14

Managing Database Performance

ORACLE

Objectives

After completing this lesson, you should be able to:
- Monitor the performance of sessions and services
- Describe the benefits of Database Replay

Tuning Activities

The three activities in performance management are:

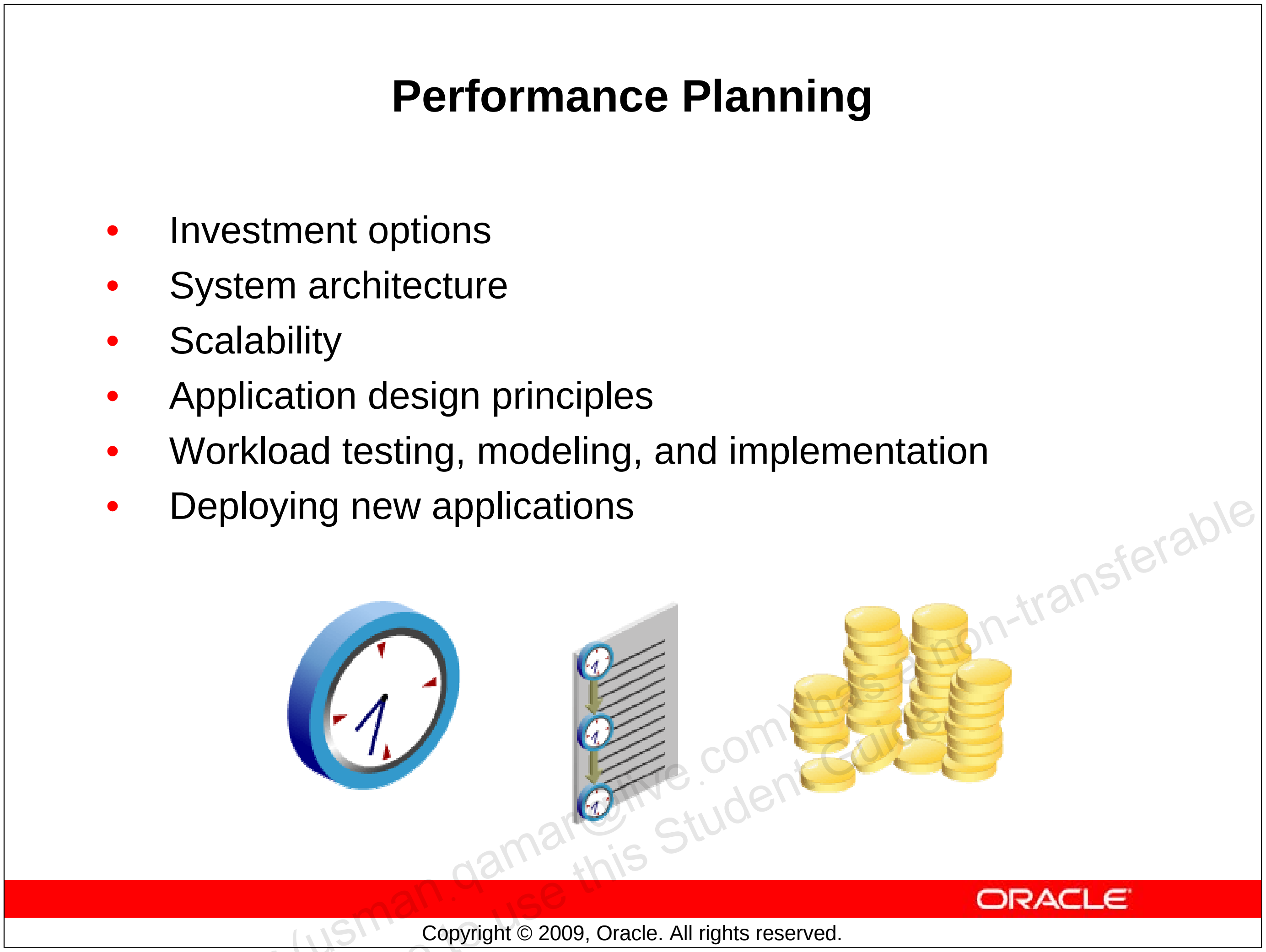
- Performance planning
- Instance tuning
- SQL tuning

Tuning Activities

The three facets of tuning involve performance planning, instance tuning, and SQL tuning.

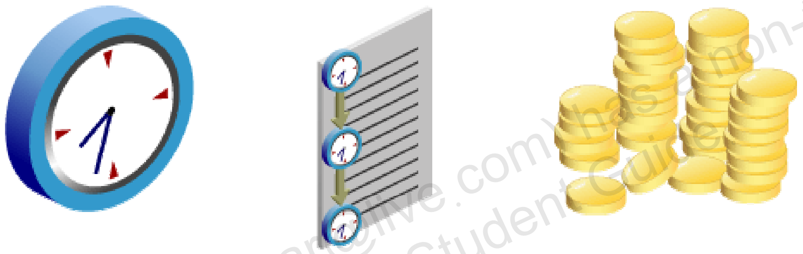
- Performance planning is the process of establishing the environment: the hardware, software, operating system, network infrastructure, and so on.
- Instance tuning is the actual adjustment of Oracle database parameters and operating system (OS) parameters to gain better performance of the Oracle database.
- SQL tuning involves making your application submit efficient SQL statements. SQL tuning is performed for the application as a whole, as well as for individual statements. At the application level, you want to be sure that different parts of the application are taking advantage of each other’s work and are not competing for resources unnecessarily. In this lesson, you learn about some common actions that you can take to tune specific SQL statements.

Note: For more information about performance tuning, refer to the *Oracle Database Performance Tuning Guide*.



Performance Planning

- Investment options
- System architecture
- Scalability
- Application design principles
- Workload testing, modeling, and implementation
- Deploying new applications



Performance Planning

There are many facets to performance planning. Planning must include a balance between performance (speed), cost, and reliability. You must consider the investment in your system architecture: the hardware and software infrastructure needed to meet your requirements. This, of course, requires analysis to determine the value for your given environment, application, and performance requirements. For example, the number of hard drives and controllers has an impact on the speed of data access.

The ability of an application to scale is also important. This means that you are able to handle more and more users, clients, sessions, or transactions, without incurring a huge impact on overall system performance. The most obvious violator of scalability is serializing operations among users. If all users go through a single path one at a time, then as more users are added, there are definitely adverse effects on performance. This is because more and more users line up to go through that path. Poorly written SQL also affects scalability. It requires many users to wait for inefficient SQL to complete; each user competing with the other on a large number of resources that they are not actually in need of.

The principles of application design can greatly affect performance. Simplicity of design, use of views and indexes, and data modeling are all very important.

Performance Planning (continued)

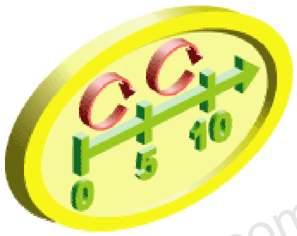
Any application must be tested under a representative production workload. This requires estimating database size and workload, and generating test data and system load.

Performance must be considered as new applications (or new versions of applications) are deployed. Sometimes design decisions are made to maintain compatibility with old systems during the rollout. A new database should be configured (on the basis of the production environment) specifically for the applications that it hosts.

A difficult and necessary task is testing the existing applications when changing the infrastructure. For example, upgrading the database to a newer version, or changing the operating system or server hardware. Before the application is deployed for production in the new configuration, you want to know the impact. The application will almost certainly require additional tuning. You need to know that the critical functionality will perform, without errors.

Instance Tuning

- Have well-defined goals.
- Allocate memory to database structures.
- Consider I/O requirements in each part of the database.
- Tune the operating system for optimal performance of the database.



