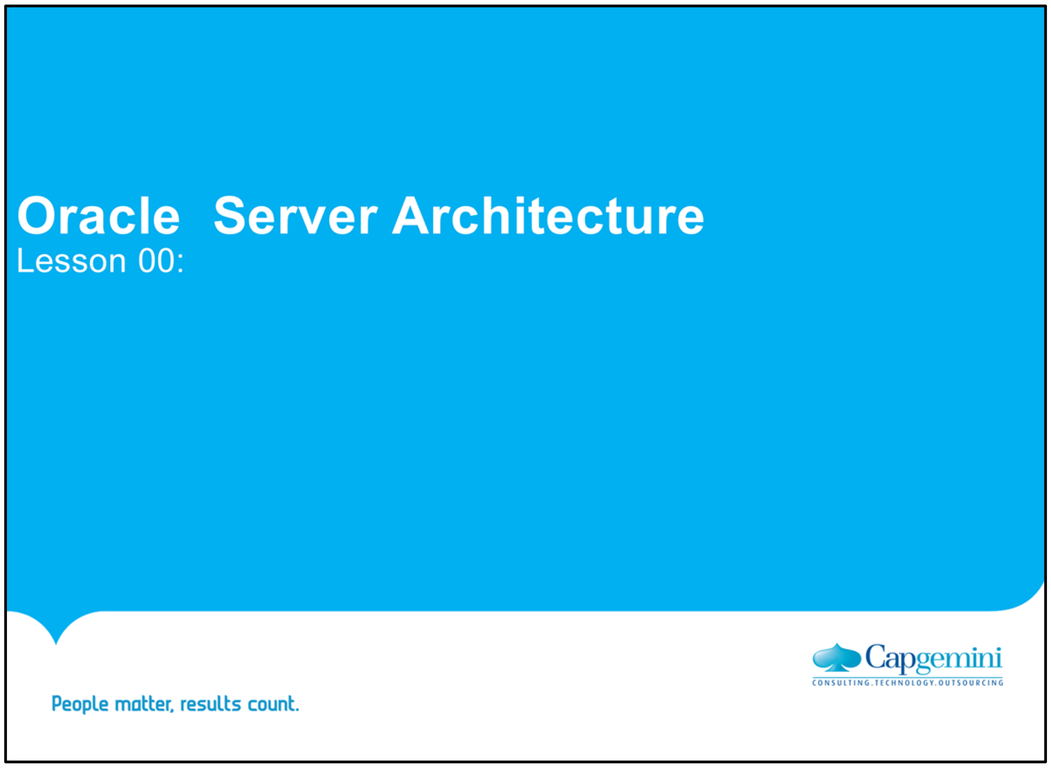
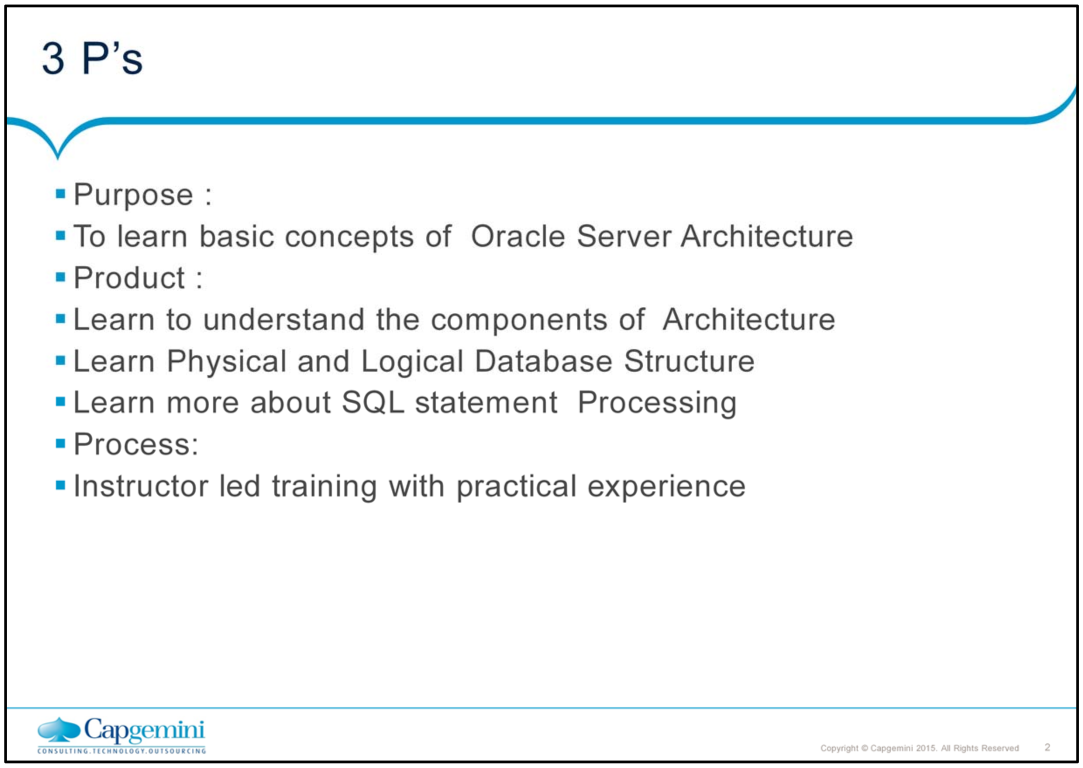


Oracle Server Architecture

Lesson 00:

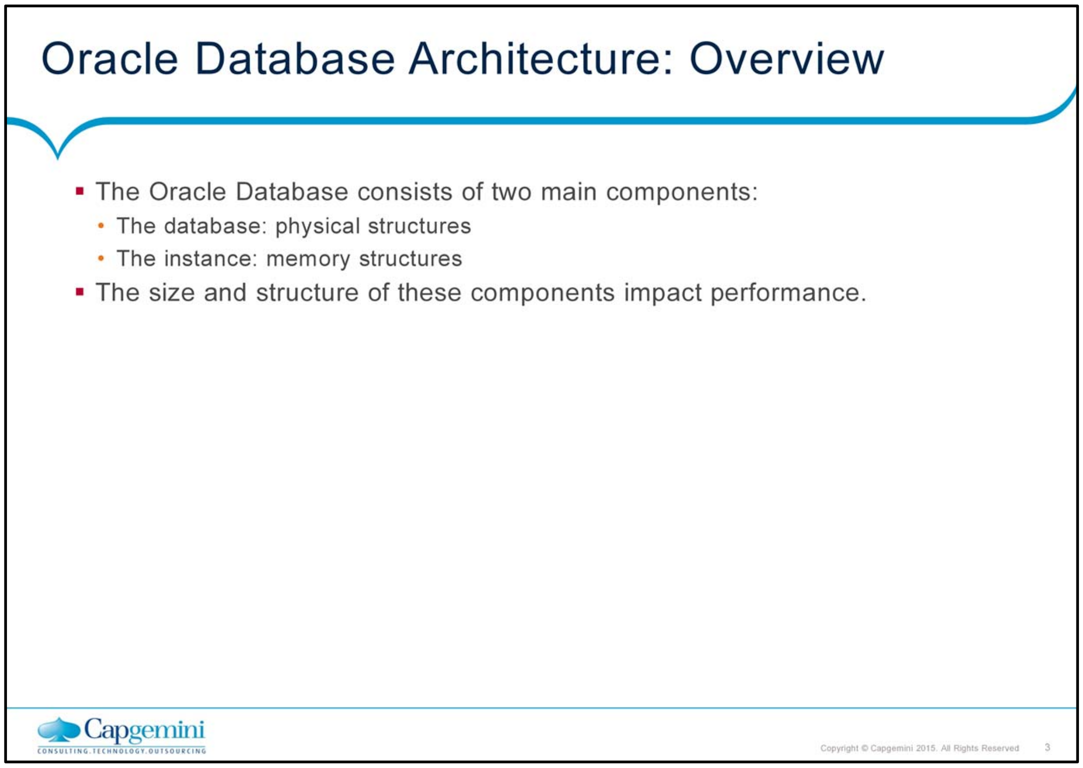
People matter, results count.



