

# Oracle Workload Characterization

Andy Rivenes  
AppsDBA Consulting

## Abstract

Workload characterization is the process of identifying classes of workload, measuring those classes and then identifying their impact to the business. The goal is to understand the impact of those classes on database workload and to enable the organization to better schedule its business processes.

## Introduction

It's one thing to know that your database server is running at some percent of its capacity, that you have specific peak workload periods and that during those peak workloads your database server is approaching its maximum capacity. It is another matter entirely to actually understand what is responsible for that workload, and to be able to identify the specific business processes that comprise that workload. Workload characterization is the process of identifying business processes, or classes of workload, and measuring their contribution to overall system workload. Only when this information is available can true workload management occur. It becomes possible to estimate what the impact of re-scheduling business processes will have on overall system workload. It may also help identify workload waste, and opportunities for insuring that critical business processes have enough compute resources available to insure timely completion. And lastly, it can give the business valuable insight into how their applications actually use their computer resources.

My original goal for this paper was to take the workload utilization of an Oracle database at a given point in time and be able to create a simple pie chart that showed the contribution to total workload broken down by workload "classes". This simplistic goal morphed into a more formal exploration of how to characterize Oracle database workload. The result is this paper, which will explore how to identify and classify workload in an Oracle database, and will discuss some of the measurement options available.

## Workload Characterization

What is workload characterization? The term gets mentioned a lot, but really defining what it means in a practical sense has always been difficult for me. I think that workload characterization can be best summed up as, what is using the "system" and how much it is using it? Menascé and Almeida [3] define workload characterization as follows:

Process of partitioning the global workload of a computer system into smaller sets or workload components composed of transactions or requests having similar characteristics, and assigning values that represent their typical resource demand and intensity.

In an article titled "An Introduction to Workload Characterization"[2], Ron Lee credits "Dr. Domenico Ferrari, a professor in the Computer Science Division of the University of California at Berkeley" with the following concept of "levels" of workload characterization:

**Figure 2: Levels of Workload Characterization**

Functional Level	The User's View
Logical Level	The Programmer's View
Physical Level	The Computer's View

I like this view of workload characterization, and for the purposes of our discussion I believe that viewing workload at the functional level allows us to classify workload into "classes" or groups that are then easy to attribute to workload. This leads to a simple pie chart as illustrated in figure 1. By breaking down usage into classes we can create a "resource profile" view of workload.

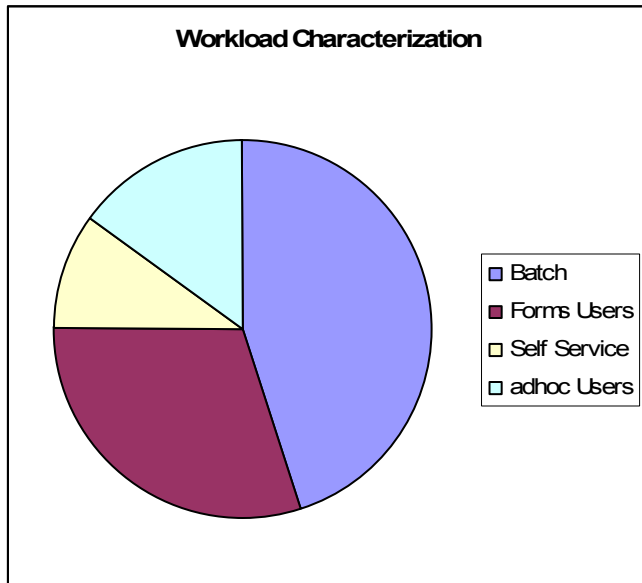


Figure 1. Sample workload characterization.

The clear advantage of being able to characterize workload this way is that we can now see what is consuming resources from a business perspective. In this case, if we were interested in reducing workload on the system, the logical starting point would be "Batch" workload, since it is consuming the largest share of resources. Now, this is not to say that there is a performance problem. Workload characterization is not a

performance metric. It is simply a tool that helps us identify what is consuming resources and can indicate areas to target for performance optimization, workload reduction or work shift management.

When assessing system workload, being able to classify workload can be a huge help. In other words, if batch workload is overwhelming the capacity of a system, it may not be obvious to what degree and by how much unless you can separate it from interactive usage. This may not be as simple as you might think. You may be able to see that a lot of batch jobs are running, and that interactive usage is high as well. But in the absence of a clear bottleneck, or any one dominant job, it can be very difficult to quantify the impact of any one process type on the system as a whole. For instance, figure 1 may be representative of workload during the normal business day and interactive performance may be fine. But at night, when more batch jobs are scheduled to run, performance for overseas users may be unacceptably slow. During this work shift workload may be closer to the profile in figure 2.

