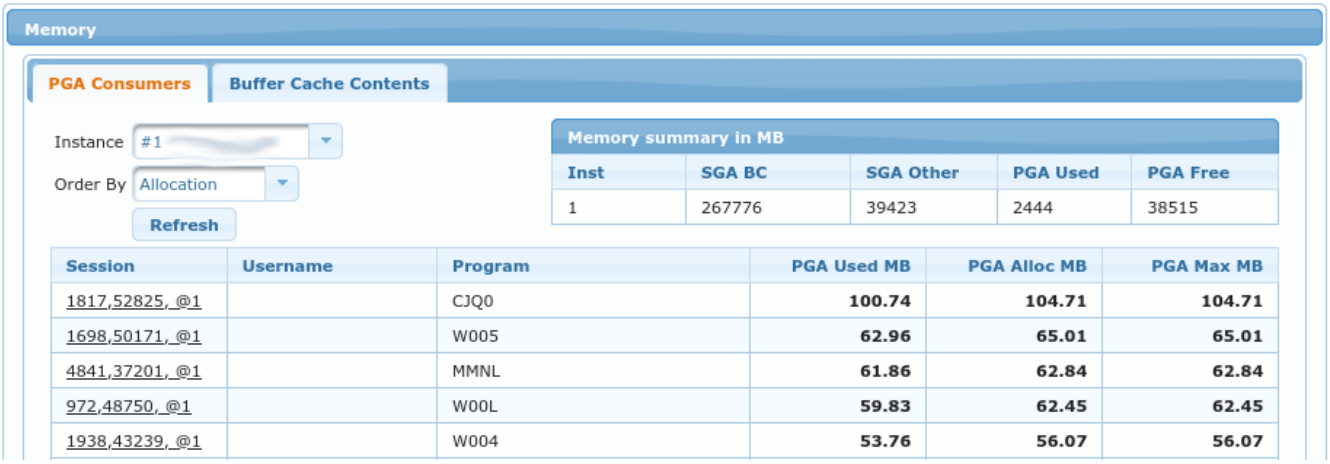
Active Transactions

Addr	Session	Username	Logon Time	Start Time	Status	Used UBLK	Used UREC	Used MB
0000000144EC86E0	20, 48039, @1	A	2020-04-11 04:30:53	04/22/20 06:10:22	ACTIVE	31873	1686532	249.01
0000000144F38FB8	194, 49189, @1	K	2020-04-26 11:50:43	04/26/20 11:50:43	ACTIVE	1	1	.01
0000000144EC6310	435, 10789, @1	HE	2020-04-30 14:17:12	04/30/20 14:17:11	ACTIVE	1	1	.01

`Undo Consumption` part display info about UNDO tablespaces and their usage. Data comes from `dba_data_files` and `gv$transaction` views.

`Active Transactions` display list of currently active transactions as available from `gv$transaction`.

Performance → Memory



Memory

PGA Consumers | Buffer Cache Contents

Instance #1

Order By Allocation

Refresh

Memory summary in MB

Inst	SGA BC	SGA Other	PGA Used	PGA Free
1	267776	39423	2444	38515

Session	Username	Program	PGA Used MB	PGA Alloc MB	PGA Max MB
1817,52825, @1		CJQ0	100.74	104.71	104.71
1698,50171, @1		W005	62.96	65.01	65.01
4841,37201, @1		MMNL	61.86	62.84	62.84
972,48750, @1		W00L	59.83	62.45	62.45
1938,43239, @1		W004	53.76	56.07	56.07

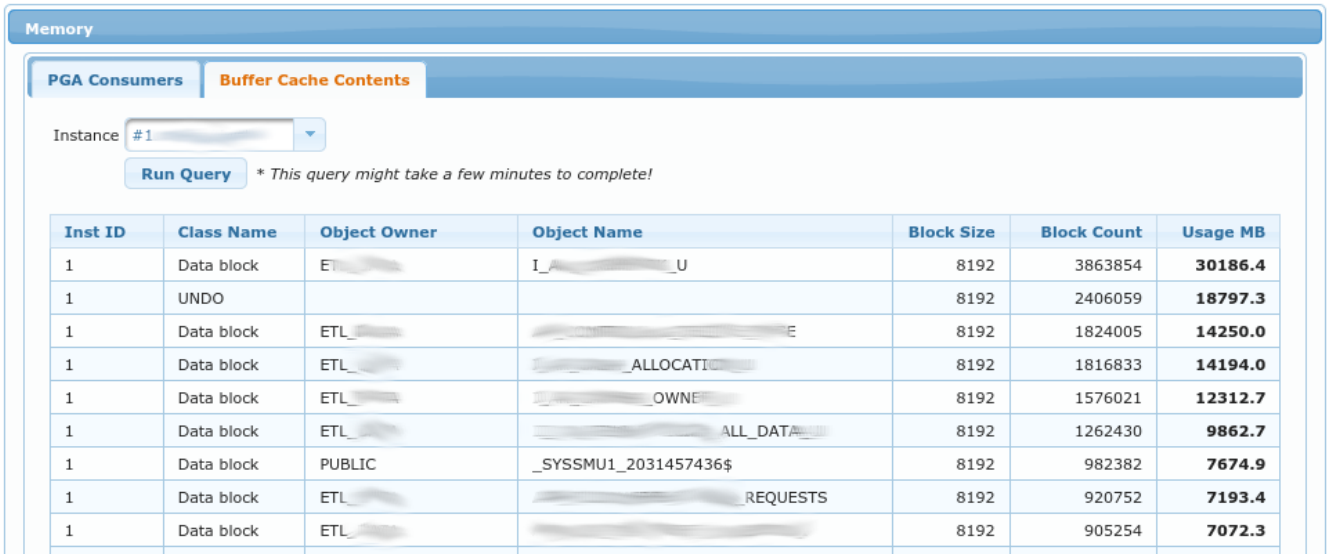
`Memory Summary` displays how much memory is dedicated to PGA and SGA (in megabytes):

- `Inst` - instance id for which data is displayed
- `SGA BC` - amount of memory dedicated to *Buffer Cache*
- `SGA Other` - amount of memory dedicated to all other parts of SGA
- `PGA Used` - amount of PGA memory used
- `PGA Free` - amount of PGA memory free

Table bellow displays amount of PGA memory occupied by session

- `Session` - session id for which the data is displayed. It contains a link to session details.
- `Username` - database username for this session
- `Program` - program name for session
- `PGA Used MB` - amount pf PGA currently *used*
- `PGA Alloc MB` - amount of PGA memory currently *allocated*
- `PGA Max MB` - amount of PGA memory allocated at the peak point of the session.

Buffer Cache Contents

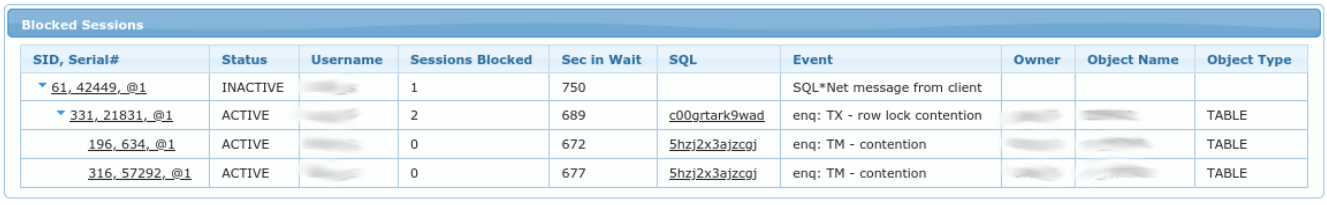


Inst ID	Class Name	Object Owner	Object Name	Block Size	Block Count	Usage MB
1	Data block	E	I_ _U	8192	3863854	**30186.4**
1	UNDO			8192	2406059	**18797.3**
1	Data block	ETL_	E	8192	1824005	**14250.0**
1	Data block	ETL_	_ALLOCATI	8192	1816833	**14194.0**
1	Data block	ETL_	_OWNE	8192	1576021	**12312.7**
1	Data block	ETL_	ALL_DATA	8192	1262430	**9862.7**
1	Data block	PUBLIC	_SYSSMU1_2031457436$	8192	982382	**7674.9**
1	Data block	ETL_	REQUESTS	8192	920752	**7193.4**
1	Data block	ETL_		8192	905254	**7072.3**

This view displays contents of *Buffer Cache*. It is based on `gv$bh` view.