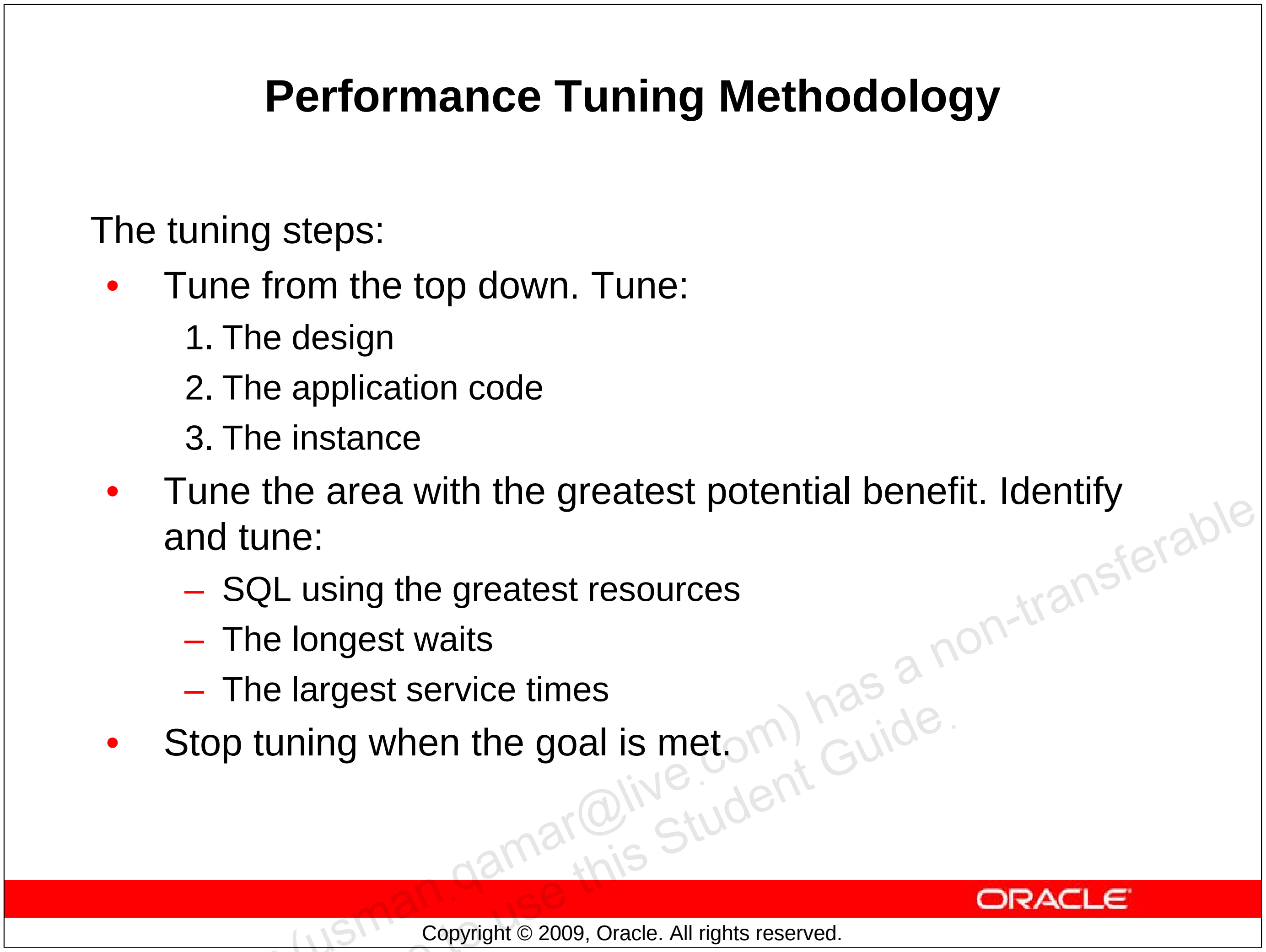
Instance Tuning

At the start of any tuning activity, it is necessary to have specific goals. A goal such as “Process 500 sales transactions per minute” is easier to work toward than one that says, “Make it go as fast as you can, and we’ll know when it’s good enough.”

You must allocate Oracle database memory suitably for your application to attain optimum performance. You have a finite amount of memory to work with. Too little memory allotted to certain parts of the Oracle database can cause inefficient background activity, which you may not even be aware of without doing some analysis.

Disk I/O is often the bottleneck of a database and, therefore, requires a lot of attention at the outset of any database implementation.

The operating system configuration can also affect the performance of an Oracle database. For more information, see the *Oracle Database Installation Guide* for your particular platform.



Performance Tuning Methodology

The tuning steps:

- Tune from the top down. Tune:
  1. The design
  2. The application code
  3. The instance
- Tune the area with the greatest potential benefit. Identify and tune:
  - SQL using the greatest resources
  - The longest waits
  - The largest service times
- Stop tuning when the goal is met.

Performance Tuning Methodology

Oracle has developed a tuning methodology based on years of experience. The basic steps are:

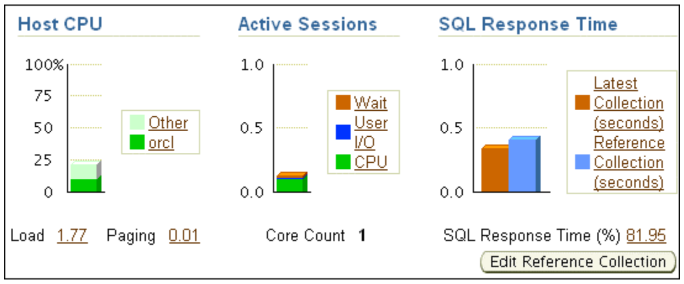
1. Check the OS statistics and general machine health before tuning the instance to be sure that the problem is in the database. Use the Enterprise Manager database home page.
2. Tune from the top down. Start with the design, then the application, and then the instance. For example, try to eliminate the full tables scans causing the I/O contention before tuning the tablespace layout on disk. This activity often requires access to the application code.
3. Tune the area with the greatest potential benefit. The tuning methodology presented in this course is simple. Identify the biggest bottleneck and tune it. Repeat this step. All the various tuning tools have some way to identify the SQL statements, resource contention, or services that are taking the most time. The Oracle database provides a time model and metrics to automate the process of identifying bottlenecks. The Advisors available in Oracle Database 11*g* use precisely this methodology.
4. Stop tuning when you meet your goal. This step implies that you set tuning goals.

This is a general approach to tuning the database instance and may require multiple passes.

Performance Monitoring

With Enterprise Manager:

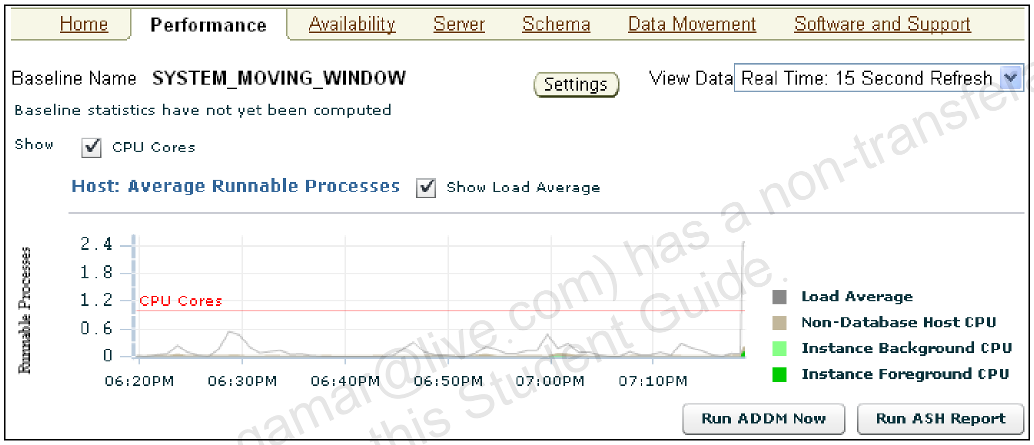
- Performance overview
- Graphs of metrics and details



Load 1.77 Paging 0.01

Core Count 1

SQL Response Time (%) 81.95



Baseline Name SYSTEM_MOVING_WINDOW

Baseline statistics have not yet been computed

View Data Real Time: 15 Second Refresh

Show ✓ CPU Cores

✓ Show Load Average

Performance Monitoring

You can respond to changes in performance only if you know the performance has changed. Oracle Database 11*g* provides several ways to monitor the current performance of the database instance. The database home page of Enterprise Manager (EM) provides a quick check of the health of the instance and the server, with graphs showing CPU usage, active sessions, and SQL response time. The home page also shows any alerts that have been triggered.

The Performance tab in EM shows several graphs of performance metrics from several directions. You can see performance in terms of CPU, average active sessions, throughput, I/O, and other dimensions. From the Performance page, you can follow links to detailed information: including sessions and individual SQL statements.

The information displayed in EM is based on performance views that exist in the database. You can access these views directly with SQL*Plus. Occasionally, you may need to access these views for some detail about the raw statistics.

Performance Tuning Data

Type of data gathered:

- Cumulative statistics:
  - Wait events with time information
  - Time model
- Metrics: Statistic rates
- Sampled statistics: Active session history
  - Statistics by session
  - Statistics by SQL
  - Statistics by service
  - Other dimensions