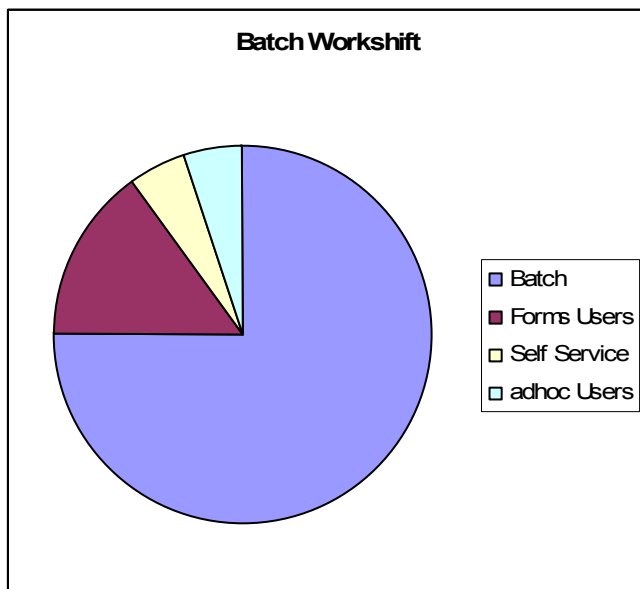


Figure 2. Batch work shift workload.

If that is the case, then steps may need to be taken to protect interactive user response times by better controlling batch workload. In the case of an Oracle Applications system fewer concurrent manager queue processes may need to be allocated. It could also be the case that some unforeseen class of transactions causes high resource consumption. Traditional workload measurement will only show utilization and not what is consuming that utilization from a business' perspective.

The following will explore how workload characterization can be done in Oracle based systems, how it can be combined with existing workload measurement and response time optimization methods, and how it can be used to better manage existing workload usage.

## Methodology

The methodology behind workload management is composed of the following:

- Workload measurement
- Workload characterization
- Workload classification

Workload measurement is the process of measuring the work performed by the Oracle database. This process is covered in great detail in the AppsDBA paper, “Oracle Workload Measurement”[4]. Workload characterization and workload classification are the topics of this paper. These processes lead to the ability to perform the following more efficiently:

- Capacity planning
- Workload reduction
- Performance optimization

With an interval based workload measurement tool, trending information about the database workload over time allows for more effective capacity planning. This topic is covered in more detail in “Oracle Workload Measurement” as are the last two items, workload reduction and performance optimization.

## Workload Classes

So, the real question is how can we create the pie chart? Clearly we have to be able to define groups of workload, and obviously these will be dependent on the type of application(s) running in the database. Unfortunately Oracle has not made this particularly easy to do. Oracle is well instrumented to provide the logical and physical level workload information through various v\$ views and extended SQL trace data, but at the functional level we are dependent on the application to provide us with enough information to classify the usage that we can measure.

If we can classify workload based on functional or business processes we then have the opportunity to be able to measure the workload for those classes. This is then fairly easy to translate into some useful metric that we can visualize. Since there is no “native” method of identifying applications and their transactions in Oracle, we have to define how we expect the application to be instrumented. By default, other than the login user name, there is no easy way to differentiate what kind of transaction is being executed or even by who if common login users are used. Fortunately Oracle has provided tools to do this and continues to expand them with newer releases.

Instrumentation has become a popular topic recently, but Oracle has made features available since as early as Oracle version 7.2 to aid the application developer in the

instrumentation of application code. The package DBMS\_APPLICATION\_INFO has procedures available to the application developer that allow the setting of MODULE and ACTION information in the database. This has been exposed in the database through the v\$session view<sup>1</sup>. DBMS\_APPLICATION\_INFO also supports setting a CLIENT\_INFO field. This field allows the application programmer to provide information about the connected session.

Another ability that has been available since at least Oracle version 7 is the ability to audit sessions. This is a very low cost method of identifying the work of an entire database session. Upon login a row is inserted into an audit table and then at logoff that row is updated with session usage information. If the application uses individual logons then this can be a useful method for tracking database usage. The downside is that the usage information is only updated at logoff, so if the application uses "pooled" connections there may not be any way to separate information about individual application sessions. However, in many cases this may not pose too large of a problem since the pooled connections can typically be classified as one group and interval based collection can typically capture the usage information.