- `Inst ID` - instance ID for which data is displayed
- `Class Name` - Name of class (type/what is cached)
- `Object Owner` - Owner of the table/object which is cached
- `Object Name` - Name of the table/object which is cached
- `Block Size` - Block size of the tablespace for the segment of this table/object
- `Block Count` - Amount of blocks cached
- `Usage MB` - amount of MB cached for this object

Performance → Blocked Sessions



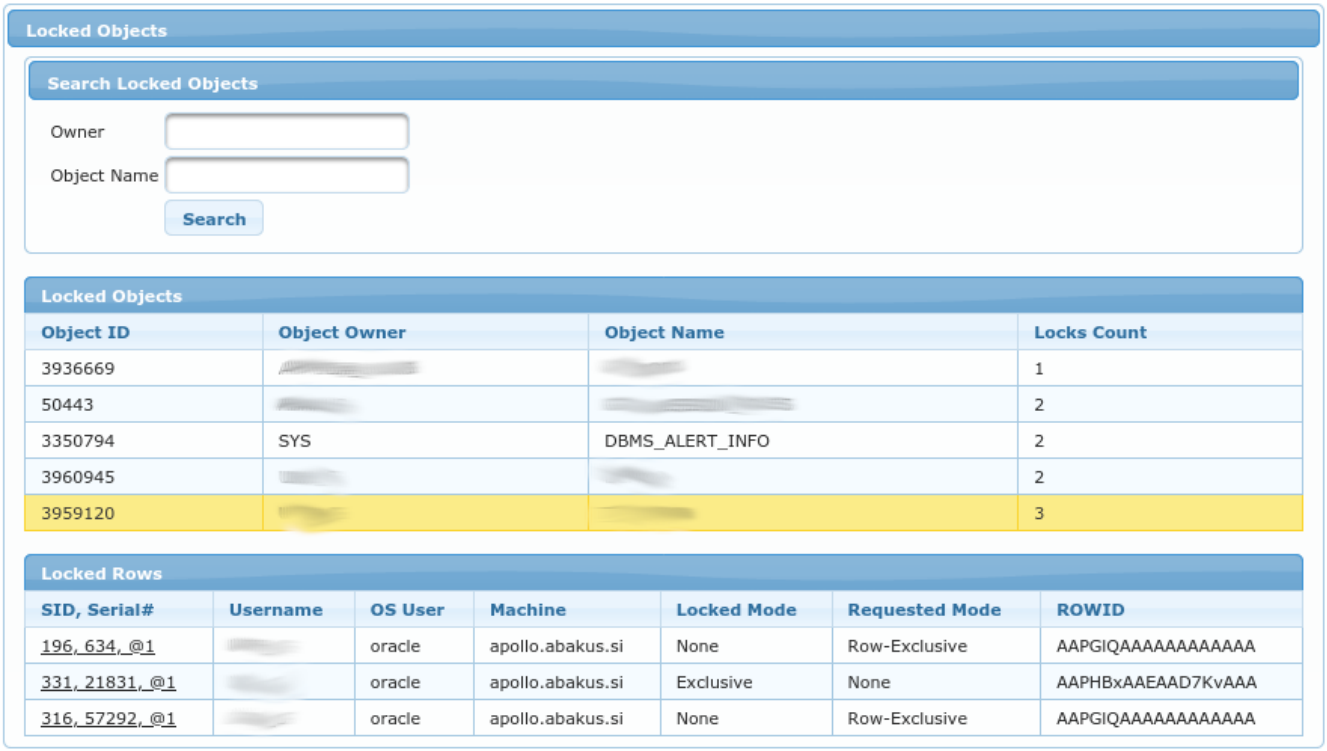
SID, Serial#	Status	Username	Sessions Blocked	Sec in Wait	SQL	Event	Owner	Object Name	Object Type
61, 42449, @1	INACTIVE		1	750		SQL*Net message from client			
331, 21831, @1	ACTIVE		2	689	c00grtark9wad	enq: TX - row lock contention			TABLE
196, 634, @1	ACTIVE		0	672	5hzj2x3ajzcgj	enq: TM - contention			TABLE
316, 57292, @1	ACTIVE		0	677	5hzj2x3ajzcgj	enq: TM - contention			TABLE

This view displays a tree table of all blocked sessions. Blockers are parent nodes and waiters are child nodes.

- `SID, Serial#` - session id of blocker/waiter; it also contains a link which opens session details for this session.
- `Status` - column from `v$session.status`; it shows if the session is active or not
- `Username` - username for this database session
- `Sessions Blocked` - amount of sessions being blocked by this session
- `Sec in Wait` - column from `v$session.seconds_in_wait` - number of seconds waiting for the current event.

- `SQL` - which SQL is currently being executed by this session. It also contains a link which opens SQL Details for this session.
- `Event` - event on which the session is currently wainting.
- `Owner` - owner of object for which the session is trying to obtain the lock
- `Object Name` - name of the object for which the session is trying to obtain the lock
- `Object Type` - type of the object for which the session is trying to obtain the lock

Performance → Locked Objects



Search Locked Objects

Form allows searching for specific objects.

- `Owner` - owner of the object that is being searched for.
- `Object Name` - name of the object that is being searched for.

Both fields are using `LIKE` operator, so you can use `%` as a wildcard.

Locked Objects

Table displays list of locked objects (according to filter in the form above).

- `Object ID` - ID as specified in `dba_objects.object_id`
- `Object Owner` - owner of the locked object
- `Object Name` - name of the locked object
- `Locks Count` - amount of locks that are held on this object

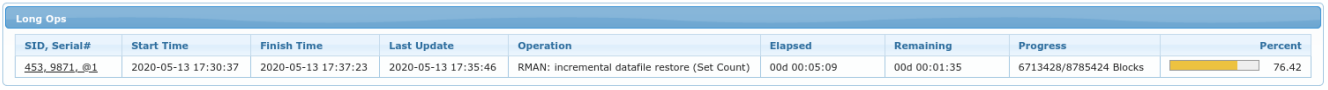
You can select any of the rows to refresh the child table `Locked Rows`.

Locked Rows

This is a child table of `Locked Objects` above. It displays list of rows locked in selected table (if object is a table).

- `SID, Serial#` - session id of the session that hold the lock
- `Username` - lock holder username
- `OS User` - OS username of lock holder (refers to `v$session.os_user` column)
- `Machine` - Machine of lock holder (refers to `v$session.machine` column)
- `Locked Mode` - In what mode the lock is acquired
- `Requested Mode` - In what mode the lock is requested, but not yet acquired
- `ROWID` - of the row being locked (when available and when appropriate for the lock type)

Performance → Long Ops



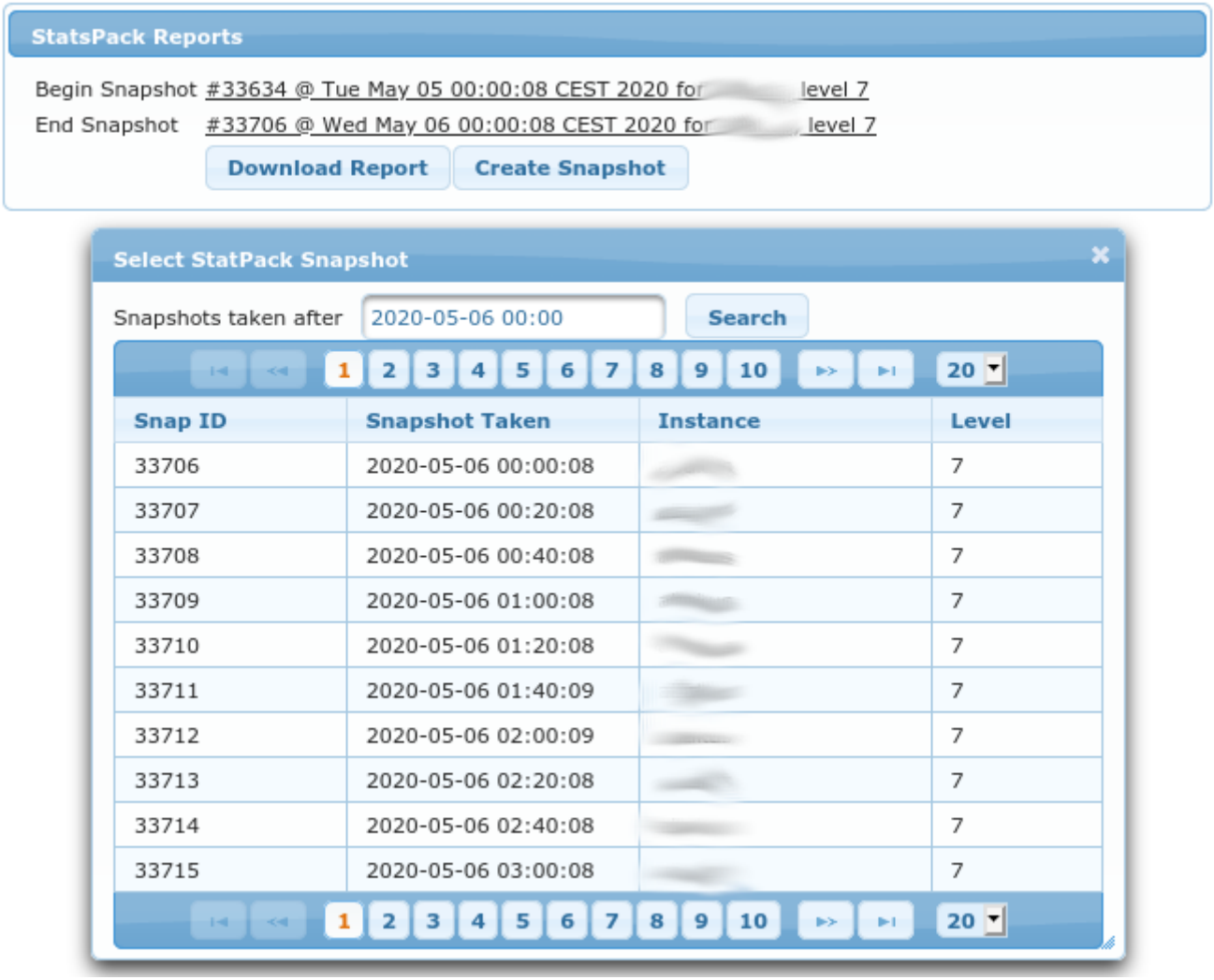
Long Ops

SID, Serial#	Start Time	Finish Time	Last Update	Operation	Elapsed	Remaining	Progress	Percent
453, 9871, @1	2020-05-13 17:30:37	2020-05-13 17:37:23	2020-05-13 17:35:46	RMAN: incremental datafile restore (Set Count)	00d 00:05:09	00d 00:01:35	6713428/8785424 Blocks	76.42

This view is based on `v$session_longops` and it shows progress of long running operations.