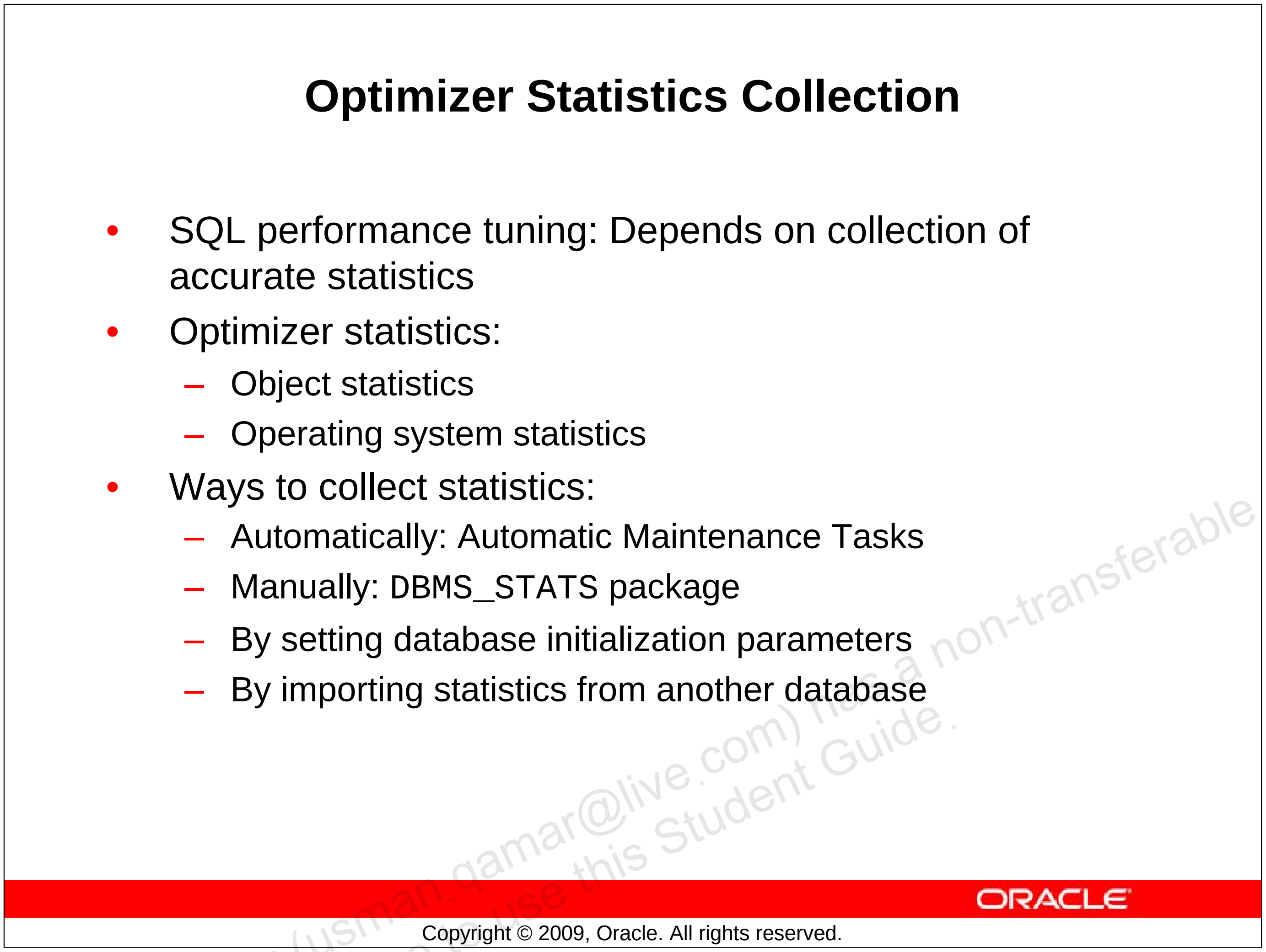
Performance Tuning Data

The Oracle database server software captures information about its own operation. Three major types of data are collected: cumulative statistics, metrics, and sampled statistics.

Cumulative statistics are counts and timing information of a variety of events that occur in the database server. Some are quite important, such as buffer busy waits. Others have little impact on tuning, such as index block split. The most important events for tuning are usually the ones showing the greatest cumulative time values. The statistics in Oracle Database 11*g* are correlated by the use of a time model. The time model statistics are based on a percentage of DB time, giving them a common basis for comparison.

Metrics are statistic counts per unit. The unit could be time (such as seconds), transaction, or session. Metrics provide a base to proactively monitor performance. You can set thresholds on a metric causing an alert to be generated. For example, you can set thresholds for when the reads per millisecond exceed a previously recorded peak value or when the archive log area is 95% full.

Sampled statistics are gathered automatically when `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`. Sampled statistics allow you to look back in time. You can view session and system statistics that were gathered in the past, in various dimensions, even if you had not thought of specifying data collection for these beforehand.



Optimizer Statistics Collection

- SQL performance tuning: Depends on collection of accurate statistics
- Optimizer statistics:
  – Object statistics
  – Operating system statistics
- Ways to collect statistics:
  – Automatically: Automatic Maintenance Tasks
  – Manually: `DBMS_STATS` package
  – By setting database initialization parameters
  – By importing statistics from another database

Optimizer Statistics Collection

Optimizer statistics are collections of data that are specific details about database objects. These statistics are essential for the query optimizer to choose the best execution plan for each SQL statement. These statistics are gathered periodically and do not change between gatherings.

The recommended approach to gathering optimizer statistics is to allow the Oracle database to automatically gather the statistics. The Automatic Maintenance Tasks can be created automatically at database creation time and is managed by the Scheduler. It gathers statistics on all objects in the database that have either missing or stale optimizer statistics by default. You can change the default configuration through the Automatic Maintenance Tasks page.

System statistics describe the system's hardware characteristics, such as I/O and CPU performance and utilization, to the query optimizer. When choosing an execution plan, the optimizer estimates the I/O and CPU resources required for each query. System statistics enable the query optimizer to more accurately estimate I/O and CPU costs, and thereby choose a better execution plan. System statistics are collected using the `DBMS_STATS.GATHER_SYSTEM_STATS` procedure. When the Oracle database gathers system statistics, it analyzes system activity in a specified period of time. System statistics are not automatically gathered. Oracle Corporation recommends that you use the `DBMS_STATS` package to gather system statistics.

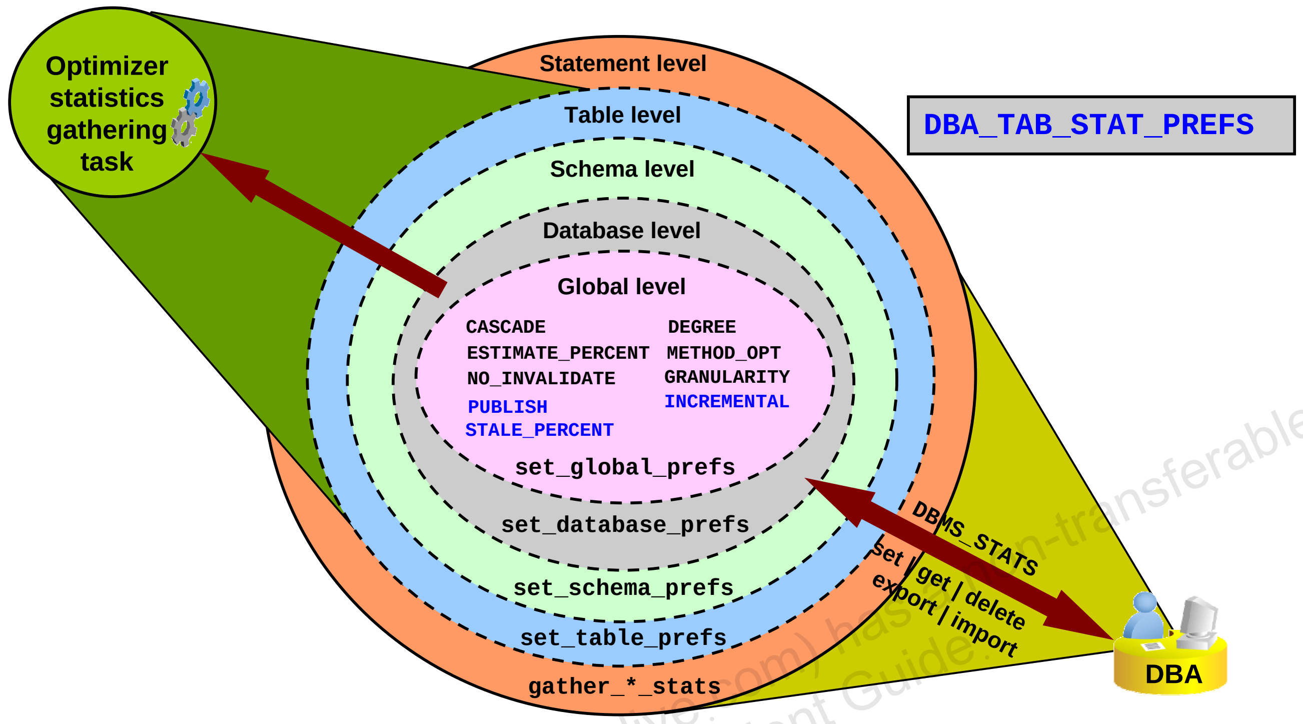
Optimizer Statistics Collection (continued)

If you choose not to use automatic statistics gathering, then you must manually collect statistics in all schemas, including system schemas. If the data in your database changes regularly, you also need to gather statistics regularly to ensure that the statistics accurately represent characteristics of your database objects. To manually collect statistics, use the `DBMS_STATS` package. This PL/SQL package is also used to modify, view, export, import, and delete statistics.

You can also manage optimizer and system statistics collection through database initialization parameters. For example:

- The `OPTIMIZER_DYNAMIC_SAMPLING` parameter controls the level of dynamic sampling performed by the optimizer. You can use dynamic sampling to estimate statistics for tables and relevant indexes when they are not available or are too out of date to trust. Dynamic sampling also estimates single-table predicate selectivity when collected statistics cannot be used or are likely to lead to significant errors in estimation.
- The `STATISTICS_LEVEL` parameter controls all major statistics collections or advisories in the database and sets the statistics collection level for the database. The values for this parameter are `BASIC`, `TYPICAL`, and `ALL`. You can query the `V$STATISTICS_LEVEL` view to determine which parameters are affected by the `STATISTICAL_LEVEL` parameter.
  Note: Setting `STATISTICS_LEVEL` to `BASIC` disables many automatic features and is not recommended.

Statistic Preferences: Overview



Statistic Preferences: Overview

The automated statistics-gathering feature was introduced in Oracle Database 10*g*, Release 1 to reduce the burden of maintaining optimizer statistics. However, there were cases where you had to disable it and run your own scripts instead. One reason was the lack of object-level control. Whenever you found a small subset of objects for which the default gather statistics options did not work well, you had to lock the statistics and analyze them separately by using your own options. For example, the feature that automatically tries to determine adequate sample size (`ESTIMATE_PERCENT=AUTO_SAMPLE_SIZE`) does not work well against columns that contain data with very high frequency skews. The only way to get around this issue was to manually specify the sample size in your own script.

Note: You can describe all the effective statistics preference settings for all relevant tables by using the `DBA_TAB_STAT_PREFS` view.

Using Statistic Preferences

- PUBLISH: Used to decide whether to publish the statistics to the dictionary or to store them in a pending area before
- STALE_PERCENT: Used to determine the threshold level at which an object is considered to have stale statistics. The value is a percentage of rows modified since the last statistics gathering.
- INCREMENTAL: Used to gather global statistics on partitioned tables in an incremental way

```
exec dbms_stats.set_table_prefs('SH','SALES','STALE_PERCENT','13');
```

Using Statistic Preferences

The Statistic Preferences feature in Oracle Database 11g introduces flexibility so that you can rely more on the automated statistics-gathering feature to maintain the optimizer statistics when some objects require settings that are different from the database default.