3 P's

- Purpose :
- To learn basic concepts of Oracle Server Architecture
- Product :
- Learn to understand the components of Architecture
- Learn Physical and Logical Database Structure
- Learn more about SQL statement Processing
- Process:
- Instructor led training with practical experience

2



Oracle Database Architecture: Overview

- The Oracle Database consists of two main components:
  - The database: physical structures
  - The instance: memory structures
- The size and structure of these components impact performance.

3

Oracle Database Architecture: Overview

An Oracle server consists of an Oracle database and an Oracle instance.

The database consists of physical files such as:

The control file where the database configuration is stored

The redo log files, which have information required for database recovery

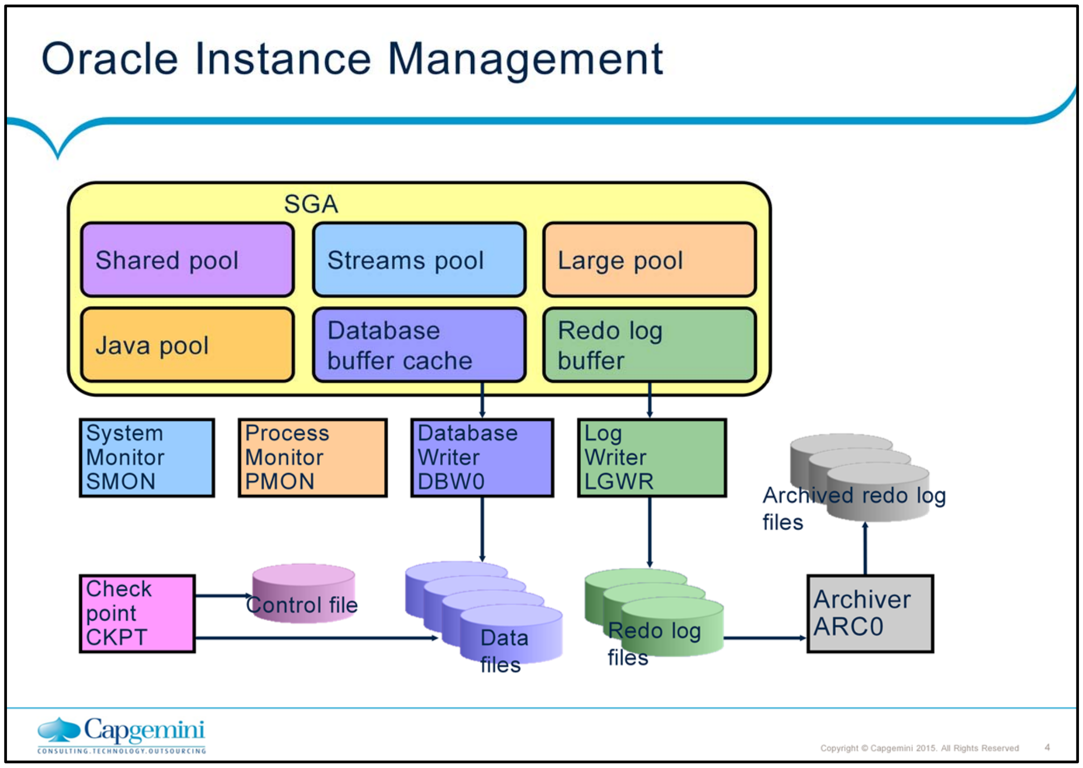
The data files where all data is stored

The parameter file, which contains the parameters that control the size and properties of an instance

The password file, which contains the superuser (SYSOPER and SYSDBA) passwords

The instance consists of memory structures such as System Global Area (SGA) and Program Global Area (PGA) and background processes that perform tasks within the database as well as the server processes that are initiated for each user session.

The size and structure of an Oracle database and instance impact performance. The physical structure of the database impacts the I/O to hard disks. It is therefore important to both size and place the physical files in such a way that I/O across disks is distributed evenly and waits are minimized. The size of the various memory areas of the instance directly impacts the speed of SQL processing.



Oracle Instance Management

Oracle Instance Management

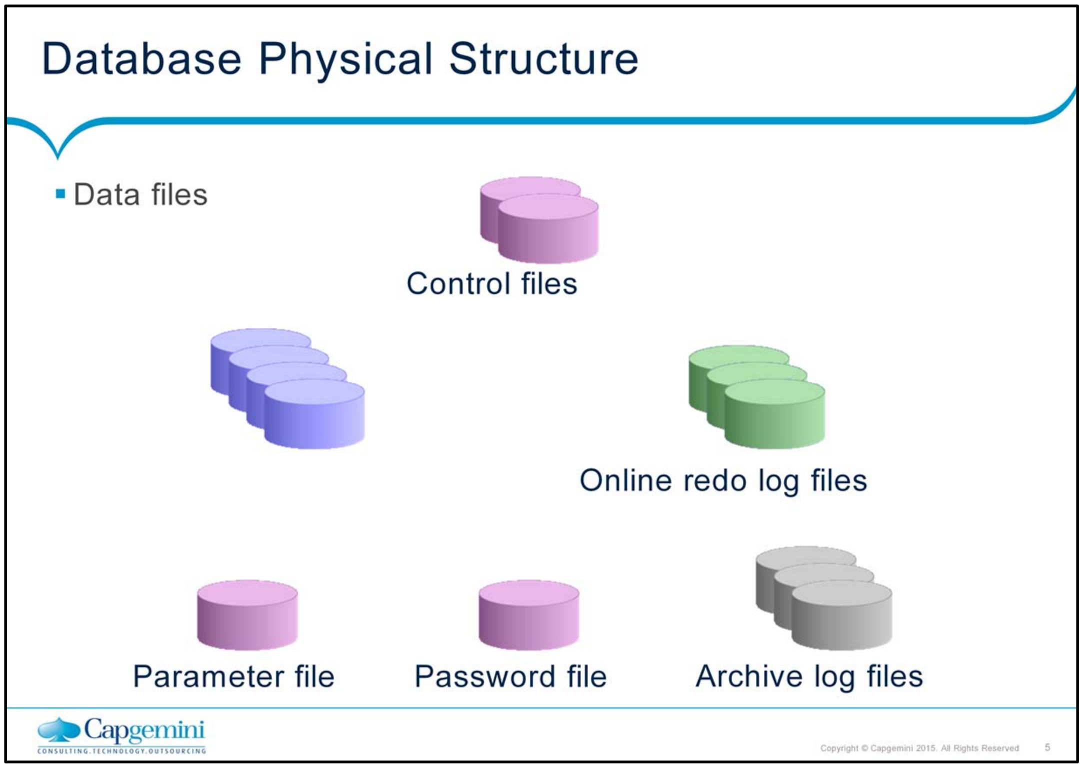
The instance controls access to the database. Only after the instance is started and the database is opened can users access the database.

An Oracle instance consists of the memory area known as the System Global Area (SGA) and some background processes and server processes.

The instance is idle (nonexistent) until it is started. When the instance is started, an initialization parameter file is read and the instance is configured accordingly. The initialization parameter file specifies various parameters that provide the size of the SGA.

You must take care in the initial design of the database system to avoid bottlenecks that may lead to performance problems. In addition, you need to consider:

- Allocating memory to database structures
- Determining I/O requirements of different parts of the database
- Tuning the operating system for optimal performance of the database



Database Physical Structure

- Data files

Database Physical Structure

The files that make up an Oracle database affect performance because their size and location affect the distribution of I/O, as well as scalability. These files are organized into the following:

Control files: These files contain data about the database itself. These files are critical to the database. Without them you cannot open the database.

Data files: These files contain the data of the database.