There are other methods of collection as well. Logoff triggers can be used (similar to session auditing but allows a more tailored approach to the information collected), interval based snapshots can be used, and application level logging can be used (e.g. using application code to track the usage information).

## **Instrumentation**

The following will detail some of the methods that can be employed to provide application instrumentation. This in turn then provides the mechanism to allow the capturing of workload data by classification. The key to being able to classify workload in an Oracle database is in the ability to identify and then measure that workload. The following section will explore some of the features available in Oracle to instrument applications.

### **Session Identification**

The first step in being able to identify workload is to be able to identify the users and their sessions that are connected to the database. Figure 3 shows the formatted output from a query of the view v\$session from an Oracle9i database. Note that there are many more columns available in the v\$session view, but these columns pertain to identifying a particular session.

---

<sup>1</sup> It should be noted that these fields are also recorded in extended SQL trace data as well, and can greatly aid in the analysis of individual process cost.

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
ELLIS13	942	39481		iaswww-1	httpd@iaswww-1 (TNS V1-V3)		
DCHR	48	58159		llapp-3	llserver@llapp-3 (TNS V1-V3)		
DCHR	217	51993		llapp-3	llserver@llapp-3 (TNS V1-V3)		
WEB	835	16059		iasapp-1 Client	JDBC Thin		

Figure 3. Formatted output from v\$session (Oracle9i).

Figure 4 shows the addition of the "SERVICE\_NAME" column that was added to the v\$session view in Oracle 10g.

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action	Service Name
AUTHENSVR_MGR	172	10628		nspws-2				SYS\$USERS
COLLAB_MGR	173	7518		nspws-1				SYS\$USERS
COLLAB_MGR	175	343		nspws-2				SYS\$USERS
SYSTEM	177	5050		nspdb-1		SQL*Plus		SYS\$USERS
AUTHENSVR_MGR	180	9876		nspws-1				SYS\$USERS
COLLAB_MGR	182	612		nspws-1				SYS\$USERS
AUTHENSVR_MGR	183	228		nspws-1				SYS\$USERS
COLLAB_MGR	187	7044		nspws-2				SYS\$USERS
COLLAB_MGR	188	669		nspws-1				SYS\$USERS
AGOLDNER	189	11281		AMANDA-PC				SYS\$USERS
COLLAB_MGR	191	5491		nspws-2				SYS\$USERS
COLLAB_MGR	193	6561		nspws-2				SYS\$USERS

Figure 4. Formatted output from v\$session (Oracle 10g).

Services provide an extremely powerful mechanism for classifying workload and will be explored further in the next section. By default, foreground sessions are assigned the service name "SYS\$USERS" and background sessions are assigned the service name "SYS\$BACKGROUND".

Oracle has also added additional attributes to extended SQL trace data as well:

```
*** ACTION NAME:() 2005-08-26 11:26:18.647
*** MODULE NAME:(SQL*Plus) 2005-08-26 11:26:18.647
*** SERVICE NAME:(SYS$USERS) 2005-08-26 11:26:18.647
*** SESSION ID:(202.11594) 2005-08-26 11:26:18.647
```

Previously this information was not available. Prior to 10g only the "SESSION ID" attribute was available. Oracle did provide the APPNAME line when an application set either module or action using dbms\_application\_info.

```
*** SESSION ID:(296.24736) 2003-11-20 16:14:07.011
APPNAME mod='SQL*Plus' mh=3669949024 act='' ah=4029777240
```

## Services

Services are controlled as part of the initial database connection. The intent of a service is to enable the grouping of common database connections or allowing the distribution of connections (e.g. RAC). There may be multiple applications or application functions in a database that can be grouped into different services. Oracle is also using services to classify workloads. Classifying database usage into multiple services allows for finer control of resources, allows better identification and measurement, and provides an isolation level from the actual database.

For example, an HR application may run on a database called PSPRD and have an Oracle Networking connect string connecting to the PSPRD database. Now if that application were to be moved to the OAPRD database those connect strings that refer to the PSPRD database would need to change to the OAPRD database. If however, the SERVICE\_NAMES parameter had been used, then only the Oracle Networking configuration files would have needed changing. An even more important benefit is that now database connections can be uniquely identified by the "service" making the connection, as in HR. Although a simplistic example, Oracle has already used the service concept in both Oracle Networking and Oracle RAC to distinguish between application services. Now Oracle has exposed the service name connection to the database through the SERVICE\_NAME column in v\$session.