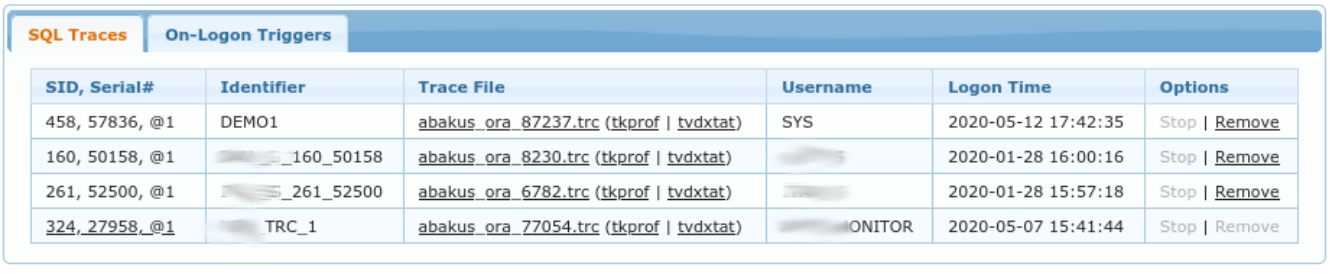
- `SID, Serial#` - Identifier of the session processing the long-running operation. If multiple sessions are cooperating in the long-running operation, then SID corresponds to the main or master session.
- `Start Time` - Starting time of the operation
- `Finish Time` - Estimated finish time
- `Last Update` - Time when statistics were last updated for the operation
- `Operation` - Brief description of the operation
- `Elapsed` - Amount of elapsed time from the start of the operation
- `Remaining` - Amount of time estimated to the end of the operation
- `Progress` - `SOFAR`/`TOTALWORK` `UNITS` from v$session_longops.
- `Percent` - amount of work completed in percents.

Performance → Statspack



This view allows you to manually create new statspack snapshot (by clicking `Create Snapshot` button) and to generate *Statspack Report*. Report is generated by selecting two snapshots to compare, you can select them by clicking on labels besides `Begin Snapshot` and `End Snapshot.

Performance → SQL Trace



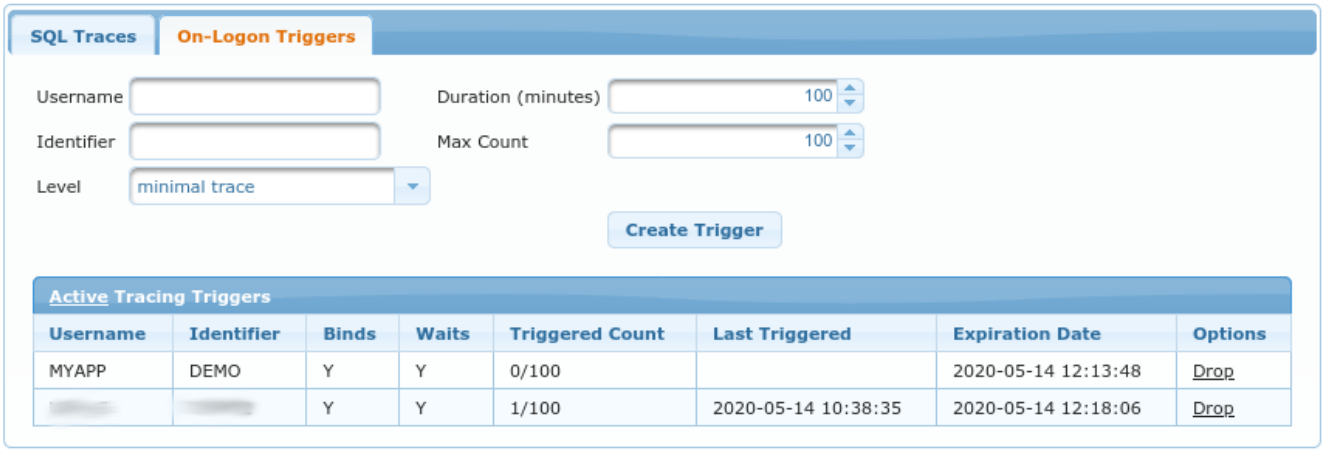
IMPORTANT	You can start tracing any session in `Session Detail` view. (@todo: link).

`SQL Traces` lists all of the trace files created by APPM. It allows you to `Stop` currently active trace and to `Remove` obsolete trace files. Note that you can only `Remove` those trace files for which session has already ended (trace file of an active session cannot be deleted through APPM).

You can download raw trace files by clicking on their name such as `dbname_ora_<pid>.trc` or you can download aggregated reports based on those trace files. Two such options exists:

- `tkprof`, which is official, Oracle supplied tool
- `tvd$xtat`, this is a free tool by Christian Antognini (Trivadis)

On-Logon Triggers

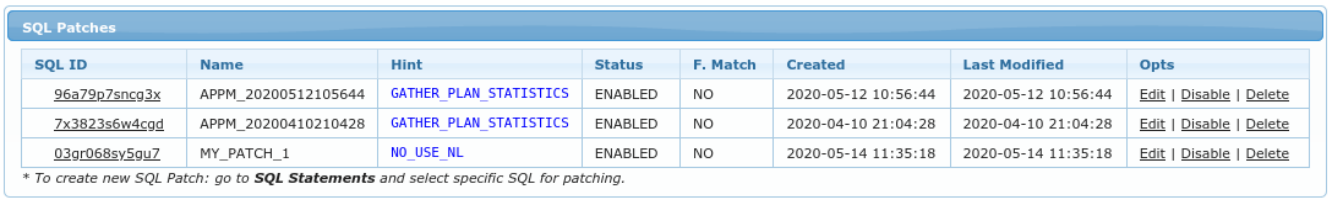


For tracing short-lived sessions or things that happen right after the logon, there is an option to create `ON LOGON TRIGGER` which automatically starts the SQL trace. Following options are available when creating such trigger:

- `Username` - Oracle schema/username which will be traced after logon
- `Duration (minutes)` - Drop logon trigger after this many minutes have passed (thus making sure we don't create such trigger and forget about it)
- `Identifier` - tracefile identifier, this is prefix for the trace filename. It is also alias for those traces in this GUI (see previous chapter)
- `Max Count` - number of logons to trace, after this many logons the logon trigger is dropped (thus making sure we don't flood the system with too many trace files)
- `Level` - how much detail do we need; it determines whether or not do we need bind variables and wait events present in the trace files.

Table under `Active Tracing Triggers` displays list of currently active triggers and allows you to immediately drop the specific trigger if necessary.

Performance → SQL Patch



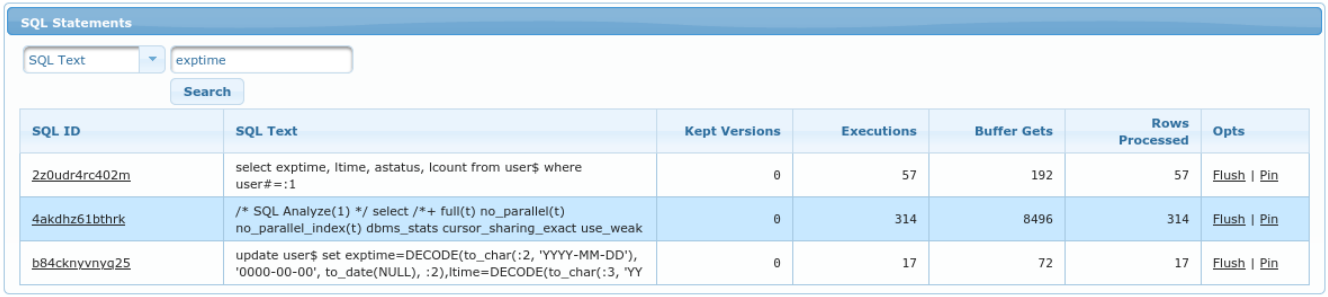
IMPORTANT	You can create new SQL Patch in SQL Details view (@todo: link).

This page lists all SQL Patches.

Columns of the table are as follows:

- `SQL ID` - SQL ID of the statement for which the patch was created. If patch was created manually (without APPM), then this column displays SQL signature number. Editing of such manually created patch is not possible.
- `Name` - User defined (or auto generated) name of SQL Patch. This is unique among all SQL patches.
- `Hint` - Hints that are applied using this patch. Those written in blue color are valid hint names and those in red color have syntax error.
- `Status` - Tells if this patch is enabled or disabled. Only one SQL Patch may be enabled at a time. (Note to advanced user: only one patch per group may be active, but APPM only operates in DEFAULT group).
- `F. Match` - If force matching is used.
- `Created` - Date when the patch was created.
- `Last Modified` - Date when the patch was last modified.
- `Opts` - You may enable/disable or delete or edit any patch listed.

Performance → SQL Statements



This page allows searching through all SQL Statements which are in SGA. It allows searching by:

- `SQL Text` - any part of sql text. Note that whatever you enter has appended and prepended `%` (for `LIKE` operator)
- `SQL ID` - by `sql_id`
- `Hash Value` - by `plan_hash_value`
- `Pinned Only` - display only the SQLs that are currently pinned in memory

Results table has following columns:

- `SQL ID` - sql_id; it contains a link to `SQL Detail` (@todo: link).
- `SQL Text` - first part of sql text (if it is too long). You can put cursor on the text and wait second - tooltext shold pop up and display complete SQL.
- `Kept Versions` - Amount of cursors kept in memory
- `Executions` - Number of executions since last hard parse
- `Buffer Gets` - Amount of buffer gets since last hard parse
- `Rows Processed` - Amount of rows processed since last hard parse

- `Opts` - There are two options: `flush` (which will cause this sql_id to be flushed from SGA and thus require hard parse on next execution) and `pin` (to make sure that sql_id won't age out of memory).