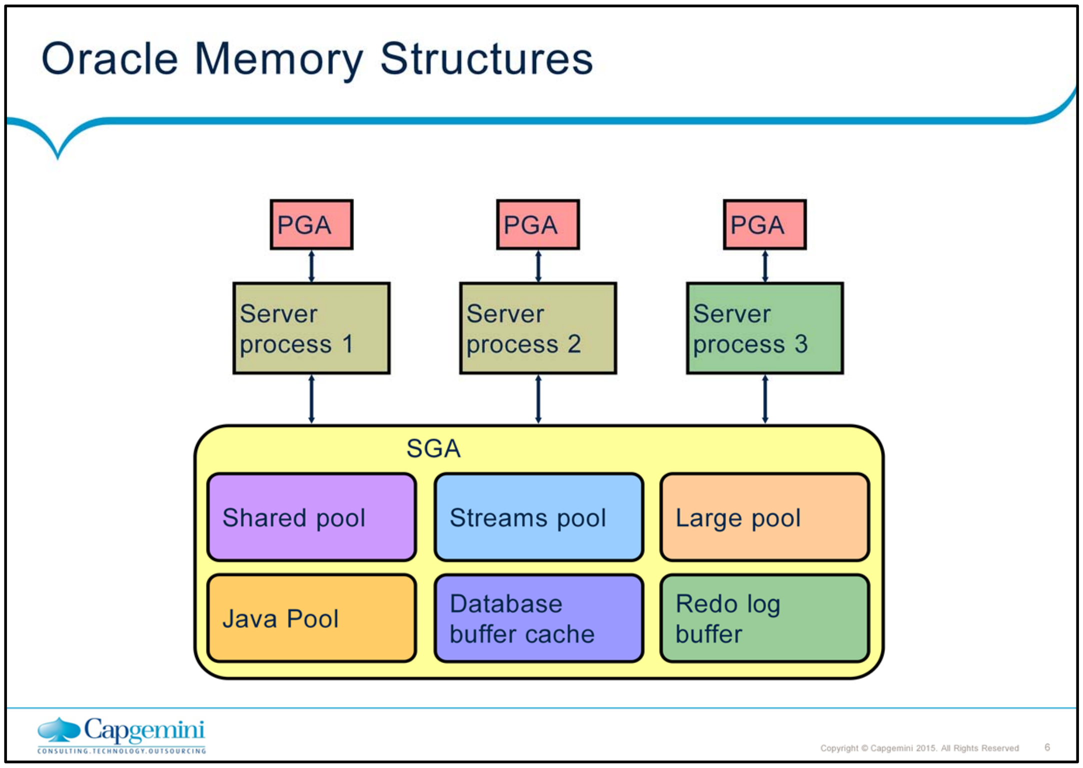
Online redo log files: These files allow for instance recovery of the database. If the database were to crash, the database can be recovered with the information in these files.

There are other files that are not officially part of the database but are important to the successful running of the database:

Parameter file: The parameter file is used to define how the instance is configured on startup.

Password file: This file enables superusers to connect remotely to the database and perform administrative tasks.

Archive log files: These files ensure database recovery and are copies of the online redo log files. Using these files and a backup of the database, you can recover a lost data file.



Oracle Memory Structures

Oracle Memory Structures

The default Oracle database, created by the Oracle Universal Installer (OUI), is preconfigured with initial settings for the memory parameters. The performance of the database depends on the sizing of these memory parameters, so you should fine-tune it to meet the requirements of your growing database.

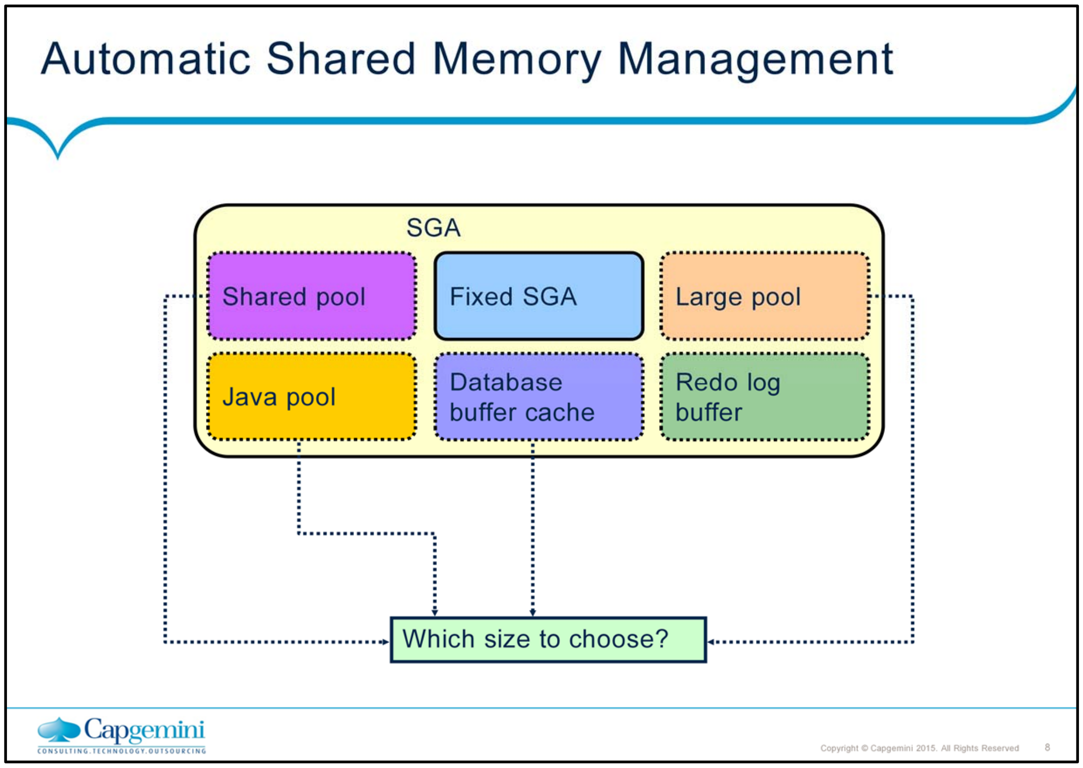
Two memory parameters, `PGA_AGGREGATE_TARGET` and `SGA_TARGET`, are provided that allow the database to automatically resize the memory structures within the SGA and PGA. These parameters can be set on the basis of the recommendations of Automatic Database Diagnostics Monitor (ADDM), which is available with the Enterprise Edition of Oracle Database 10*g*, or you can manually run several advisors and use the combined recommendations of these advisors to set the sizes appropriately.

The basic memory structures associated with an Oracle instance include:

- System Global Area (SGA): Shared by all server and background processes
- Program Global Area (PGA): Exclusive to each server and background process. There is one PGA for each server process.

Oracle Memory Structures (continued)

- The System Global Area (SGA) is a shared memory area that contains data and control information for the instance.
- The SGA consists of the following data structures:
  - Database buffer cache: Caches blocks of data retrieved from the data files
  - Redo log buffer: Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on disk
  - Shared pool: Caches various constructs that can be shared among users
  - Large pool: Optional area in the SGA that provides large memory allocations for Oracle backup and restore operations, I/O server processes, and session memory for the shared server
  - Java pool: Used for all session-specific Java code and data within the Java Virtual Machine (JVM)
  - Streams pool: Used by Oracle Streams
- The Program Global Area (PGA) is a memory region, which contains data and control information for each server process. A server process is a process that services a client's requests. Each server process has its own private PGA that is created when the server process is started. Only a server process can access its own PGA.
- Generally, the PGA contains the following:
  - Private SQL area: Contains data such as bind information and run-time memory structures. Each session that issues a SQL statement has a private SQL area.
  - Session memory: Memory allocated to hold session variables and other information related to the session