Oracle has also taken this a bit further with the ability to control services through the DBMS\_SERVICE package and in Enterprise Manager.

## Client Identification

The "client identifier" field in the v\$session view is meant to allow application user identification when using shared connections. In an Oracle Applications database this attribute is used by various database components to identify lightweight application users who authenticate as the same database user. As an example I might sign on to the application as "Rivenes1", but when I'm connected to the database I will be connected as the "apps" user. By setting the client identifier to "Rivenes1" there is the ability to know who that connected user is. Oracle provides the DBMS\_SESSION.SET\_IDENTIFIER procedure, the OCI attribute OCI\_ATTR\_CLIENT\_IDENTIFIER, or the Java method Oracle.jdbc.OracleConnection.setClientIdentifier. Figure 5 shows an example using the DBMS\_SESSION.SET\_IDENTIFIER procedure.

```

User      Oracle      Client      Client      Client
Name      SID Serial#  Identifier  Machine      Info      Module      Action
-----
APPS      164   15069      client      client
SQL>

SQL> exec dbms_session.set_identifier('Rivenes1');

PL/SQL procedure successfully completed.

SQL>

The resulting v$session entry:

User      Oracle      Client      Client      Client
Name      SID Serial#  Identifier  Machine      Info      Module      Action
-----
APPS      164   15069  Rivenes1   client
SQL>

```

Figure 5. Client Identification.

## Application Information

As described earlier application information can be set with the `dbms_application_info` package. By default, Oracle populates the "module" column of `v$session` with the same value as the "program" column. The following shows the "default" settings for a given session:

```

USERNAME      PROGRAM      MODULE
-----
STATS         JDBC Thin Client  JDBC Thin Client

```

There are three specific calls in `dbms_application_info` that are of interest when instrumenting an application for workload characterization. These are the `set_client_info`, `set_module` and `set_action` procedures. Each of these will be explored in this section and how they affect the columns in the `v$session` view.

### set client info

The Oracle documentation states that the `SET_CLIENT_INFO` procedure "Supplies any additional information about the client application."<sup>2</sup>. The following shows the use of the `SET_CLIENT_INFO` procedure to set the value of Client Info to "DBAMON".

<sup>2</sup> Oracle9i Supplied PL/SQL Packages and Types Reference, Release 2 (9.2)

### Initial connect, v\$session:

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
PERFSTAT	164	15069		appsdba		SQL*Plus	

### SQL command:

```
SQL> exec dbms_application_info.set_client_info('DBAMON');
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

### v\$session:

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
PERFSTAT	164	15069		appsdba	DBAMON	SQL*Plus	

### set module

The SET\_MODULE procedure enables the application developer to set the current application module name. This might simply be the name of the application running or it could be the name of the procedure or function being run. The SET\_MODULE procedure allows the "action" to be set as well (this can also be set independently with the SET\_ACTION procedure). The following shows the setting and unsetting of these fields in the v\$session view.

### SQL command:

```
SQL> exec dbms_application_info.set_module('module','step1');
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

### v\$session view:

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
PERFSTAT	164	15069		appsdba	DBAMON	module	step1

### SQL command:

```
SQL> exec dbms_application_info.set_module(null,null);
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

## v\$session view:

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
PERFSTAT	164	15069		appsdba	DBAMON		

## set action

The SET\_ACTION procedure sets just the action. The "Action" field will be set in the v\$session view and in the extended SQL trace file.

## SQL command:

```
SQL> exec dbms_application_info.set_action('step2');
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

## v\$session view:

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
PERFSTAT	164	15069		appsdba	DBAMON	module	step2

## SQL command:

```
SQL> exec dbms_application_info.set_action(null);
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

## v\$session view:

User Name	Oracle SID	Serial#	Client Identifier	Client Machine	Client Info	Module	Action
PERFSTAT	164	15069		appsdba	DBAMON	module	

## Tools

These fields can also be set in different ways depending on the tool used. SQL\*Plus provides the ability to set the module field with the "set appinfo" command. The following shows an example in SQL\*Plus for 10g Release 2:

```
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
```

```
SQL> set pages 9999
SQL> show appinfo
appinfo is OFF and set to "SQL*Plus"
SQL> set appinfo on
SQL> show appinfo
appinfo is ON and set to "SQL*Plus"
SQL> set appinfo 'AppDBA Example'
SQL> variable mod varchar2(30)
```

```

SQL> variable act varchar2(30)
SQL> execute dbms_application_info.read_module(:mod, :act);

PL/SQL procedure successfully completed.