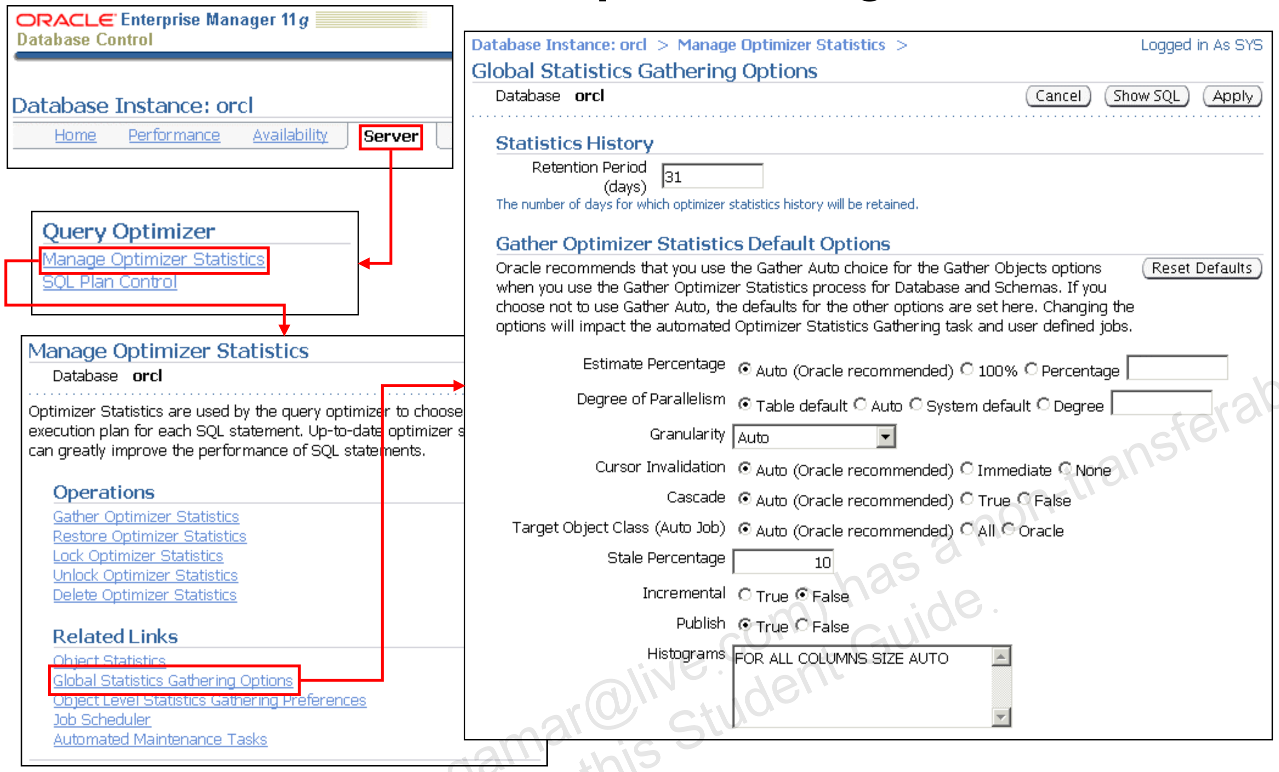
This feature allows you to associate the statistics-gathering options that override the default behavior of the GATHER_*_STATS procedures and the automated Optimizer Statistics Gathering task at the object or schema level. You can use the DBMS_STATS package to manage the gathering statistics options.

You can set, get, delete, export, and import those preferences at the table, schema, database, and global levels. Global preferences are used for tables that do not have preferences, whereas database preferences are used to set preferences on all tables.

The following options are new in Oracle Database 11g, Release 1:

- PUBLISH is used to decide whether to publish the statistics to the dictionary or to store them in a pending area before.
- STALE_PERCENT is used to determine the threshold level at which an object is considered to have stale statistics. The value is a percentage of rows modified since the last statistics gathering. The example changes the 10 percent default to 13 percent for SH.SALES only.
- INCREMENTAL is used to gather global statistics on partitioned tables in an incremental way.

Setting Global Preferences with Enterprise Manager



Manage Optimizer Statistics

Database orcl

Optimizer Statistics are used by the query optimizer to choose execution plan for each SQL statement. Up-to-date optimizer s can greatly improve the performance of SQL statements.

Operations
- Gather Optimizer Statistics
- Restore Optimizer Statistics
- Lock Optimizer Statistics
- Unlock Optimizer Statistics
- Delete Optimizer Statistics

Related Links
- Object Statistics
- Global Statistics Gathering Options
- Object Level Statistics Gathering Preferences
- Job Scheduler
- Automated Maintenance Tasks

Global Statistics Gathering Options

Database orcl

Statistics History

Retention Period (days) 31

The number of days for which optimizer statistics history will be retained.

Gather Optimizer Statistics Default Options

Oracle recommends that you use the Gather Auto choice for the Gather Objects options when you use the Gather Optimizer Statistics process for Database and Schemas. If you choose not to use Gather Auto, the defaults for the other options are set here. Changing the options will impact the automated Optimizer Statistics Gathering task and user defined jobs.

Estimate Percentage	[✓ Auto (Oracle recommended)] 100% Percentage
Degree of Parallelism	[✓ Table default] Auto System default Degree
Granularity	Auto
Cursor Invalidation	[✓ Auto (Oracle recommended)] Immediate None
Cascade	[✓ Auto (Oracle recommended)] True False
Target Object Class (Auto Job)	[✓ Auto (Oracle recommended)] All Oracle
Stale Percentage	10
Incremental	True [✓ False]
Publish	[✓ True] False
Histograms	FOR ALL COLUMNS SIZE AUTO

Setting Global Preferences with Enterprise Manager

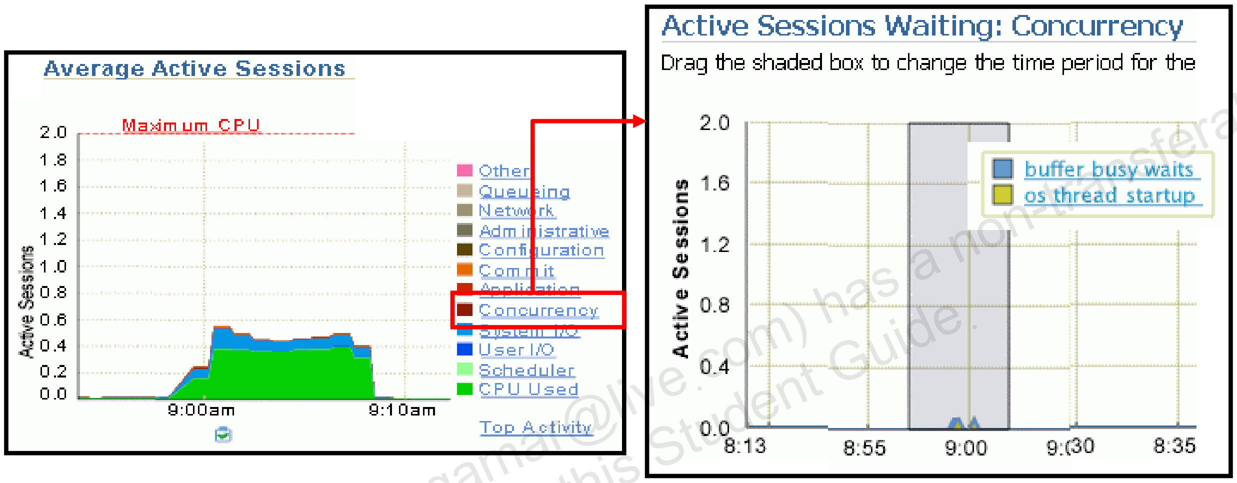
It is possible to control global preference settings by using Enterprise Manager. You do so on the Manage Optimizer Statistics page, which you access from the Database home page by clicking the Server tab, then the Manage Optimizer Statistics link, and then the Global Statistics Gathering Options link.

On the Global Statistics Gathering Options page, change the global preferences in the Gather Optimizer Statistics Default Options section. When finished, click the Apply button.

Note: To change the statistics gathering options at the object level or schema level, click the Object Level Statistics Gathering Preferences link on the Manage Optimizer Statistics page.

Oracle Wait Events

- A collection of wait events provides information about the sessions or processes that had to wait or must wait for different reasons.
- These events are listed in the V$EVENT_NAME view.



Oracle Wait Events

Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention. Remember that these are only symptoms of problems, not the actual causes.

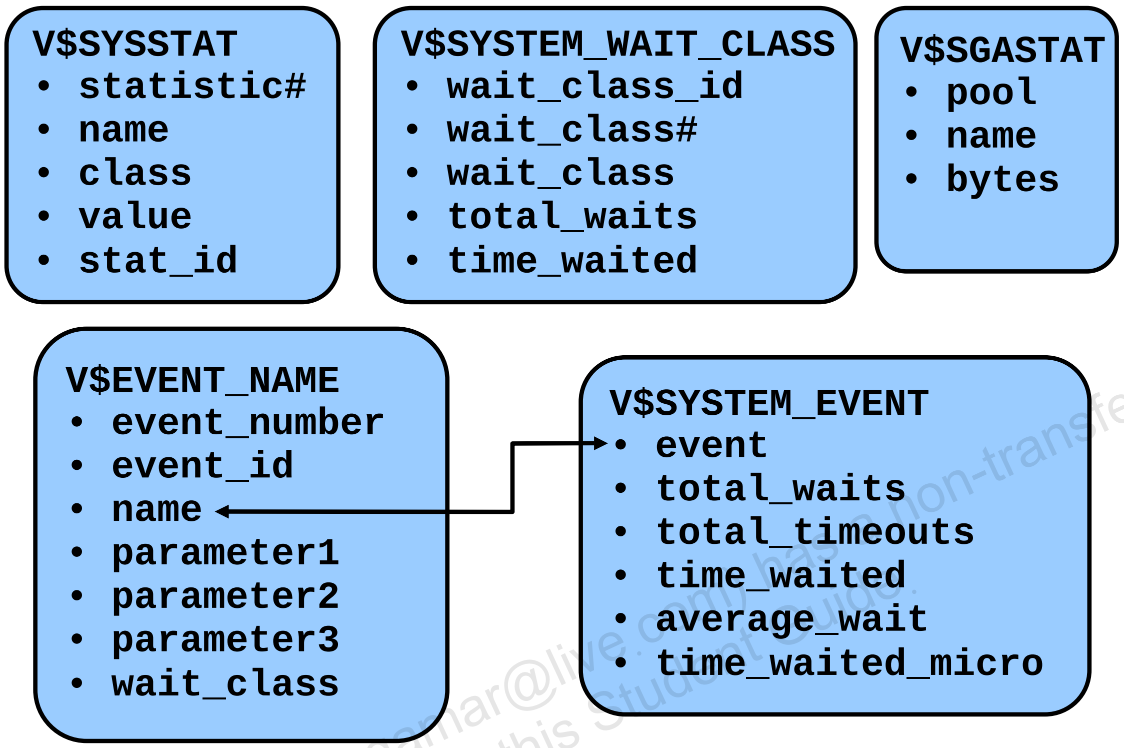
Wait events are grouped into classes. The wait event classes include: Administrative, Application, Cluster, Commit, Concurrency, Configuration, Idle, Network, Other, Scheduler, System I/O, and User I/O.