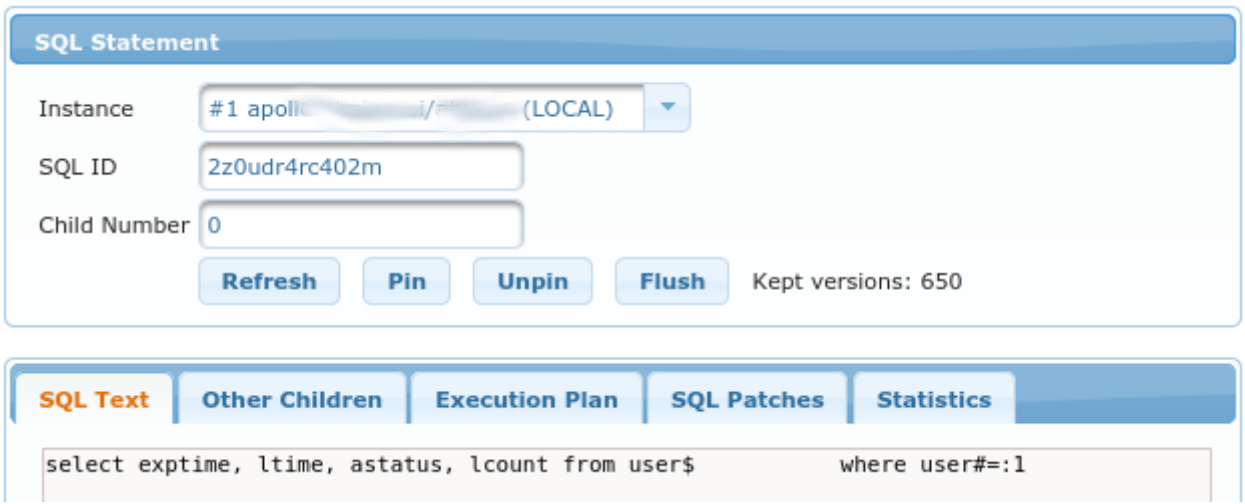
Performance → Alert Log

The page displays contents of alert log. It also allows to download the complete alert log.

- `Grep` - allows to filter alert log for specific lines, e.g. `ORA-00600%`
- `Tail` - amount of last lines to display bellow

Note that this way application developers can access the alert log without ssh/rdp access to the database server.

SQL Detail



SQL Details are displayed for SQL identified by:

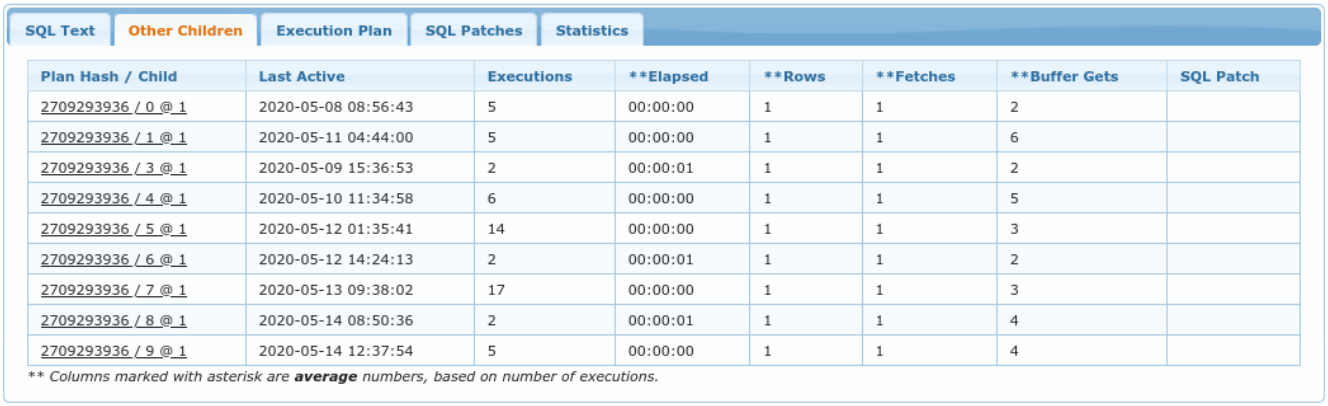
- `Instance` - this is `inst_id`

- `SQL ID` - this identifies the SQL
- `Child Number` - specific child number, can be `0` if child is not known

SQL Text

displays complete SQL Text for selected `sql_id`.

Other Children



Plan Hash / Child	Last Active	Executions	**Elapsed	**Rows	**Fetches	**Buffer Gets	SQL Patch
2709293936 / 0 @ 1	2020-05-08 08:56:43	5	00:00:00	1	1	2	
2709293936 / 1 @ 1	2020-05-11 04:44:00	5	00:00:00	1	1	6	
2709293936 / 3 @ 1	2020-05-09 15:36:53	2	00:00:01	1	1	2	
2709293936 / 4 @ 1	2020-05-10 11:34:58	6	00:00:00	1	1	5	
2709293936 / 5 @ 1	2020-05-12 01:35:41	14	00:00:00	1	1	3	
2709293936 / 6 @ 1	2020-05-12 14:24:13	2	00:00:01	1	1	2	
2709293936 / 7 @ 1	2020-05-13 09:38:02	17	00:00:00	1	1	3	
2709293936 / 8 @ 1	2020-05-14 08:50:36	2	00:00:01	1	1	4	
2709293936 / 9 @ 1	2020-05-14 12:37:54	5	00:00:00	1	1	4	

*** Columns marked with asterisk are **average** numbers, based on number of executions.*

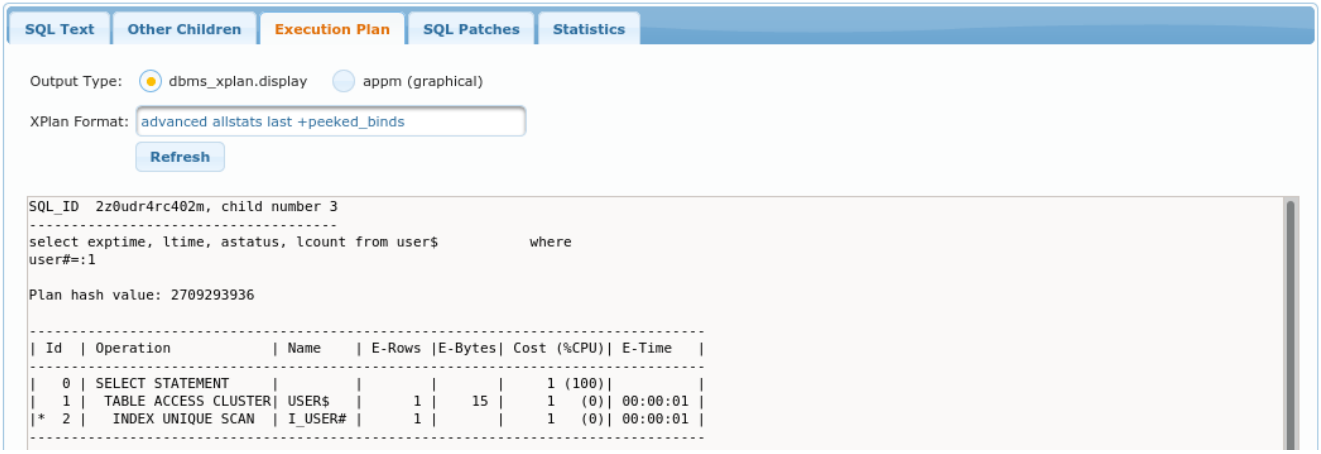
This tab displays list of all known SQL Plans. This is basically list of plans from `v$sql`.

If child number was known when the dialog was open, then this child number is highlighted with green color as seen on the screenshot.

Columns are as following:

- `Plan Hash / Child` - Plan hash is numeric representation of the current SQL plan for this cursor. Comparing one PLAN_HASH_VALUE to another easily identifies whether or not two plans are the same (rather than comparing the two plans line by line). Child instance is number of child cursor.
- `Last Active` - Time at which the query plan was last active.
- `Executions` - Number of executions that took place on this object since it was brought into the library cache.
- `**Elapsed` - Elapsed time used by this cursor for parsing, executing, and fetching. If the cursor uses parallel execution, then ELAPSED_TIME is the cumulative time for the query coordinator, plus all parallel query slave processes. This value is divided by number of executions.
- `**Rows` - Total number of rows the parsed SQL statement returned. This value is divided by number of executions.
- `**Fetches` - Number of fetches associated with the SQL statement. This valid is divided by number of executions.
- `**Buffer Gets` - Number of buffer gets for this child cursor. This value is divided by number of executions.
- `SQL Patch` - Name of SQL Patch used to produce this plan, if any.