SQL> print mod

MOD
-----
AppDBA Example

SQL> print act

ACT
-----

SQL> show appinfo
appinfo is ON and set to "AppDBA Example"
SQL> set appinfo off
SQL>

```

## Collection

Performing workload characterization touches on many of the same issues that create problems when trying to perform workload measurement at the session level. Typically we want to use the `v$session` and `v$sesstat` views when trying to capture session level workload information, and invariably we discover that we're missing data. Potentially lots of data. I cover this problem in detail in the paper "Oracle Workload Measurement"[4], but suffice it to say that many types of session workload don't lend themselves well to interval snapshots.

How workload information is collected and classified then will be highly dependent upon the application and its workload characteristics. It is important to note that we really aren't interested in individual sessions anyway, but instead want to focus on the classes of workload in the system. What we're really after is being able to classify the aggregate types of workload in a system for any given interval. The reality of this is that we typically have to use a combination of interval sampling and logoff totals to accomplish our collection.

In an Oracle Applications system for example, concurrent manager jobs have a broad enough mix of runtimes that collection by interval is usually not very productive. This type of workload tends to lend itself better to logoff collection with a logoff trigger. There is still the problem of cross interval time spanning, but for the most part a logoff trigger provides better accuracy than interval sampling. Forms users on the other hand tend to stay connected for a longer period of time and a logoff trigger misses the peaks and valleys of that type of workload. Therefore interval sampling tends to provide a more accurate picture of the workload over a given interval.

In general connections that span collection intervals should use interval snapshots, and connections that last less than a collection interval should use some form of logoff based trigger. The bottom line here is that the collection technique has to be tailored to

the type of workload being measured, and that requires knowledge of the system being analyzed.

## Classifying Workload

Characterizing workload is the process of identifying *classes* of workload, measuring those classes and then calculating the costs of those classes of workload. The purpose of workload characterization is to be able to allocate available capacity to the classes that are most important to the business.

The following will provide an example of how instrumentation can be used to classify workload in an Oracle database. Four basic steps are explored: identify, classify, measure and interpret.

### Identify

The first step is to identify workload classes for the system being diagnosed. In general all systems will be different. However, there are some classes that will be common to similar types of applications. The script *usercat.sql* in Appendix A will produce the output in figure 6 and can be used to help identify workload classes.

User Name	Client OS User	Client Machine	Program	Module	Action	Logical Reads	Physical Reads
APPS	applmgr	app1		FNDESCGN	US PO Inquiry	3,023	13
APPS	applmgr	app2		POXPOVPO	FRM:JULIE :US PO Inquiry	3,430	104
APPS	applmgr	app2		FNDESCGN	US PO O Inquiry	3,101	26
APPS	applmgr	app2	httpd@app2 (TNS V1-V3) V1-V3)	httpd@ app2 (TNS V1 -V3)		18,598	679

Figure 6. Workload Identification

### Classify

Workload classes can be created once the types of workload are identified. Figure 7 shows an example of workload classes as defined in the AppsDBA WORKMON utility. In this case the tool isn't important it's the ability to define workload classifications.

Class Description	Class Type	Column Name	Column Value
Background processes	INTERVAL	TYPE	BACKGROUND
APP User	INTERVAL	USERNAME	APP_DATA
Interval Totals	INTERVAL	USERNAME	
LOADING_META_DATA User	INTERVAL	USERNAME	LOADING_META_DATA
SQL*Plus Connections	INTERVAL	PROGRAM	SQLPLUS
STATS User	INTERVAL	USERNAME	STATS

Figure 7. Workload Classes

## Measure

Once workload classes are identified then measurements can be grouped by class and summarized. An example is shown in figure 8. Again, this was created with the AppsDBA WORKMON utility, but could be created in any variety of ways.

Run Date	Intvl	Description	Count Total	Logical Reads	Physical Reads	CPU Time (min.)
01/26/2006	00:10	Interval Totals	512	80,887,158	2,350,458	19.20
		OA Forms Users	61	34,425,797	113,017	5.63
		Ext App	41	1,195,564	153,933	1.77
		Custom Users	11	236,176	186,101	0.69
		Background processes	11	30,366	24,955	0.00
		Interface App	24	13,198	3,321	0.05
		OA Self Service	9	22	0	0.00
		Concurrent manager jobs	11	0	1,898,109	0.08

Figure 8. Workload Class Measurements

## Interpret

Interpretation leads to understanding the overall impact of each workload class on the total workload. Based on the information available a pie chart or some other visual tool can be created and decisions can then be made about work shift allocations or redistribution, and about possibly re-scheduling work to non-critical time periods.

## Conclusion

Workload characterization is identifying, classifying and then measuring workload with the intent of identifying different types, or classes, of workload and their contribution to overall system level workload. The goal is not performance tuning, but rather an understanding of the costs of workload classes with the goal of further aiding the efforts of workload management. The ability to identify workload class contributions can enable the enterprise to then manage workload and capacity planning. In the case of managing

workload, the ability to shift or reduce one or more classes of workload in order to gain “headroom” for another class of workload is a common and important task. Especially when done around known peak processing periods (i.e. like quarter end or year end processing). Also understanding the impact of different types of workload classes to the environment can be an eye opener. The bottom line is that workload characterization is another valuable tool that encompasses workload management.

## Appendix A

### *usercat.sql*