There are more than 800 wait events in the Oracle database, including `free buffer wait`, `latch free`, `buffer busy waits`, `db file sequential read`, and `db file scattered read`.

Using EM, you can view wait events by opening the Performance page and viewing the “Average Active Sessions” graph, as shown in the slide. By clicking the link for a particular wait event class, you can drill down to the specific wait events by using the Top Activity interface. In this example, there was a very small set of `buffer busy waits`.

For a list of the most common Oracle events, refer to the *Oracle Database Reference 11g* documentation.

Instance Statistics



Instance Statistics

To effectively diagnose performance problems, statistics must be available. The Oracle database instance generates many types of cumulative statistics for the system, sessions, and individual SQL statements at the instance level. The Oracle database also tracks cumulative statistics on segments and services. When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in.

Note: Instance statistics are dynamic and are reset at every instance startup. These statistics can be captured at a point in time and held in the database in the form of snapshots.

Wait Events Statistics

All the possible wait events are cataloged in the `V$EVENT_NAME` view.

Cumulative statistics for all sessions are stored in `V$SYSTEM_EVENT`, which shows the total waits for a particular event since instance startup.

When you are troubleshooting, you need to know whether a process has waited for any resource.

Systemwide Statistics

All the systemwide statistics are cataloged in the `V$STATNAME` view: Over 400 statistics are available in Oracle Database 11*g*.

The server displays all calculated system statistics in the `V$SYSSTAT` view. You can query this view to find cumulative totals since the instance started.

Instance Statistics (continued)

Example

```
SQL>  SELECT name, class, value FROM v$sysstat;
NAME                               CLASS      VALUE
-------------------------------  ------ ----------
...
table scans (short tables)            64     135116
table scans (long tables)             64        250
table scans (rowid ranges)            64          0
table scans (cache partitions)        64          3
table scans (direct read)             64          0
table scan rows gotten                64   14789836
table scan blocks gotten              64     558542
...
```

Systemwide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on. Each of the system statistics can belong to more than one class, so you cannot do a simple join on `V$SYSSTATS.CLASS` and `V$SYSTEM_WAIT_CLASS.WAIT_CLASS#`.

You can also view all wait events for a particular wait class by querying `V$SYSTEM_WAIT_CLASS`, as in this example (with formatting applied):

```
SQL> SELECT * FROM V$SYSTEM_WAIT_CLASS
  2  WHERE wait_class LIKE '%I/O%';

CLASS_ID   CLASS# WAIT_CLASS   TOTAL_WAITS TIME_WAITED
---------- ------------------- ----------- -----------
1740759767     8 User I/O          1119152       39038
4108307767     9 System I/O         296959       27929
```

SGA Global Statistics

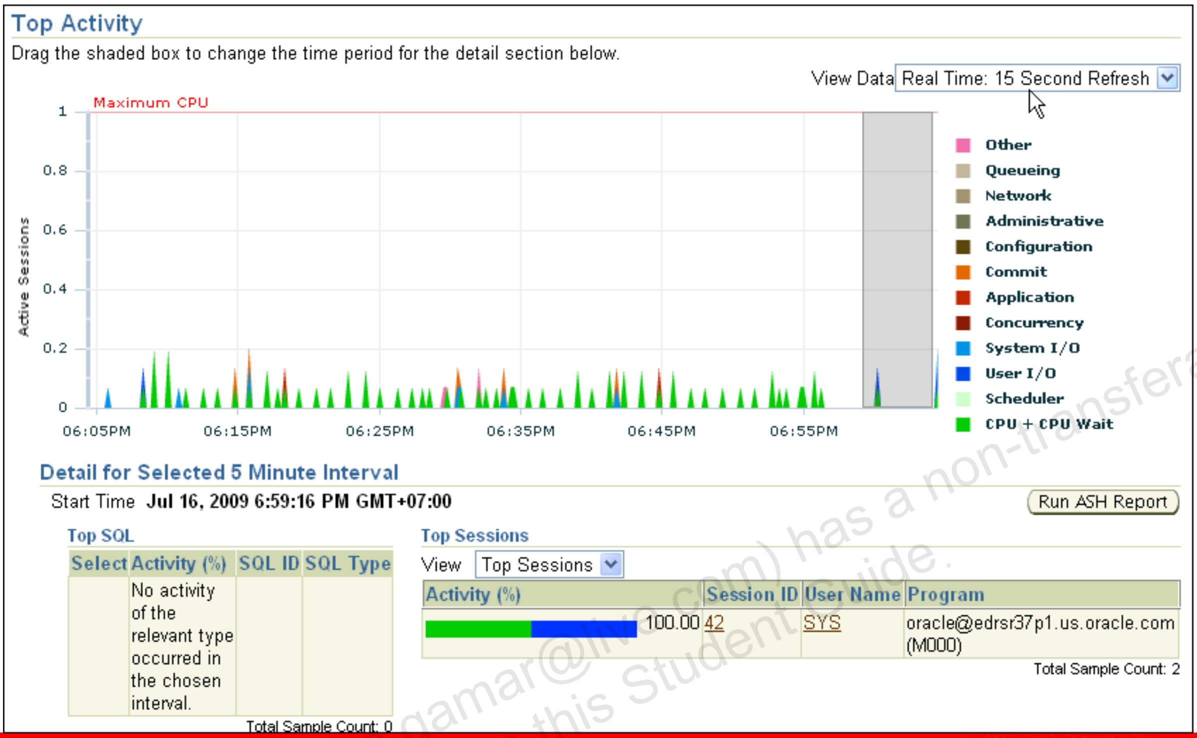
The server displays all calculated memory statistics in the `V$SGASTAT` view. You can query this view to find cumulative totals of detailed SGA usage since the instance started, as in the following example:

```
SQL>  SELECT * FROM v$sgastat;
POOL            NAME                            BYTES
------          -------------------------  ----------
                fixed_sga                     7780360
                buffer_cache                 25165824
                log_buffer                     262144
shared pool     sessions                      1284644
shared pool     sql area                     22376876
...
```

The results shown are only a partial display of the output.

When the `STATISTICS_LEVEL` parameter is set to `BASIC`, the value of the `TIMED_STATISTICS` parameter defaults to `FALSE`. Timing information is not collected for wait events and much of the performance-monitoring capability of the database is disabled. The explicit setting of `TIMED_STATISTICS` overrides the value derived from `STATISTICS_LEVEL`.

Monitoring Session Performance



Detail for Selected 5 Minute Interval

Start Time **Jul 16, 2009 6:59:16 PM GMT+07:00**

Top SQL

Select	Activity (%)	SQL ID	SQL Type
	No activity of the relevant type occurred in the chosen interval.		

Total Sample Count: 0

Top Sessions

Activity (%)	Session ID	User Name	Program
100.00	42	SYS	oracle@edrsr37p1.us.oracle.com (M000)