Automatic Shared Memory Management

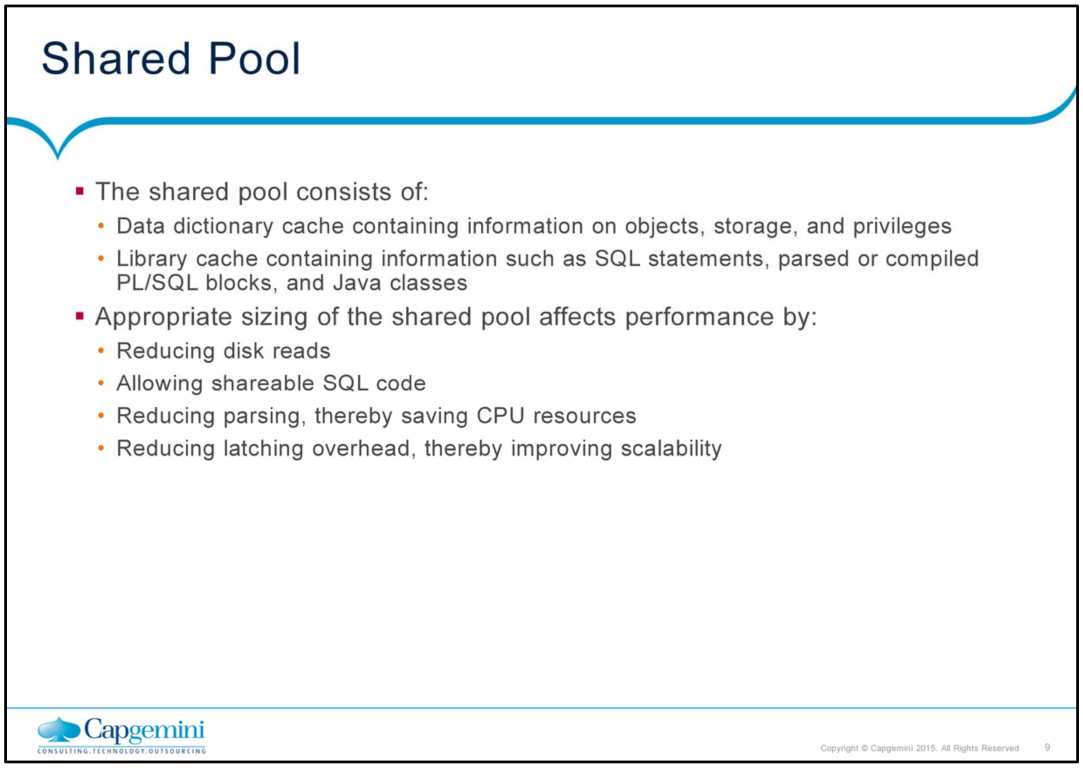
SGA
Shared pool
Fixed SGA
Large pool
Java pool
Database buffer cache
Redo log buffer
Which size to choose?

Automatic Shared Memory Management

In earlier releases of the Oracle Database, you did not have exact control over the total size of the SGA. The reason was that memory was allocated by Oracle for the fixed SGA and for other internal metadata allocations over and above the total size of the user-specified SGA parameters. In prior releases, you needed to manually specify the amount of memory to be allocated for the database buffer cache, shared pool, Java pool, and large pool. It was often a challenge to optimally size these components. Undersizing often resulted in poor performance and out-of-memory errors (`ORA-4031`), whereas oversizing wasted memory.

In Oracle Database 10*g*, you can use the Automatic Shared Memory Management (ASMM) feature to enable the database to automatically determine the size of each of these memory components within the limits of the total SGA size.

Oracle Database 10*g* uses an SGA size parameter (`SGA_TARGET`) that includes all the memory in the SGA, including all the automatically sized components, manually sized components, and any internal allocations during startup. ASMM simplifies the configuration of the SGA by enabling you to specify a total memory amount to be used for all SGA components. The Oracle Database then periodically redistributes memory between these components according to workload requirements.



Shared Pool

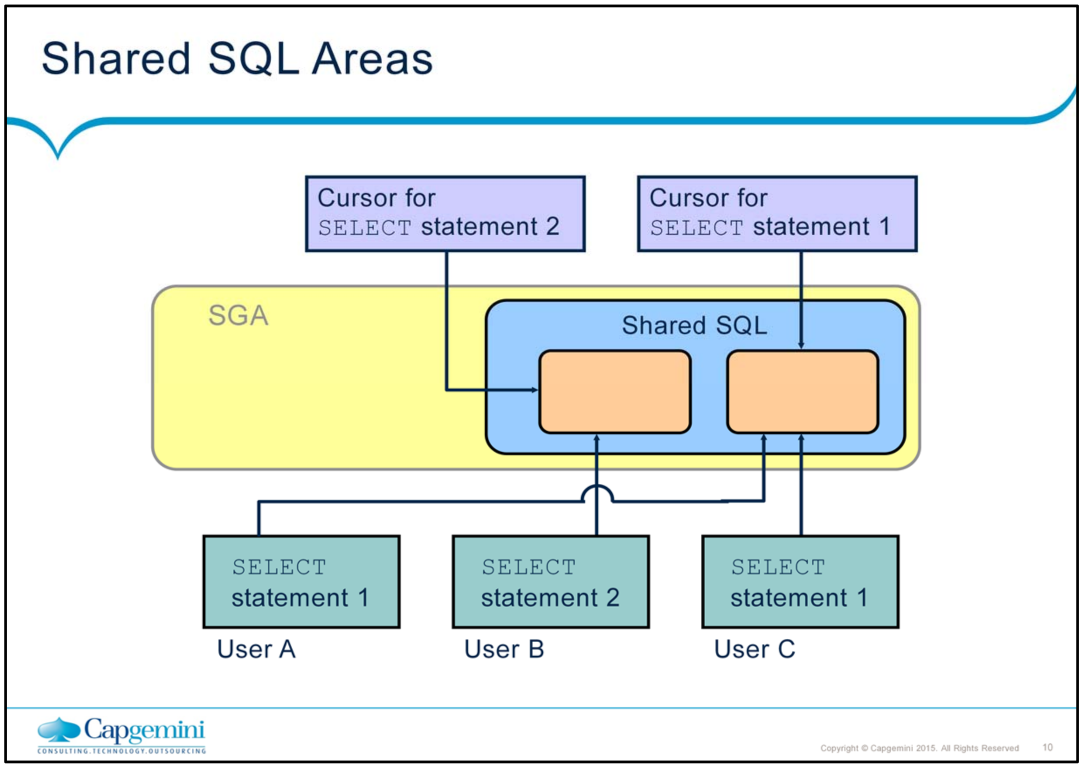
- The shared pool consists of:
  - Data dictionary cache containing information on objects, storage, and privileges
  - Library cache containing information such as SQL statements, parsed or compiled PL/SQL blocks, and Java classes
- Appropriate sizing of the shared pool affects performance by:
  - Reducing disk reads
  - Allowing shareable SQL code
  - Reducing parsing, thereby saving CPU resources
  - Reducing latching overhead, thereby improving scalability

Shared Pool

The shared pool is a portion of the SGA that contains shared memory constructs such as shared SQL areas, the data dictionary cache, and the fully parsed or compiled representations of PL/SQL blocks.

The main components of the shared pool are the library cache and the dictionary cache. The library cache stores the executable (parsed or compiled) form of recently referenced SQL and PL/SQL code. The dictionary cache stores information such as usernames, segment information, tablespace information, and sequence numbers.

The size of the shared pool affects the number of disk reads. When a SQL statement is executed, the server process checks the dictionary cache for information on object ownership, location, and privileges. If not present, this information is loaded into the dictionary cache through a disk read. The server process then verifies if a parsed form of the same statement exists in the library cache. If this information exists in the cache, then the statement is executed immediately. If not, then the statement is parsed and stored in the library cache for future use and then executed. Since disk reads and parsing are expensive operations, it is preferable that repeated executions of the same statement find required information in memory. Information in the dictionary and library cache are aged out using a least recently used (LRU) algorithm.



Shared SQL Areas

Shared SQL Areas

A shared SQL area is required to process every unique SQL statement submitted to a database.

A shared SQL area contains information such as the execution plan for the corresponding statement.

A single shared SQL area is used by multiple users that issue the same statement. In an OLTP environment, a single area therefore needs fewer hard parses, resulting in better resource utilization as well as leaving more shared memory for other uses.