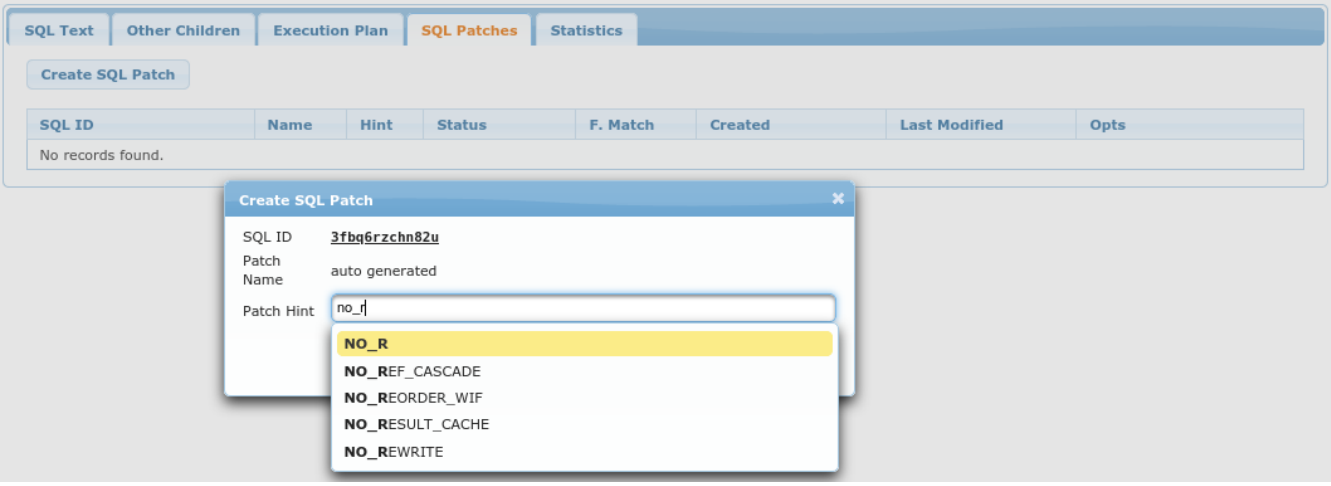
Execution Plan



This tab display execution plan for selected child number. Plan is displayed either with `dbms_xplan` or graphically if `Output Type` is selected to be `appm (graphical)`.

`XPlan Format` determines options passed directly to `dbms_xplan`.

SQL Patches



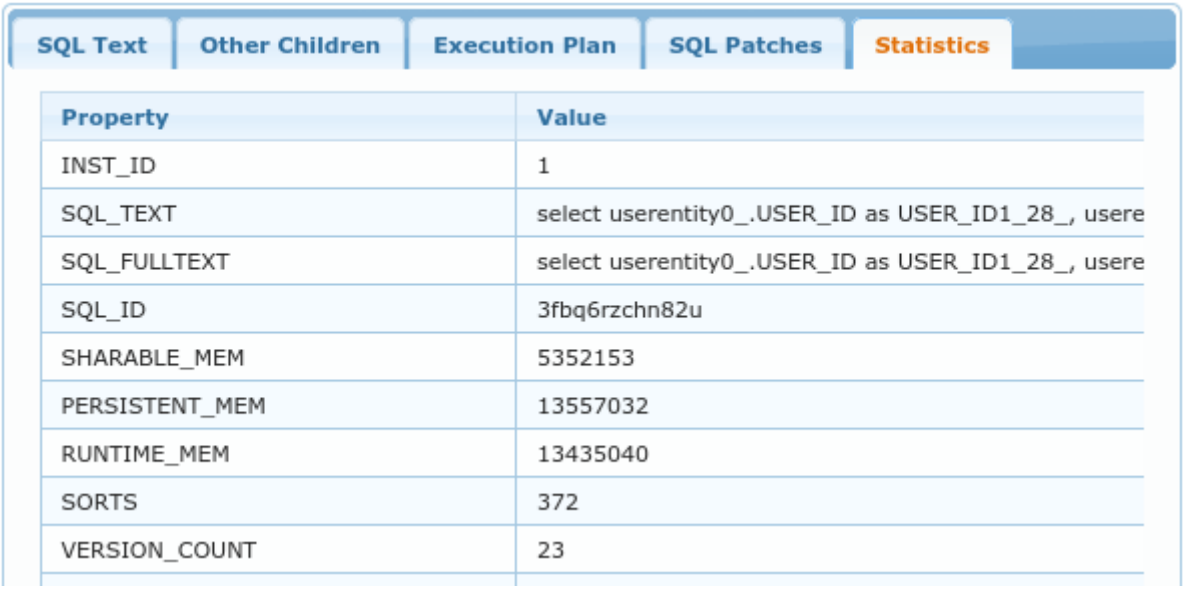
This is where you can create new `SQL Patch` for selected sql_id. List of all SQL Patches (for all `sql_ids`) is also available at `Performance → SQL Patches` (@todo: link). Columns of this table are also explained under `Performance → SQL Patches`.

`Create SQL Patch` opens up the dialog in which you can enter:

- `Patch Name` - *optionally*, it is auto-generated if you don't enter it
- `Patch Hint` - actual hint to apply to given `sql_id`.
- `Enable block input` - is used to display textarea where you can paste *Outline Data* from plan (or specify a bunch of hints manually).

Note that after you click `Create SQL Patch` the sql is automatically flushed from SGA because we want it to be hard parsed next time it executes. This also means that after creating a Patch this view won't be able to display any data regarding given `sql_id`, because it is not available anymore (until the next execution).

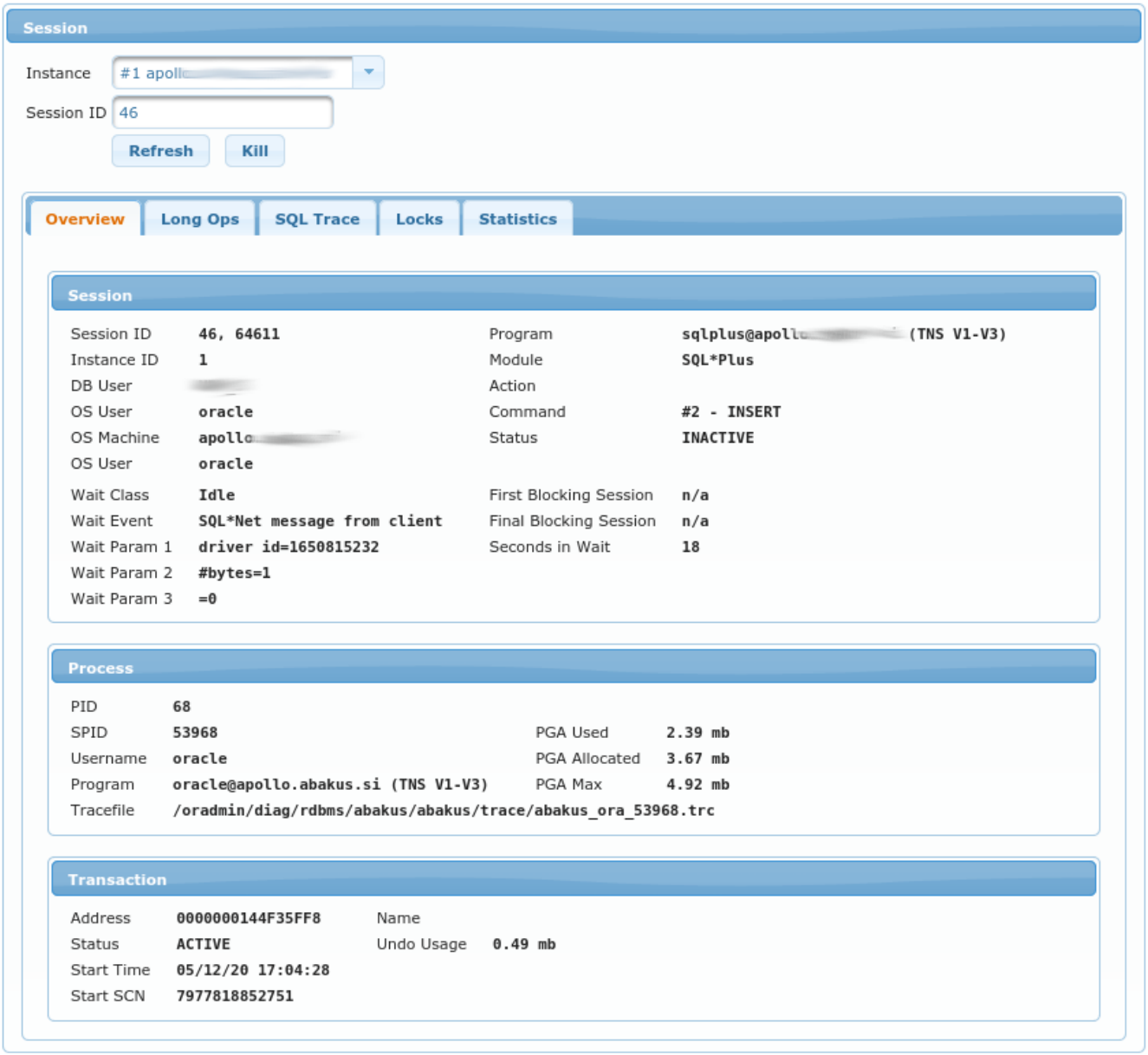
Statistics



Property	Value
INST_ID	1
SQL_TEXT	select userentity0_.USER_ID as USER_ID1_28_, usere
SQL_FULLTEXT	select userentity0_.USER_ID as USER_ID1_28_, usere
SQL_ID	3fbq6rzchn82u
SHARABLE_MEM	5352153
PERSISTENT_MEM	13557032
RUNTIME_MEM	13435040
SORTS	372
VERSION_COUNT	23

This tab displays all columns from `v$sqlarea` for given `sql_id`.

Session Detail



Session view displays session details based on `Instance` and `Session ID` (those two fields refer to `gv$session.inst_id` and `v$session.sid`).