Total Sample Count: 2

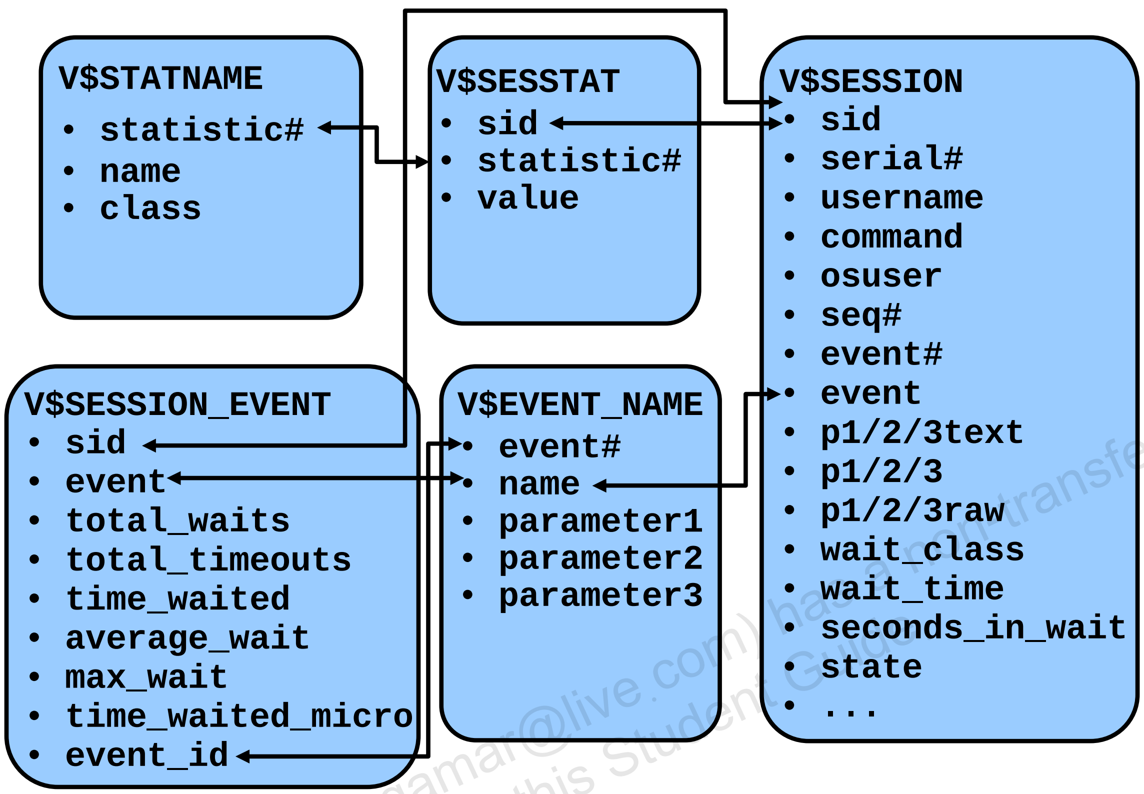
ORACLE

Monitoring Session Performance

Enterprise Manager provides session detail pages so that you can view the wait events occurring in individual sessions. On the Performance tabbed page, click Top Activity to view the summary of all sessions. In the lower right corner of the Top Activity page is a listing of the Top Sessions. Click the session identifier to view the Session Details page.

The Top Activity page and Session Details page are based on performance view in the database.

Displaying Session-Related Statistics



Displaying Session-Related Statistics

You can display current session information for each user logged on by querying V$SESSION. For example, you can use V$SESSION to determine whether a session represents a user session, or was created by a database server process (BACKGROUND).

You can query either V$SESSION or V$SESSION_WAIT to determine the resources or events for which active sessions are waiting.

You can view user session statistics in V$SESSTAT. The V$SESSION_EVENT view lists information about waits for an event by a session.

Cumulative values for instance statistics are generally available through dynamic performance views, such as V$SESSTAT and V$SYSSTAT. Note that the cumulative values in dynamic views are reset when the database instance is shut down.

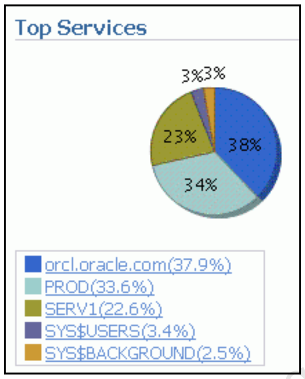
The V$MYSTAT view displays the statistics of the current session.

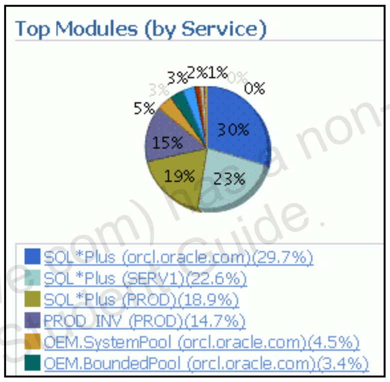
You can also query V$SESSMETRIC to display the performance metric values for all active sessions. This view lists performance metrics such as CPU usage, number of physical reads, number of hard parses, and the logical read ratio.

Displaying Service-Related Statistics

For *n*-tier environments, because session statistics are not as helpful, you can see service-level statistics in these views:

- **V$SERVICE_EVENT:** Aggregated wait counts and wait times for each service, on a per event basis
- **V$SERVICE_WAIT_CLASS:** Aggregated wait counts and wait times for each service on a wait class basis





Displaying Service-Related Statistics

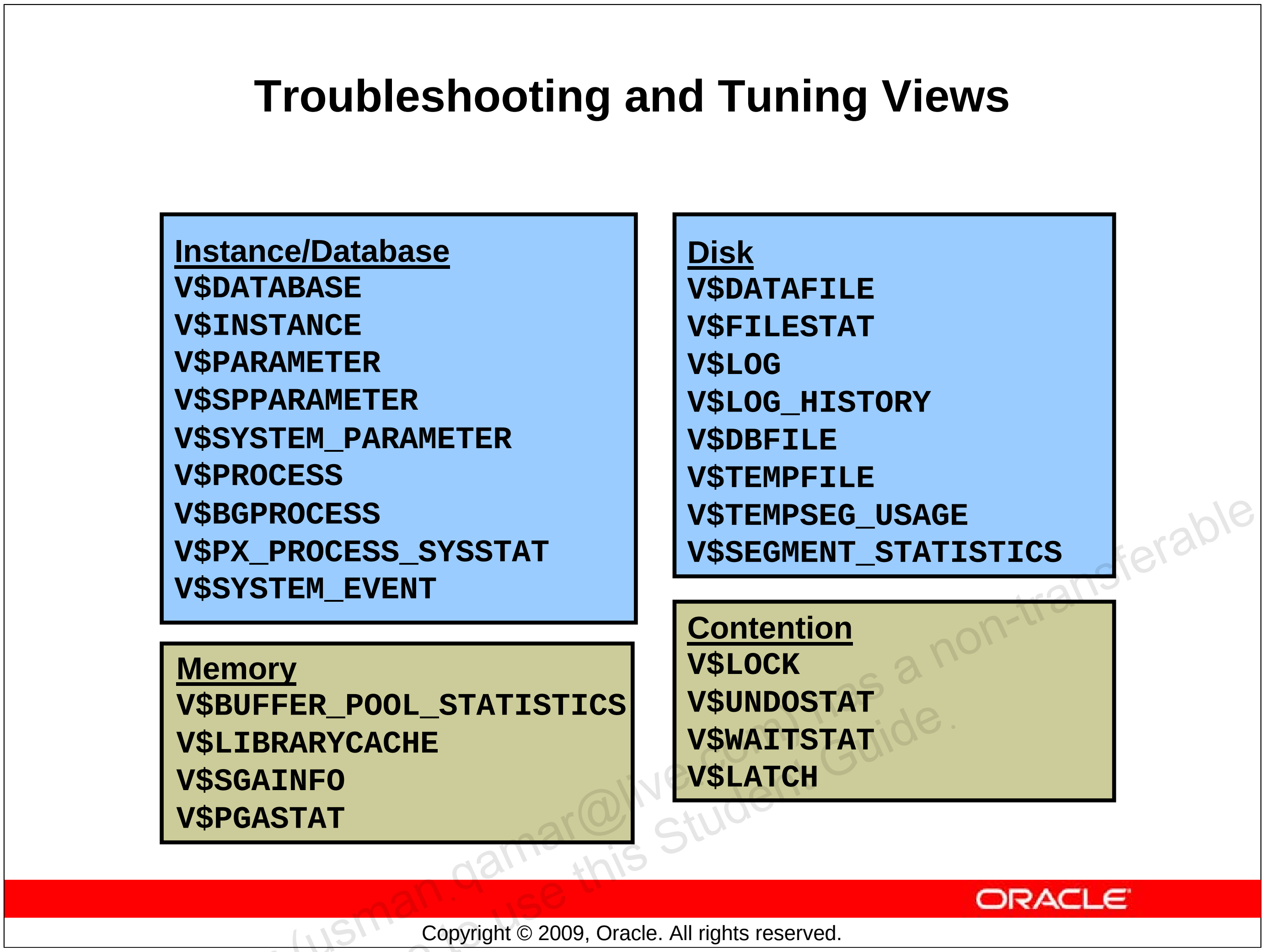
In an *n*-tier environment where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. These two views provide the same information that their like-named session counterparts provide, except that the information is presented at the service level rather than at the session level.

`V$SERVICE_WAIT_CLASS` shows wait statistics for each service, broken down by wait class.

`V$SERVICE_EVENT` shows the same information as `V$SERVICE_WAIT_CLASS`, except that it is further broken down by event ID.

Enterprise Manager also provides aggregation by service and by module and service. You can click the legend in each of the views, to view the activity and statistics for each service.

You can define a service in the database by using the `DBMS_SERVICE` package and use the net service name to assign applications to a service.



Troubleshooting and Tuning Views

Instance/Database
V$DATABASE
V$INSTANCE
V$PARAMETER
V$SPPARAMETER
V$SYSTEM_PARAMETER
V$PROCESS
V$BGPROCESS
V$PX_PROCESS_SYSSTAT
V$SYSTEM_EVENT

Disk
V$DATAFILE
V$FILESTAT
V$LOG
V$LOG_HISTORY
V$DBFILE
V$TEMPFILE
V$TEMPSEG_USAGE
V$SEGMENT_STATISTICS

Memory
V$BUFFER_POOL_STATISTICS
V$LIBRARYCACHE
V$SGAINFO
V$PGASTAT

Contention
V$LOCK
V$UNDOSTAT
V$WAITSTAT
V$LATCH

Troubleshooting and Tuning Views

The slide lists some of the views you may need to access to determine the cause of performance problems or analyze the current status of your database. Many of these views show the same data that is used in creating the reports that the Enterprise Manager produces from the Automatic Workload Repository (AWR). In some cases, the raw data is needed to fully diagnose a problem.

For a complete description of these views, refer to the *Oracle Database Reference Manual*.

Dictionary Views

- The following dictionary and special views display object statistics after use of the `DBMS_STATS` package:
  - `DBA_TABLES, DBA_TAB_COLUMNS`
  - `DBA_CLUSTERS`
  - `DBA_INDEXES`
  - `DBA_TAB_HISTOGRAMS`
- This statistical information is static until you reexecute the appropriate procedures in `DBMS_STATS`.