Cursors

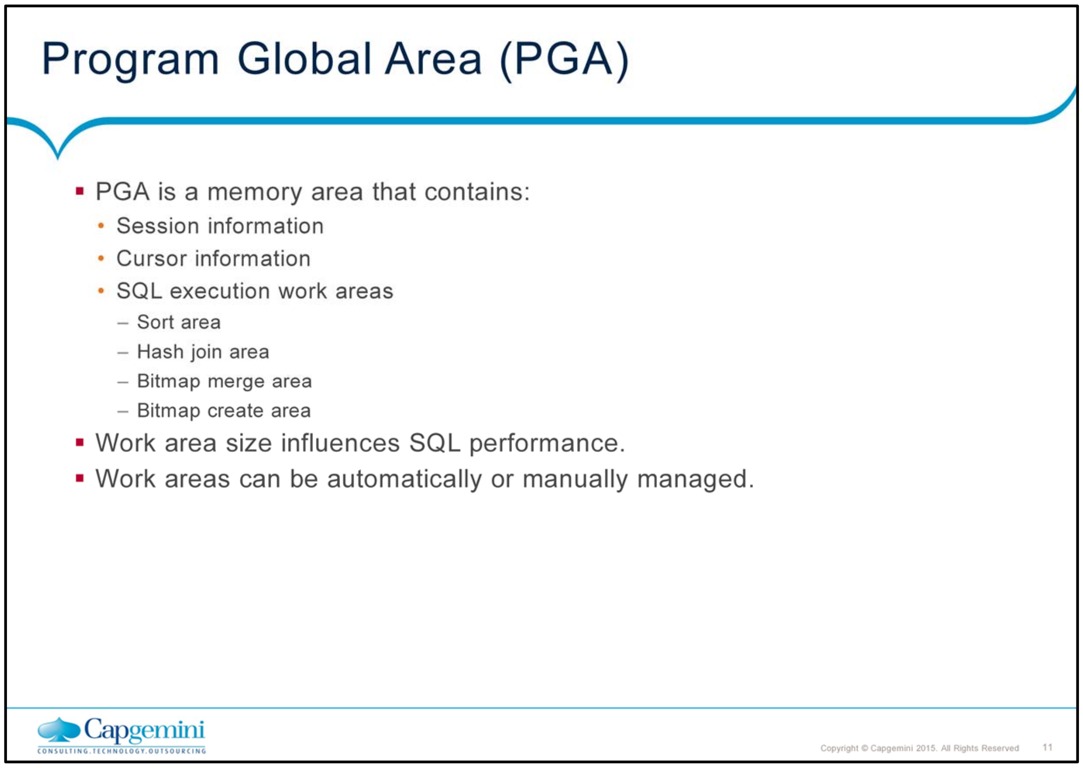
Inside the shared SQL area, each SQL statement is parsed in its own area, known as a cursor. If two users issue the exact same SQL statement, they may use the same cursor. Sharing cursors improves memory utilization and overall performance in SQL execution efficiency.

Each cursor holds the following information:

The parsed statement

The execution plan

A list of referenced objects



Program Global Area (PGA)

- PGA is a memory area that contains:
  - Session information
  - Cursor information
  - SQL execution work areas
    - Sort area
    - Hash join area
    - Bitmap merge area
    - Bitmap create area
- Work area size influences SQL performance.
- Work areas can be automatically or manually managed.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Program Global Area (PGA)

A PGA is a memory region containing data and control information for a single process. PGA is a nonshared memory area and a PGA is allocated for each server process. Only that server process can read and write to it. The Oracle Database allocates a PGA when a user connects to an Oracle database. A session is then created.

SQL Work Areas

The size of a work area can be controlled and tuned. Generally, bigger database areas can significantly improve the performance of a particular operator at the cost of higher memory consumption. Optimally, the size of a work area is big enough to accommodate the input data and auxiliary memory structures allocated by its associated SQL operator.

Complex operations such as sort-based operators (order by, group-by, rollup, window function) and joins require a big portion of the run-time area called *work areas* allocated by memory-intensive operators. For example, a sort operator uses a work area (sometimes called the *sort area*) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (also called the *hash area*) to build a hash table from its left input. If the amount of data to be processed by these two operators does not fit into a work area, then the input data is divided into smaller pieces. This allows some data pieces to be processed in memory while the rest are spilled to temporary disk storage to be processed later.

Program Global Area (PGA) (continued)

If not, response time increases, because part of the input data must be spilled to temporary disk storage. In the extreme case, if the size of a work area is far too small compared to the input data size, multiple passes over the data pieces must be performed. This can dramatically increase the response time of the operator.

Content of the PGA

The content of PGA memory can be classified as follows.

Private SQL Area

A private SQL area contains data such as bind information and run-time memory structures. Each session that issues a SQL statement has a private SQL area. Each user that submits the same SQL statement has his or her own private SQL area that uses a single shared SQL area. Thus, many private SQL areas can be associated with the same shared SQL area.

The private SQL area of a cursor is itself divided into two areas whose lifetimes are different:

The persistent area contains, for example, bind information. It is freed only when the cursor is closed.

The run-time area is freed when the execution is terminated

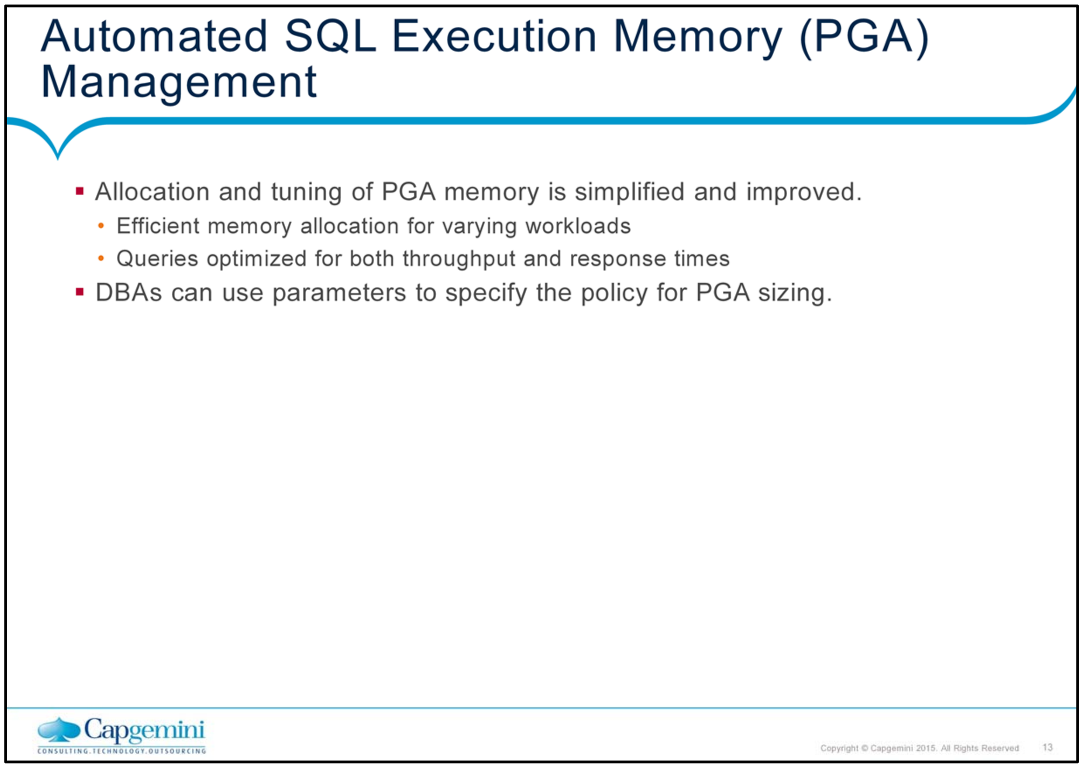
The Oracle Database creates the run-time area as the first step of an execute request. For DML statements, the Oracle Database frees the run-time area after the statement has been run. For queries, the Oracle Database frees the run-time area only after all rows are fetched or the query is canceled.

Cursors and SQL Areas

The user process manages the private SQL areas. The allocation and deallocation of private SQL areas depends on which application tool you are using. The number of private SQL areas that a user process can allocate is limited by the initialization parameter OPEN_CURSORS. The default value of this parameter is 50.

A private SQL area exists until the corresponding cursor is closed or the statement handle is freed. Although the Oracle Database frees the run-time area after the statement completes, the persistent area of the private SQL area remains waiting until the cursor is closed. Application developers close all open cursors that will not be used again to free the persistent area and to minimize the amount of memory required for users of the application.

Session memory is the memory allocated to hold a session’s variables (log-on information) and other information related to the session. For a shared server, the session memory is shared and not private.