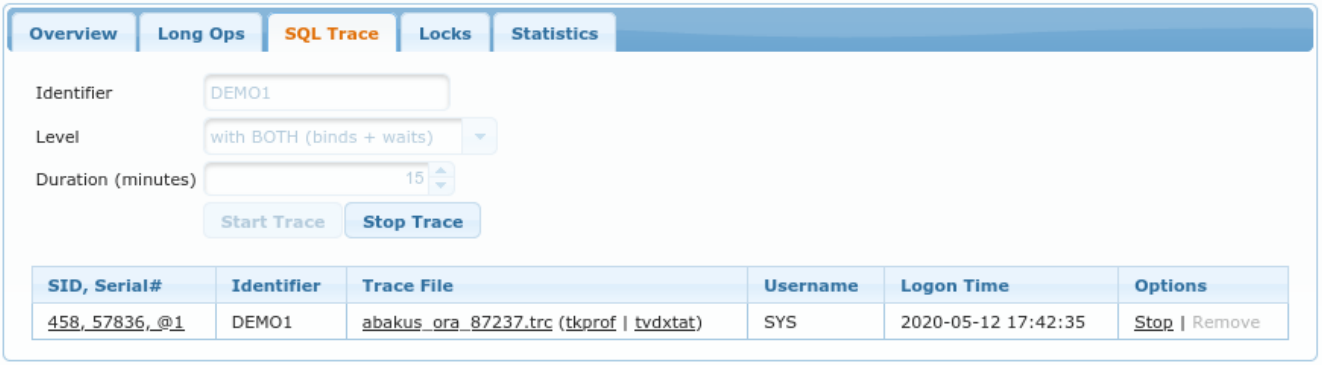
Under **Overview** tab, there are columns from:

- `v$session`
- `v$process`
- `v$transaction`

Longops

The table displays longops for this session. Columns are as documented in the official Oracle documentation: `v$session_longops`

SQL Trace



Here you can enable SQL Trace-ing for selected session.

- `Identifier` is the name for APPM so that you can later find trace files according to those identifiers.
- `Level` is the level for SQL Trace, you can trace with or without bind variables and waits (or with both) - the more details that you require, the bigger the trace files becomes :)
- `Duration` stops SQL Trace after this amount of minutes. This option here is because we don't want to enable tracing and then forget that we enabled it.

Note that tracing a session might have performance implications on the traced session.

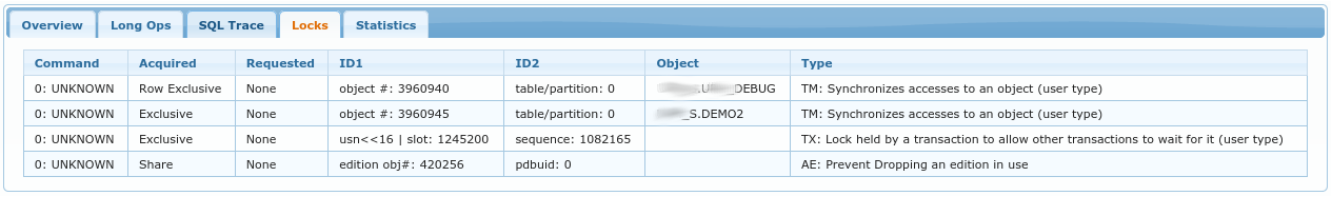
The table lists all the trace files that are available for this session. You can either download raw trace file or send it through `tkprof` or `tvdxtat` (those are both tools that allow you to create aggregated report from raw trace files).

You can also `Stop` tracing or `Remove` the trace file. Note that you cannot remove a trace file of a running session - this is why most of the time this `Remove` option is grayed-out.

All the available trace files (even for sessions that no longer exist) are available in `Performance` → `SQL Trace` view.

@todo: insert link to SQL Trace documentation

Locks



This tab displays list of all the locks that this session is holding. Data displayed is based on `v$lock` performance view.

Columns in the table are:

- `Command` is the type of command that session is currently executing
- `Acquired` is the lock type that is currently acquired

- `Requested` is the lock type that is being requested (waited on)
- `ID1` this column also displays comment of *what* this id is, e.g. `object #`
- `ID2` this column also displays comment of *what* this id is, e.g. `table/partition`.
- `Object` if `ID1` or `ID2` are referencing object from `dba_objects` then this column shows what object is being referenced.
- `Type` what kind of lock this is, column also includes comment about the type.

Statistics

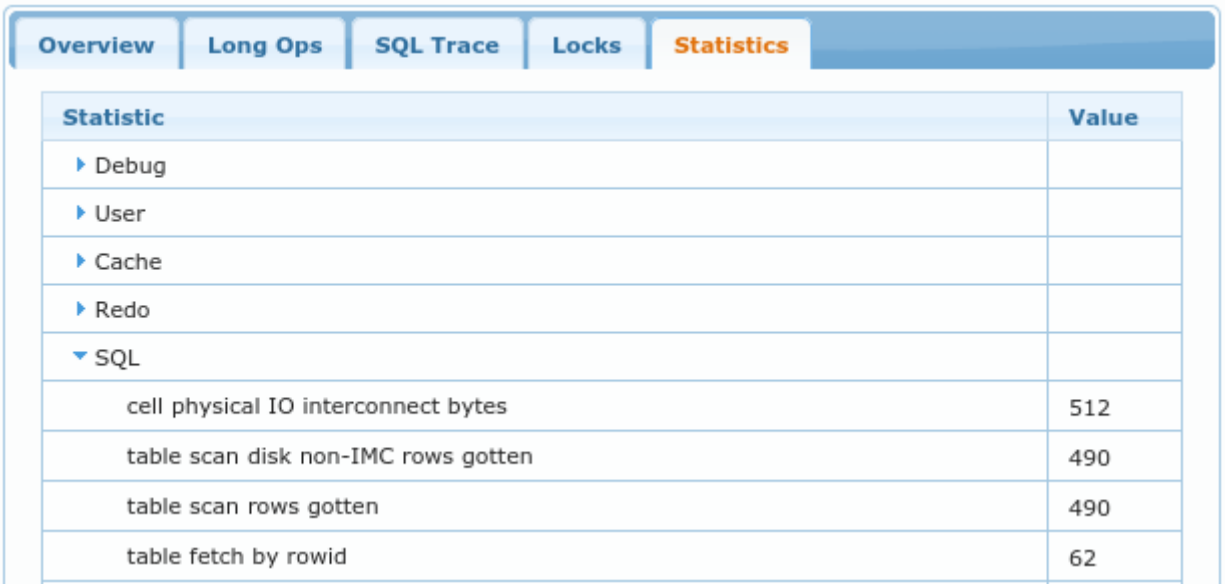


Table displays various statistics from `gv$sesstat`.

Storage (using active connection)

Storage → ASM Diskgroups & Disks

ASM Diskgroups

Group Name	State	Type	Total GB	Free GB	Required Free GB	Used PCT	Usable File GB	Offline Disks	Graph
BACK	CONNECTED	FLEX	33534	22724	0	32%	0	0	
DATA	CONNECTED	FLEX	15262	3769	0	75%	0	0	
REDO1	CONNECTED	EXTERN	300	204	0	32%	204	0	
REDO2	CONNECTED	EXTERN	300	204	0	32%	204	0	

ASM Disks

DN	Disk Name	Disk Group	Fail Group	Path	Mount Status	Header Status	State	Redundancy	Total GB	Free GB	Used PCT	Graph
0	BACK_0000	BACK	BACK_0000	/oradata/+ASM/hdd01	CACHED	MEMBER	NORMAL	UNKNOWN	5589	3787	32%	
1	BACK_0001	BACK	BACK_0001	/oradata/+ASM/hdd02	CACHED	MEMBER	NORMAL	UNKNOWN	5589	3787	32%	
2	BACK_0002	BACK	BACK_0002	/oradata/+ASM/hdd03	CACHED	MEMBER	NORMAL	UNKNOWN	5589	3787	32%	
3	BACK_0003	BACK	BACK_0003	/oradata/+ASM/hdd04	CACHED	MEMBER	NORMAL	UNKNOWN	5589	3787	32%	
4	BACK_0004	BACK	BACK_0004	/oradata/+ASM/hdd05	CACHED	MEMBER	NORMAL	UNKNOWN	5589	3787	32%	

First table displays all `ASM Disk Groups` used by this database and the second one displays all the `ASM Disks` used in those `Disk Groups`.

Report is based on the following two performance views:

- `v$asm_diskgroup`
- `v$asm_disk`

Storage → Tablespaces & Datafiles

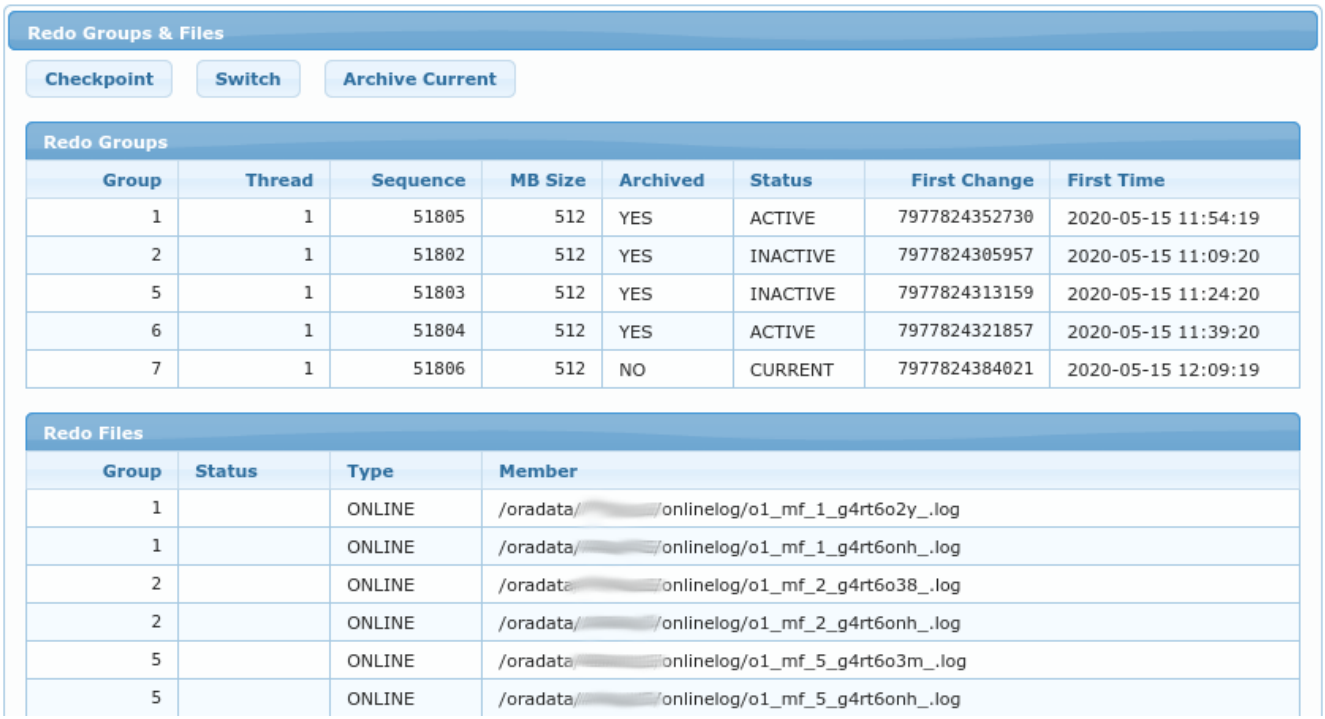
Tablespaces

Name	Parameters, status	Graph	Used GB	Allocated GB	Reserved GB	Usage
TS_STAGE	permanent, online		2357.85	2480.47	2781.25	84.78 %
TS_STAGE_INDEX	permanent, online		1101.06	1156.25	1625.00	67.76 %
USERS	permanent, online		0.53	0.59	0.98	53.78 %
SYSAUX	permanent, online		3.32	4.12	7.81	42.53 %
TS_ETL	permanent, online		23.56	36.13	62.50	37.70 %
+DATA/DWSTA/97798D260E8C2A0FE0537B00CF0A68EE/DATAFILE/ts_etl.312.1024508461	autoextend=yes, available, online		21.54	31.25	31.25	68.93 %
+DATA/DWSTA/97798D260E8C2A0FE0537B00CF0A68EE/DATAFILE/ts_etl.459.1034674223	autoextend=yes, available, online		2.01	4.88	31.25	6.46 %

This table displays list of all `Tablespaces` and their datafiles. Columns are:

- `Name` - name of the tablespace or path to specific datafile
- `Parameters, status` - type of tablespace and info about autoextend for specific datafile
- `Graph` - yellow color means used space, blue means allocated space and light-gray is all the reserved space.
- `Used GB` - amount of space occupied by segments
- `Allocated GB` - amount of space allocated by datafiles (this much of disk space is used from OS perspective)
- `Reserved GB` - amount of space available according to autoextend attribute
- `Usage` - in percent, how much space is used (available space is considered to be `Reserved GB`)

Storage → Redo Groups & Files



Redo Groups & Files

Checkpoint | Switch | Archive Current

Redo Groups

Group	Thread	Sequence	MB Size	Archived	Status	First Change	First Time
1	1	51805	512	YES	ACTIVE	7977824352730	2020-05-15 11:54:19
2	1	51802	512	YES	INACTIVE	7977824305957	2020-05-15 11:09:20
5	1	51803	512	YES	INACTIVE	7977824313159	2020-05-15 11:24:20
6	1	51804	512	YES	ACTIVE	7977824321857	2020-05-15 11:39:20
7	1	51806	512	NO	CURRENT	7977824384021	2020-05-15 12:09:19

Redo Files

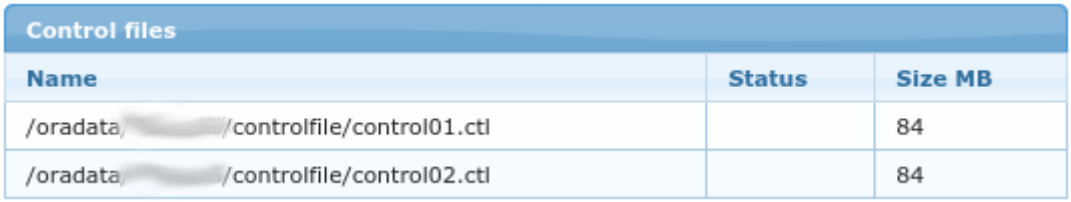
Group	Status	Type	Member
1		ONLINE	/oradata/[redacted]/onlinelog/o1_mf_1_g4rt6o2y_.log
1		ONLINE	/oradata/[redacted]/onlinelog/o1_mf_1_g4rt6onh_.log
2		ONLINE	/oradata/[redacted]/onlinelog/o1_mf_2_g4rt6o38_.log
2		ONLINE	/oradata/[redacted]/onlinelog/o1_mf_2_g4rt6onh_.log
5		ONLINE	/oradata/[redacted]/onlinelog/o1_mf_5_g4rt6o3m_.log
5		ONLINE	/oradata/[redacted]/onlinelog/o1_mf_5_g4rt6onh_.log

Buttons at the top are only available to `SYSDBA` users. They perform the following actions:

- `Checkpoint` - Performs checkpoint ("dirty" buffers are flushed to datafiles)
- `Switch` - Current log group is advanced to the next one
- `Archive Current` - Same as switch, but also archive the current one immediately.

Table `Redo Groups` displays data from `v$log` and table `Redo Files` displays data from `v$logfile`.

Storage → Control Files



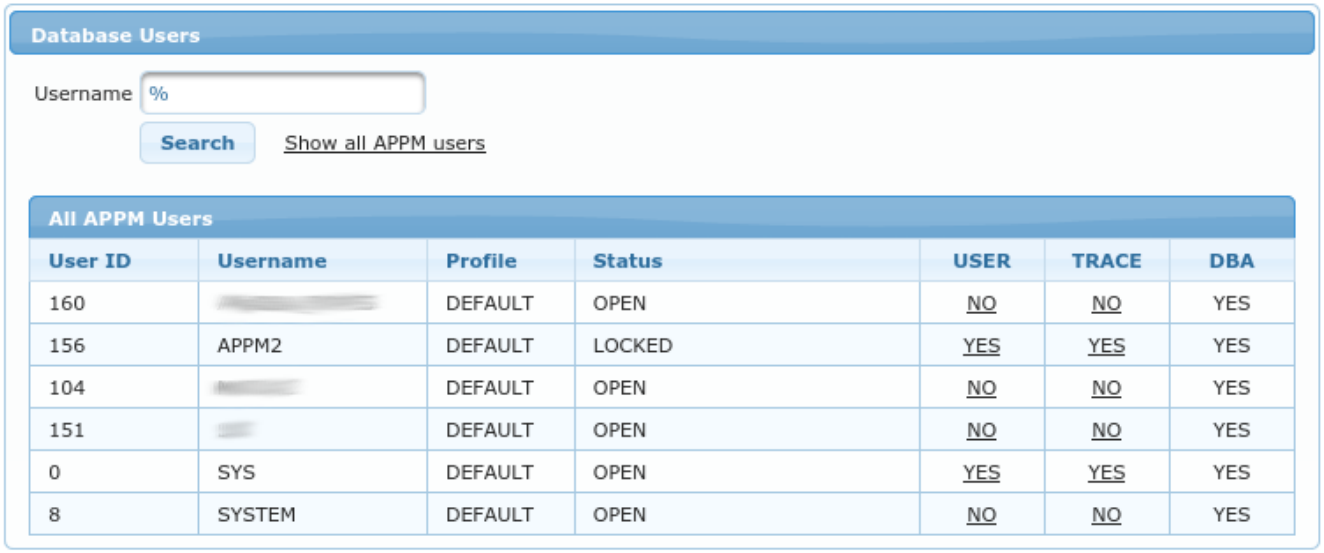
Control files

Name	Status	Size MB
/oradata/[redacted]/controlfile/control01.ctl		84
/oradata/[redacted]/controlfile/control02.ctl		84

Displays list of all control files - with their status and size in MB.

Management

Database → Users



User ID	Username	Profile	Status	USER	TRACE	DBA
160	[redacted]	DEFAULT	OPEN	NO	NO	YES
156	APPM2	DEFAULT	LOCKED	YES	YES	YES
104	[redacted]	DEFAULT	OPEN	NO	NO	YES
151	[redacted]	DEFAULT	OPEN	NO	NO	YES
0	SYS	DEFAULT	OPEN	YES	YES	YES
8	SYSTEM	DEFAULT	OPEN	NO	NO	YES