Dictionary Views

When you need to look at the optimizer statistics of specific database objects in detail, use the `DBMS_STATS` package, which collects statistics and populates columns in some `DBA_xxx` views.

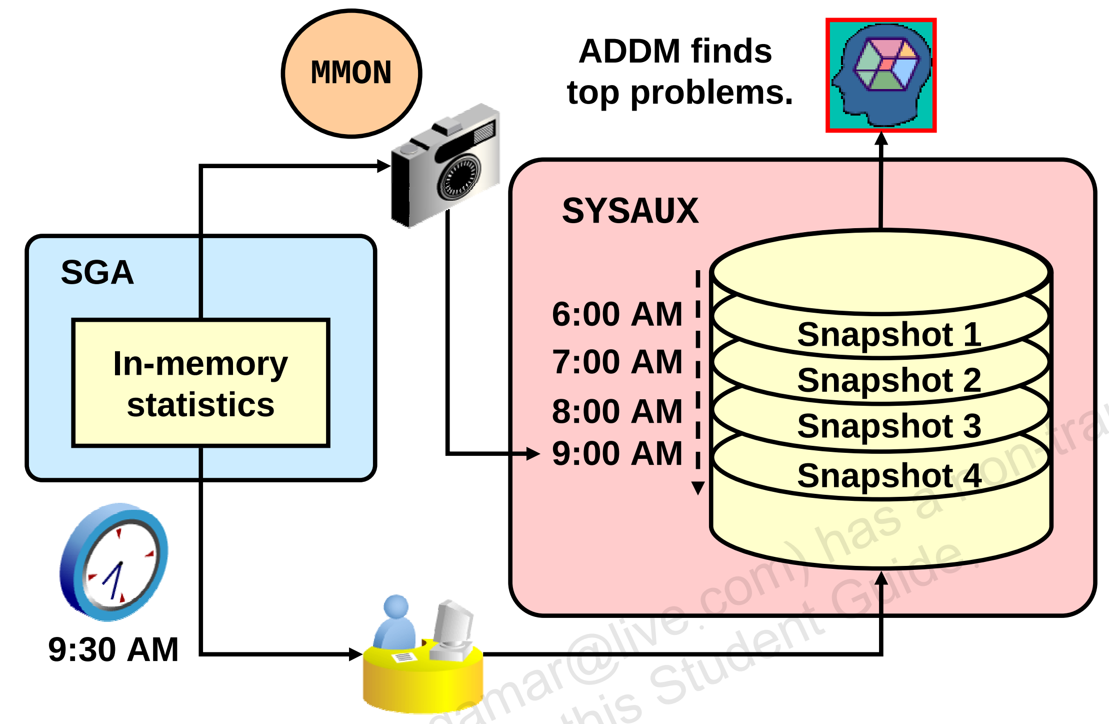
`DBMS_STATS` populates columns in the views concerned with:

- Table data storage within extents and blocks:
  - `DBA_TABLES`
  - `DBA_TAB_COLUMNS`
- Cluster data storage within extents and blocks:
  - `DBA_CLUSTERS`
- Index data storage within extents and blocks, and index usefulness:
  - `DBA_INDEXES`
- Nonindexed and indexed columns data distribution:
  - `DBA_TAB_HISTOGRAMS`

For more information about using the `DBMS_STATS` package, refer to the *Oracle Database Performance Tuning Guide*.

Performing an `ANALYZE INDEX … VALIDATE STRUCTURE` command populates the `INDEX_STATS` and `INDEX_HISTOGRAM` views that contain statistics for indexes.

Automatic Workload Repository



Automatic Workload Repository

The Automatic Workload Repository (AWR) is a collection of persistent system performance statistics owned by SYS. The AWR resides in the SYSAUX tablespace.

A *snapshot* is a set of performance statistics captured at a certain time and stored in the AWR. Each snapshot is identified by a snapshot sequence number (`snap_id`) that is unique in the AWR. By default, snapshots are generated every 60 minutes. You can adjust this frequency by changing the snapshot `INTERVAL` parameter. Because the database advisors rely on these snapshots, be aware that adjustment of the interval setting can affect diagnostic precision. For example, if the `INTERVAL` is set to 4 hours, you may miss transient events that would be noticeable in 60-minute intervals.

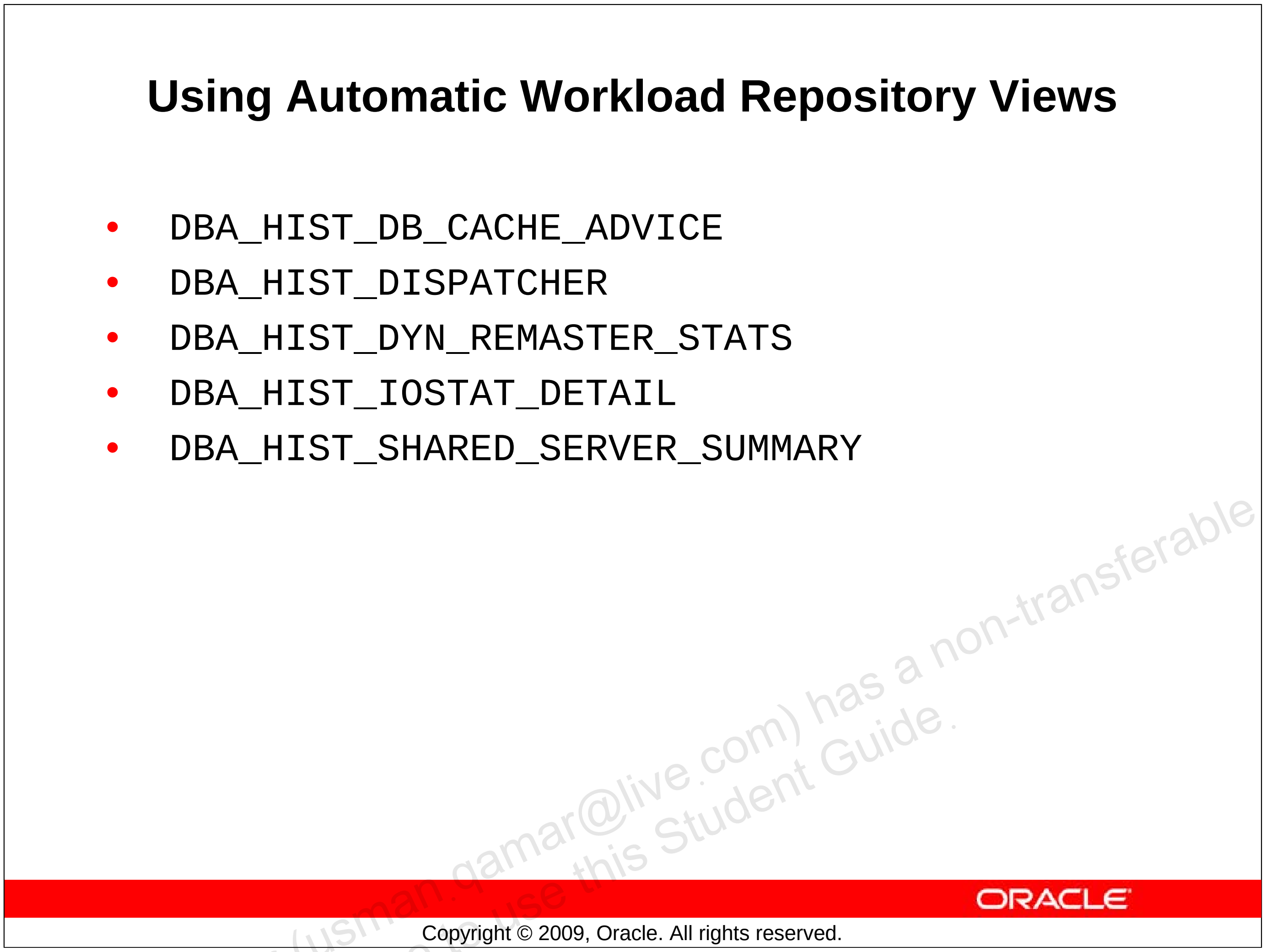
You can use the `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS` stored procedure or Database Control to change the settings that control snapshot collection. In Database Control, click Automatic Workload Repository in the Statistics Management region of the Server tabbed page. Then click Edit to make the changes. The stored procedure offers more flexibility in defining `INTERVAL` values than does Database Control.

You can take manual snapshots by using Database Control or the `DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT` stored procedure. Taking manual snapshots is supported in conjunction with the automatic snapshots that the system generates. Manual snapshots are expected to be used when you want to capture the system behavior at two specific points in time that do not coincide with the automatic schedule.

Automatic Workload Repository (continued)

Statspack is a bundled utility that provides a subset of the collection and reporting capability of the AWR. However, there is no supported path to migrate Statspack data into the workload repository. Also, the workload repository is not compatible with the Statspack schema. Statspack is not accessible through Enterprise Manager; it requires setup, and does not have automatic retention settings, or automatic purge. The Statspack utility does provide scripts for setup, automatic snapshot collection, and reporting. Statspack snapshots can be marked for retention, as part of a Statspack baseline, or purged with provided scripts.

Statspack is documented in the `$ORACLE_HOME/rdbms/admin/spdoc.txt` file.