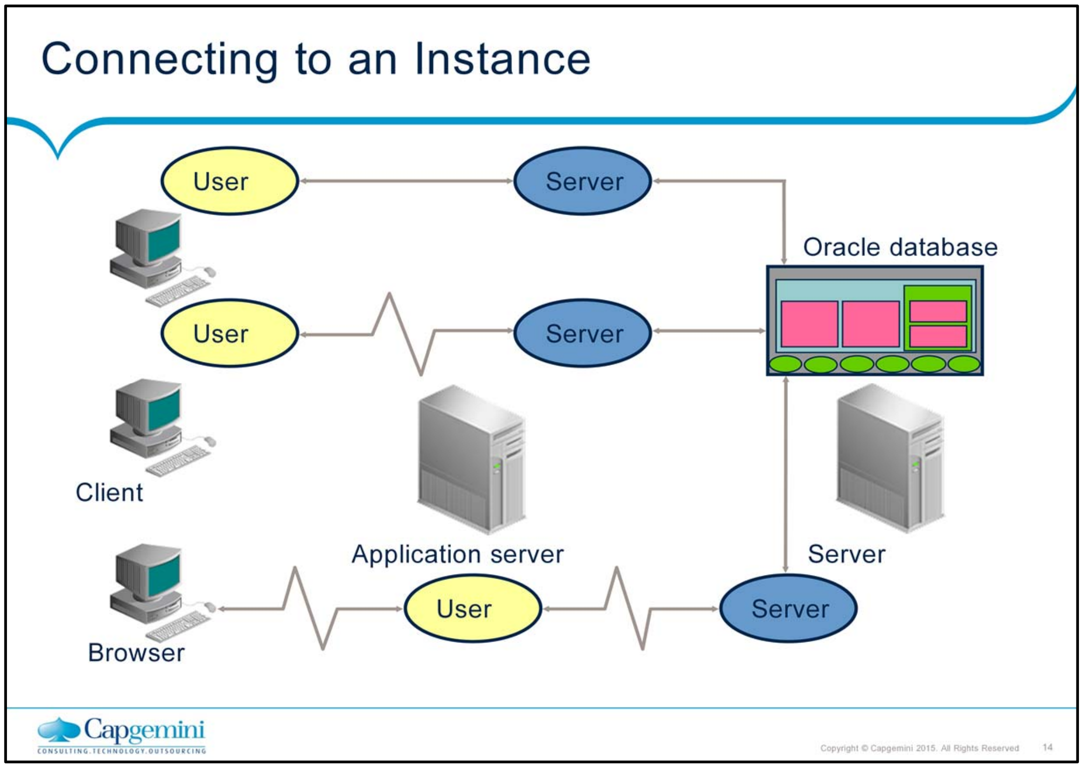
Automated SQL Execution Memory (PGA) Management

- Allocation and tuning of PGA memory is simplified and improved.
  - Efficient memory allocation for varying workloads
  - Queries optimized for both throughput and response times
- DBAs can use parameters to specify the policy for PGA sizing.

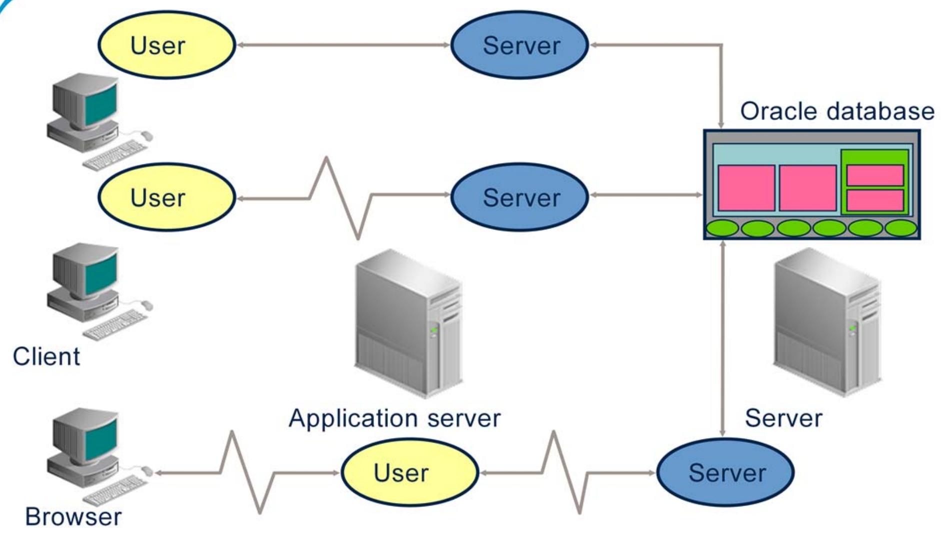
Automated SQL Execution Memory (PGA) Management

This feature provides an automatic mode for allocating memory to working areas in the PGA. It simplifies and improves the way PGA memory is allocated and tuned by the Oracle Database. DBAs can use the `PGA_AGGREGATE_TARGET` parameter to specify the total amount of memory that should be allocated to the PGA areas of the instance’s sessions. In automatic mode, working areas that are used by memory-intensive operators (sorts and hash-joins) can be automatically and dynamically adjusted.

This feature offers several performance and scalability benefits for DSS workloads or mixed workloads with complex queries. The overall system performance is maximized, and the available memory is allocated more efficiently among queries to optimize both throughput and response time. In particular, the savings that are gained from improved use of memory translate to better throughput at high loads.



Connecting to an Instance



Connecting to an Instance

When a user starts a tool (such as SQL*Plus) or connects to the database using an application, the application or tool is executed in a *user process*. When a user actually logs on to the Oracle database, a process is created on the computer running the Oracle database. The *listener* on the Oracle database actually establishes the connection and directs the request to an available *server process*. The server process communicates with the Oracle instance on behalf of the user process that runs on the client. The server process executes SQL statements on behalf of the user.

Connection

A connection is a communication pathway between a user process and an Oracle database. A database user can connect to an Oracle database in one of three ways:

The user logs on to the machine running the Oracle instance and starts an application or tool that accesses the database on that system. The communication pathway is established using the interprocess communication mechanisms available on the host operating system.

Connecting to an Instance (continued)

Connection (continued)

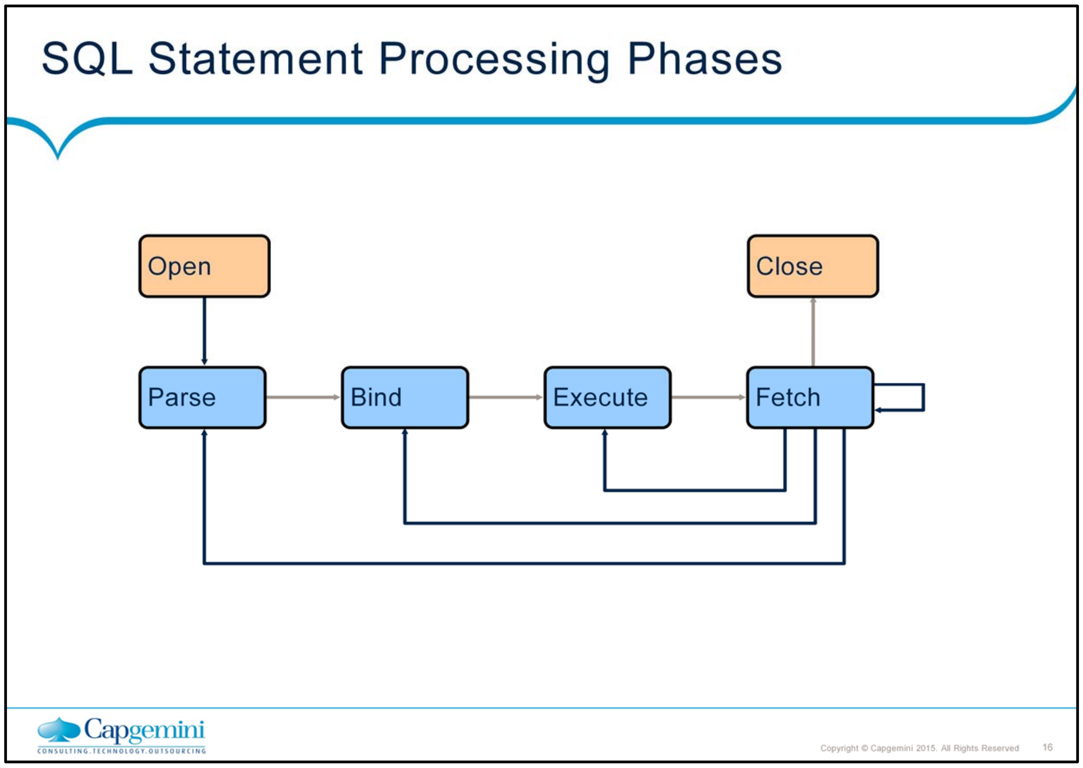
The user starts the application or tool on a local computer and connects over a network to the computer running the Oracle instance. In this configuration, called client/server, network software is used to communicate between the user and the Oracle database.