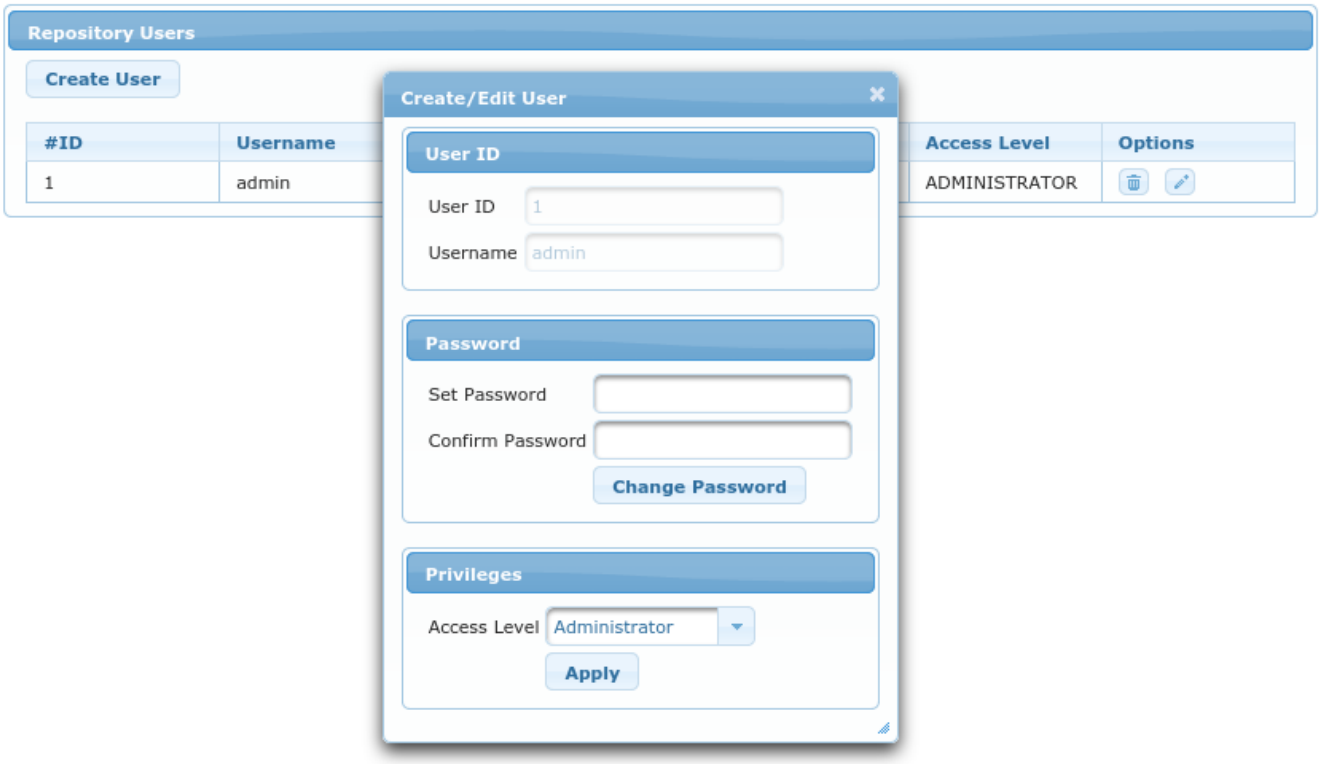
On this page you can define which Oracle users have access to database via APPM. From DBA perspecitve, APPM uses following database roles:

- `APPM_USER_ROLE` - allows to query performance-related data through APPM application. It does **not** allow to change any data or to perform any actions on database (e.g. killing or tracing database sessions)
- `APPM_TRACE_ROLE` - allows everything that APPM_USER_ROLE allows and also adds privilege to perform SQL Trace of any session.
- `DBA` or `SYSDBA` - user can also perform some critical actions through APPM (such as killing database sessions).

Table in screenshot shows list of database users. By clicking on columns `USER` and `TRACE` you can grant/revoke previoulsy described `APPM_USER_ROLE` and `APPM_TRACE_ROLE`. You cannot grant `DBA` or `SYSDBA` through GUI.

Table is displaying data from `dba_users` view.

Repository → Users

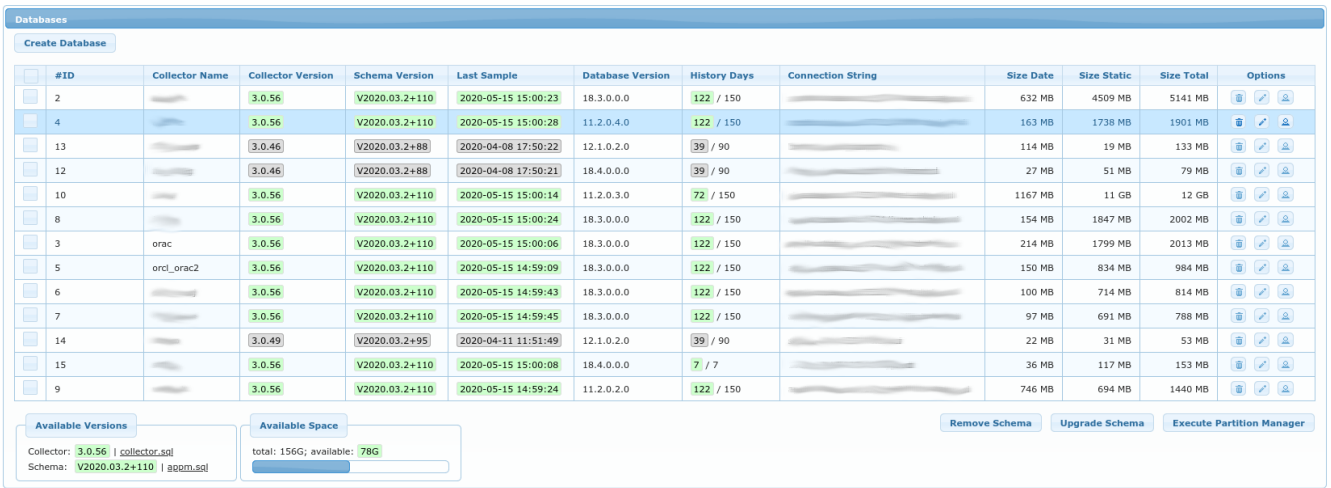


Repository users have access to APPM application and can query all sampled data. They cannot (without additional Oracle users as described in previous chapter) connnect to running Oracle instances.

There are two access levels:

- `Users` - can query anything in a repository (but cannot make any changes)
- `Administrators` - can also register new databases (or edit existing ones) and add/modify Repository Users

Repository → Databases



This is a list of all registered databases for which the APPM can receive samples (from APPM Collector, see Installation Guide on how to configure collector).

Each registered Oracle database can have following APPM relates schemas installed: * `APPM2` schema

(can be named differently) holds packages and views used by APPM application. This schema is (and should be) LOCKED. * APPM_COLLECTOR user (can be named differently) can connect to database and query only specific view in APPM2 schema, through which it obtains performance samples.

Following buttons are available:

- Create Database - opens a popup to enter details for new database
- Remove Schema - removes APPM2 schema from Oracle Database. Popup will ask for SYSDBA password in order to do that.
- Upgrade Schema - upgrades APPM2 schema on selected Oracle Databases. Popup will as for SYSDBA password in order to do that.
- Execute Partition Manager - This is done automatically once per day. You can execute it on demand by using this button. It removes obsolete data and creates partitions until today + 3 days.

Note that following two links also exist if you are not keen on entering SYSDBA password into application:

- collector.sql - creates/upgrades APPM_COLLECTOR schema on Oracle database if run as sqlplus / as sysdba @collector.sql
- appm.sql - creates/upgrades APPM2 schema on Oracle database if run as sqlplus / as sysdba @appm.sql

For each database, following parameters are stored:

- Database ID - autogenerated numerical id for each database
- Collector Name - unique name of database (APPM Collector refers to this name in its collector.ini)
- Collector Enabled - flag to know which databases are not meant to recieve any more samples (e.g. obsolete databases)
- Repository Schema - name of PostgreSQL schema which contains samples for this database
- Repository Tablespace - name of PostgreSQL tablespace which contains samples for this database
- Connection String - Oracle connection string in form of **hostname:port/service_name**
- APPM Schema - Oracle schema name, usually named APPM2
- History Days - For how many days to store samples. Most data is partitioned by dates. Partition Manager deletes data older than this number of days.

Options columns have the following buttons:

- Remove Database - which removes the database and all of its samples from repository (it does not connect to Oracle database)
- Edit Database - modify parameters of database. Usually History Days is modified to match available space required. Not all options can be changed after database is registered (e.g. Repository Schema cannot be change after created).

- `Edit Credentials` - (optional), used to store Oracle credentials used to connect to Oracle Database. This option should be used with care.

Databases

`Databases` table has some of its columns colored green, yellow or gray.

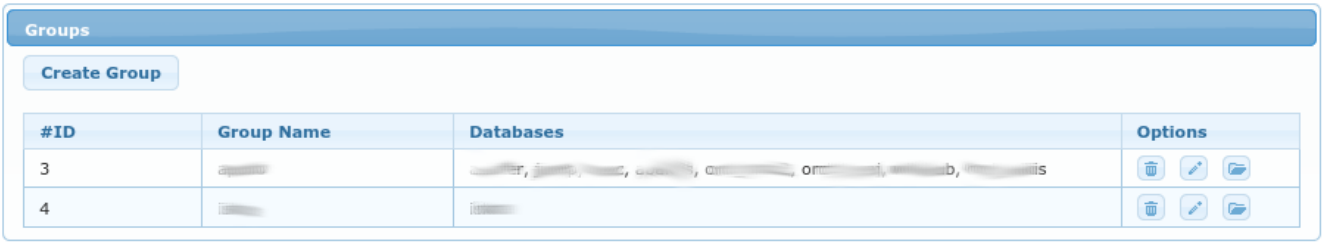
- green means that samples are up2date and version of `APPM2` schema is up2date
- yellow means that either samples are not collected in last 24h or that `APPM2` schema is outdated. It can also mean that APPM Collector software is outdate and may not be fully compatible with this version of APPM Repository.
- grey means that collector is disabled for this database so we don't care whether it is outdated or not (nothing is being collected for this database)

Available Versions & Available Space

On the bottom of the screen is a chart of how much space is still available for samples. This usually referrs to amount of space available under `/srv/appm` mountpoint.

You can free some space by setting lower `History Days` and afterward `Execute Partition Manager` or you can increase `History Days` if you see that you still have plenty of space available.

Repository → Groups



Here you have option to view databases as a group. You can think of group as a group of:

- all databases on the same host
- all pluggable databases in the same container
- all RAC databases in the same cluster

One database can be a member of many groups. Currently, only Dashboard displays charts based on groups that are defined here.