Using Automatic Workload Repository Views

- DBA_HIST_DB_CACHE_ADVICE
- DBA_HIST_DISPATCHER
- DBA_HIST_DYN_REMASTER_STATS
- DBA_HIST_IOSTAT_DETAIL
- DBA_HIST_SHARED_SERVER_SUMMARY

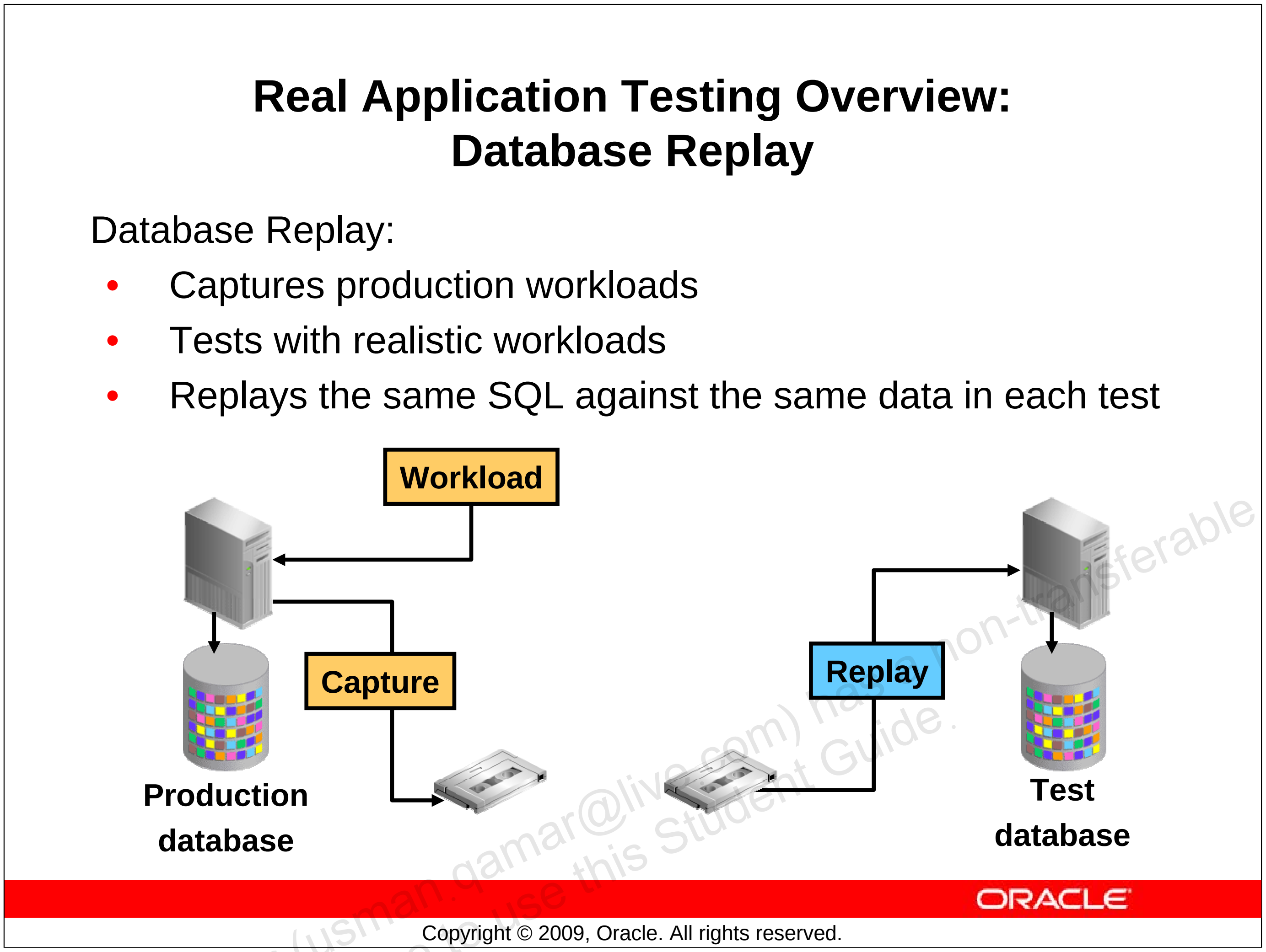
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using New Automatic Workload Repository Views

The following AWR statistics views are available in Oracle Database 11*g* R2:

- `DBA_HIST_DB_CACHE_ADVICE`: Displays historical predictions of the number of physical reads for the cache size corresponding to each row
- `DBA_HIST_DISPATCHER`: Displays historical information for each dispatcher process at the time of the snapshot
- `DBA_HIST_DYN_REMASTER_STATS`: Displays statistical information about the dynamic remastering process
- `DBA_HIST_IOSTAT_DETAIL`: Displays historical I/O statistics aggregated by file type and function
- `DBA_HIST_SHARED_SERVER_SUMMARY`: Displays historical information for shared servers, such as shared server activity, common queues, and dispatcher queues



Real Application Testing Overview: Database Replay

Database Replay:

- Captures production workloads
- Tests with realistic workloads
- Replays the same SQL against the same data in each test

Workload
Capture
Replay
Production database
Test database

Real Application Testing Overview: Database Replay

The system is going to change. The hardware must be upgraded, or the operating system, or the database version, or all of them. You have to assure the users that the application will continue to function after the upgrade, and often guarantee performance at least as good as it was before the upgrade.

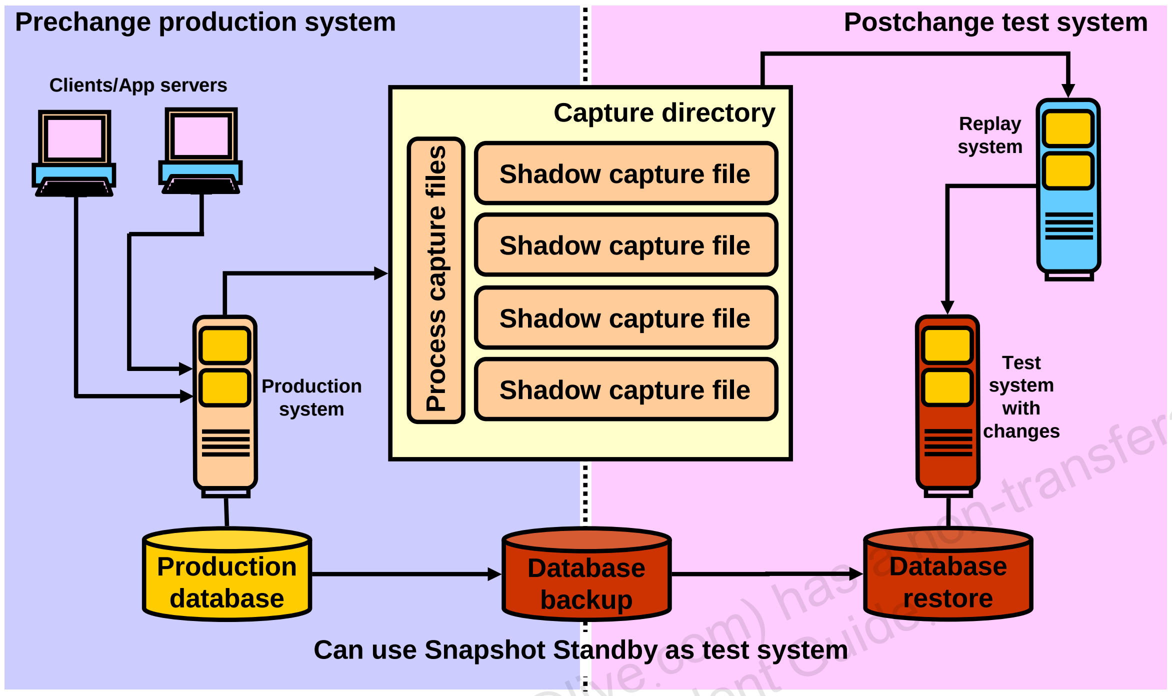
You build a test environment and test the application with a workload. You make changes and test again until you are confident that the application is working in the new environment.

This whole process depends on the test workload being representative of the actual production workload. A single statement in the production workload missing in the test workload could cause the upgrade to fail in production.

Database Replay captures the actual production workload and allows you to replay the workload in a test environment. The replay can be modified to allow for higher throughput, asynchronous application, and faster or slower replay. Database Replay can also be used to fine-tune by running exactly the same set of statements repeatedly. You can make changes to the environment and observe the differences. This capability also allows capture and replay of test cases to resolve errors.

Using actual production workloads allows you to be confident that the testing process is complete and accurate.

The Big Picture



The Big Picture

The significant benefit with Oracle Database 11*g* managing system changes is the added confidence to the business in the success of performing the change. The record and replay functionality offers confidence in the ease of upgrade during a database server upgrade. A useful application of Database Replay is to test the performance of a new server configuration. Suppose you are utilizing a single-instance database and want to move to a Real Application Clusters (RAC) setup. You can record the workload of an interesting period and then set up a RAC test system for replay. During replay, you can monitor the performance benefit of the new configuration by comparing the performance to the recorded system.

Database Replay can be used for debugging. You can record and replay sessions emulating an environment to make bugs more reproducible. Manageability feature testing is another benefit. Self-managing and self-healing systems need to implement this advice automatically ("autonomic computing model"). Multiple replay iterations allow testing and fine-tuning of the control strategies' effectiveness and stability.

To learn more about this technology, attend the *Oracle Database 11g: Performance* course.

For more information, see also the *Oracle Database Performance Tuning Guide* and the *Oracle Database PL/SQL Packages and Types Reference*.

Quiz

Select the statements that are true about statistics collection:

1. You can manually collect statistics with the `DBMS_STATS` package.
2. You can automatically collect statistics by enabling Automatic Maintenance Tasks.
3. You can import statistics from another database.
4. You can collect statistics by setting database initialization parameters.
5. You can collect statistics by manually updating the data dictionary.

Answer: 1, 2, 3, 4

Summary

In this lesson, you should have learned how to:

- Monitor the performance of sessions and services
- Describe the benefits of Database Replay

Practice 14 Overview: Monitoring Instance Performance

This practice covers the following topics:

- Monitoring Top Services and Sessions