In a three-tiered connection, the user's computer communicates over the network to an application or a network server, which is connected through a network to the machine running the Oracle instance. For example, the user runs a browser on a network computer to use an application residing on an NT server that retrieves data from an Oracle database running on a UNIX host.

Sessions

A session is a specific connection of a user to an Oracle database. The session starts when the user is validated by the Oracle database, and it ends when the user logs out or when there is an abnormal termination. For a given database user, many concurrent sessions are possible if the user logs on from many tools, applications, or terminals at the same time. Except for some specialized database administration tools, starting a database session requires that the Oracle database be available for use.

Good database connection management offers benefits in minimizing the number of connections, thereby increasing scalability.



SQL Statement Processing Phases

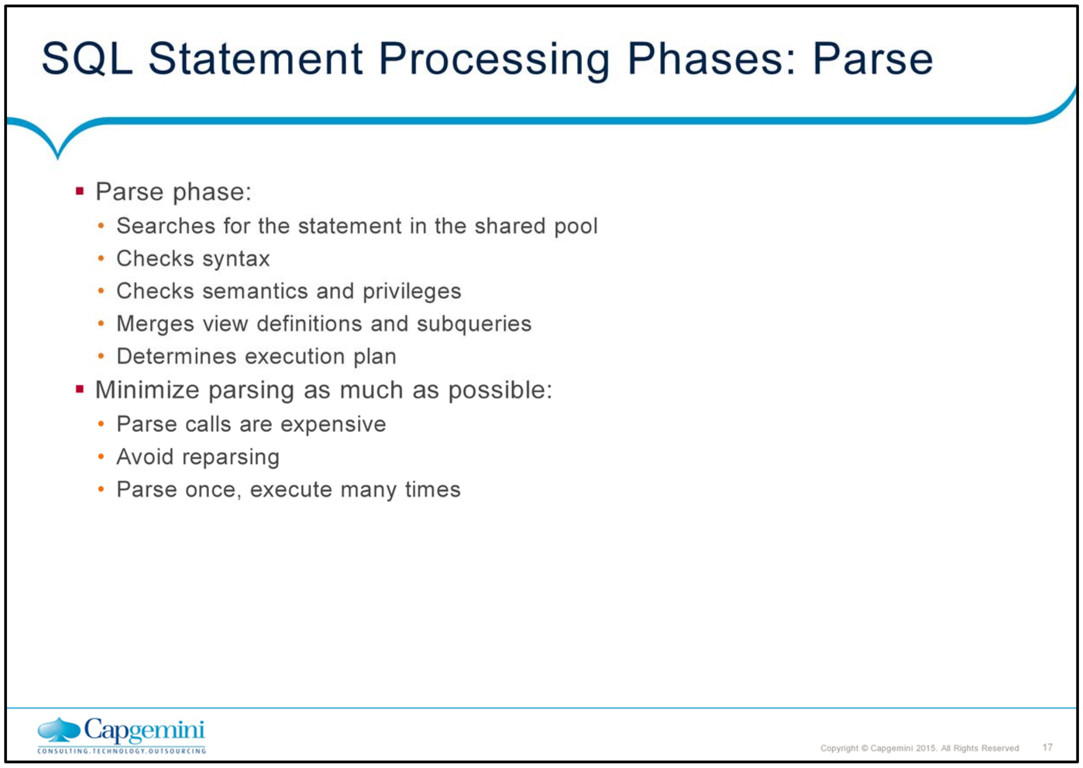
SQL Statement Processing Phases

A good understanding of SQL processing is essential for writing optimal SQL statements. In SQL statement processing, there are four important phases: parsing, binding, executing, and fetching.

The reverse arrows indicate processing scenarios (for example, Fetch—(Re)Bind—Execute—Fetch).

The Fetch phase applies only to queries and DML statements with a returning clause.

Note: A detailed description of SQL statement processing can be found in the *Oracle Database 10g Application Developers Guide: Fundamentals* and *Oracle Database 10g: Concepts*.



SQL Statement Processing Phases: Parse

- Parse phase:
  - Searches for the statement in the shared pool
  - Checks syntax
  - Checks semantics and privileges
  - Merges view definitions and subqueries
  - Determines execution plan
- Minimize parsing as much as possible:
  - Parse calls are expensive
  - Avoid reparsing
  - Parse once, execute many times

Parse Phase

Parsing is one of the stages in the processing of a SQL statement. When an application issues a SQL statement, the application makes a parse call to the Oracle database. During the parse call, the Oracle database:

Checks the statement for syntactic and semantic validity

Determines whether the process issuing the statement has privileges to run it

Allocates a private SQL area for the statement

The Oracle database first checks whether there is an existing parsed representation of the statement in the library cache. If so, the user process uses this parsed representation and runs the statement immediately. If not, the Oracle database generates the parsed representation of the statement, and the user process allocates a shared SQL area for the statement in the library cache and stores its parsed representation there.

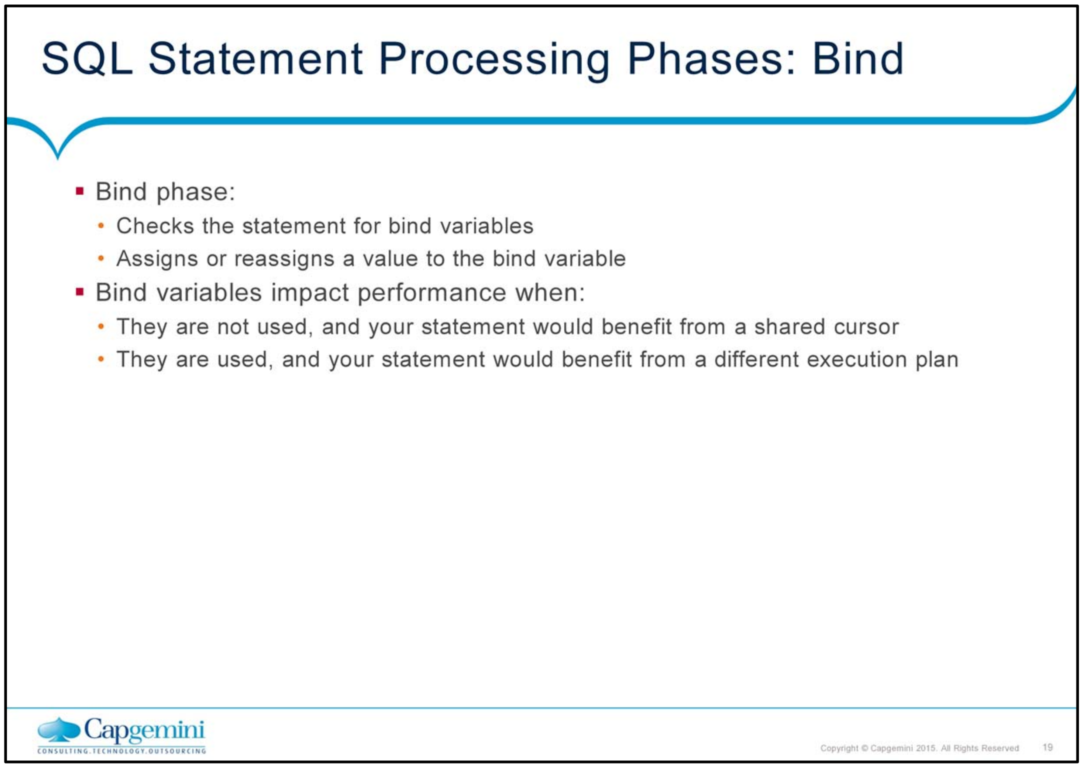
A parse operation by the Oracle database allocates a shared SQL area for a SQL statement. After a shared SQL area has been allocated for a statement, it can be run repeatedly without being reparsed. Both parse calls and parsing can be expensive relative to execution, so they should be minimized. Ideally, a statement should be parsed once and executed many times rather than reparsing for each execution.

Parse Phase (continued)

There are two types of parse operations:

- **Hard parsing:** A SQL statement is submitted for the first time, and no shareable match is found in the shared pool. Hard parses are the most resource-intensive and unscalable, because they perform all the operations involved in a parse.
- **Soft parsing:** A SQL statement is submitted, and a match *is* found in the shared pool. The match can be the result of a previous execution by another user. The SQL statement is shared, which is good for performance. However, soft parses still require syntax and security checking, which consume system resources.