```
-- FILE:      usercat.sql
--
-- AUTHOR:    Andy Rivenes, arivenes@appsdba.com, www.appsdba.com
--           Copyright (c) 2006, AppsDBA Consulting. All Rights Reserved.
--
-- DATE:     01/10/2006
--
-- DESCRIPTION:
--           This script will list categorize users currently logged on to the system.
--
-- REQUIREMENTS:
--
-- MODIFICATIONS:
--
--
-- SET PAGESIZE 9999;
-- SET VERIFY off;
-- SET FEEDBACK off;
-- SET LINESIZE 200;
-- SET TRIMSpool off;
--
-- COLUMN uname      HEADING 'User|Name'           FORMAT A20;
-- COLUMN suser      HEADING 'Client|OS User'       FORMAT A8;
-- COLUMN smach      HEADING 'Client|Machine'       FORMAT A20  WORD_WRAPPED;
-- COLUMN process    HEADING 'Client|Process'       FORMAT A20;
-- COLUMN sprog      HEADING 'Program'              FORMAT A15  WORD_WRAPPED;
-- COLUMN clinfo     HEADING 'Client|Info'          FORMAT A15  WORD_WRAPPED;
-- COLUMN mod        HEADING 'Module'              FORMAT A9;
-- COLUMN act        HEADING 'Action'              FORMAT A9;
-- COLUMN logr       HEADING 'Logical|Reads'        FORMAT 99,999,999,999;
-- COLUMN phyr       HEADING 'Physical|Reads'       FORMAT 99,999,999,999;
--
-- SELECT a.username uname,
--        a.osuser suser,
--        a.machine smach,
--        a.program sprog,
--        a.module mod,
--        a.action act,
--        st1.value logr,
--        st2.value phyr
-- FROM v$session a,
--      v$process b,
--      v$sesstat st1,
--      v$sesstat st2,
--      v$statname sn1,
--      v$statname sn2
-- WHERE a.sid = st1.sid
--       and st1.sid = st2.sid
--       AND st1.statistic# = sn1.statistic#
--       AND st2.statistic# = sn2.statistic#
--       AND a.paddr = b.addr
--       and sn1.name = 'session logical reads'
```

```
and sn2.name = 'physical reads'
```

## References

[1] Oracle Database JDBC Developer's Guide and Reference, 10g Release 1 (10.1), Part Number B10979-01

[2] R. Lee, "An Introduction to Workload Characterization", Novell, <http://support.novell.com/techcenter/articles/ana19910503.html>, 1991.

[3] D. A. Menascé, V. A. F. Almeida, "Scaling for E-Business, Technologies, Models, Performance, and Capacity Planning", Prentice Hall, Upper Saddle River, New Jersey, 2000.

[4] A. S. Rivenes, "Oracle Workload Measurement", AppsDBA Consulting, [www.appsdba.com](http://www.appsdba.com), 2005.

[5] C. V. Millsap, "How to Make an Application Easy to Diagnose", Hotsos Enterprises, Ltd., [www.hotsos.com](http://www.hotsos.com), 2004.

[6] WORKMON – A workload characterization utility freely available from AppsDBA.com at [http://www.appsdba.com/utilities\\_workmon.htm](http://www.appsdba.com/utilities_workmon.htm).