When bind variables are used properly, more soft parses are possible, thereby reducing hard parses and keeping parsed statements in the library cache for a longer period.



SQL Statement Processing Phases: Bind

- Bind phase:
  - Checks the statement for bind variables
  - Assigns or reassigns a value to the bind variable
- Bind variables impact performance when:
  - They are not used, and your statement would benefit from a shared cursor
  - They are used, and your statement would benefit from a different execution plan

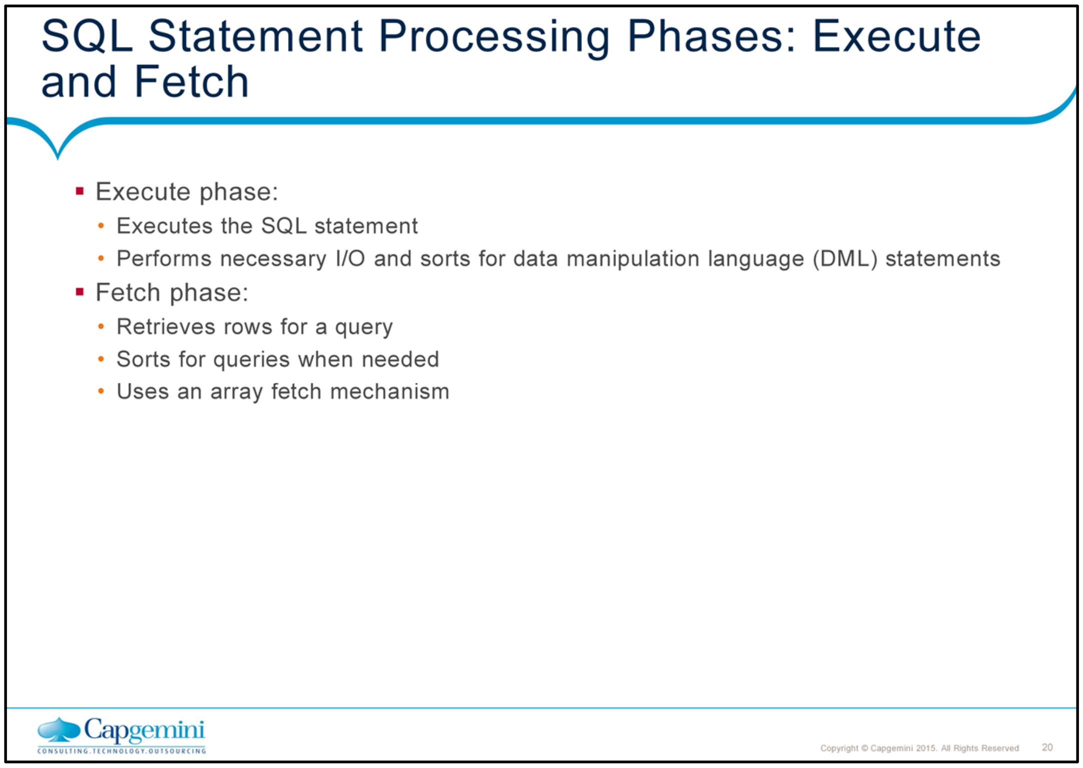
Bind Phase

During the bind phase:

The Oracle Database checks the statement for references of bind variables

The Oracle Database assigns or reassigns a value to each variable

When bind variables are used in a statement, the optimizer assumes that cursor sharing is intended and that different invocations should use the same execution plan. If different invocations of the cursor would significantly benefit from different execution plans, then using bind variables may adversely affect the performance of the SQL statement.



SQL Statement Processing Phases: Execute and Fetch

- Execute phase:
  - Executes the SQL statement
  - Performs necessary I/O and sorts for data manipulation language (DML) statements
- Fetch phase:
  - Retrieves rows for a query
  - Sorts for queries when needed
  - Uses an array fetch mechanism

Execute Phase

The Oracle Database uses the execution plan to identify the required rows of data from the data buffers. Multiple users can share the same execution plan. The Oracle Database performs physical reads or logical reads/writes for DML statements and also sorts the data when needed.

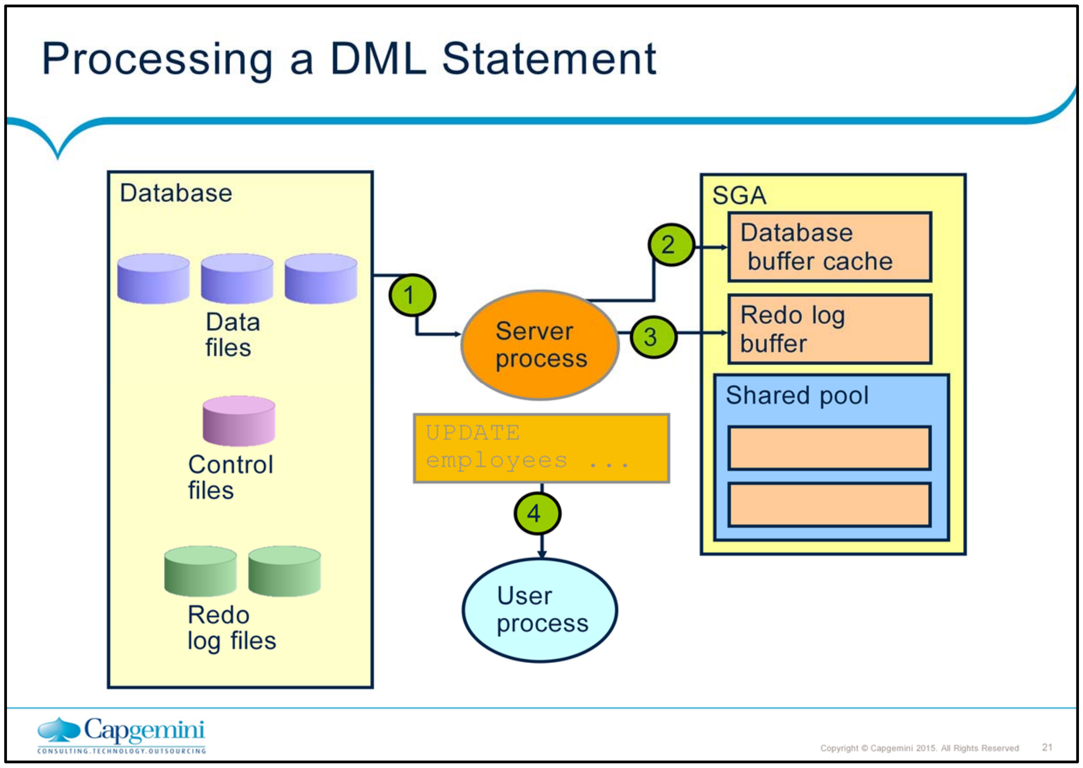
Note: Physical reads are disk reads; logical reads are blocks already in memory in the buffer cache. Physical reads are more expensive because they require I/O from disk.

Fetch Phase

The Oracle Database retrieves rows for a SELECT statement during the fetch phase. Each fetch typically retrieves multiple rows, using an array fetch. Each Oracle tool offers its own ways of influencing the array size; in SQL*Plus, you do so by using the ARRAYSIZE setting:

```
SQL> show arraysize
arraysize 15
SQL> set arraysize 1
```

SQL*Plus processes one row at a time with this setting. The default value is 15. There is no advantage in setting array sizes greater than 100 in SQL*Plus.



Processing a DML Statement

DML Processing Steps

A data manipulation language (DML) statement requires only two phases of processing:

Parse is the same as the parse phase used for processing a query.

Execute requires additional processing to make data changes.

DML Execute Phase

To execute a DML statement:

1. If the data and rollback blocks are not already in the buffer cache, the server process reads them from the data files into the buffer cache. The server process locks the rows that are to be modified.
2. The server process records the changes to be made to the data buffers as well as the undo changes. These changes are written to the redo log buffer before the in-memory data and rollback buffers are modified. This is called *write-ahead logging*.
3. The rollback buffers contain values of the data before it is modified. The rollback buffers are used to store the before image of the data so that the DML statements can be rolled back if necessary. The data buffers record the new values of the data.
4. The user gets the feedback from the DML operation (such as how many rows were affected by the operation).

DML Processing Steps (continued)

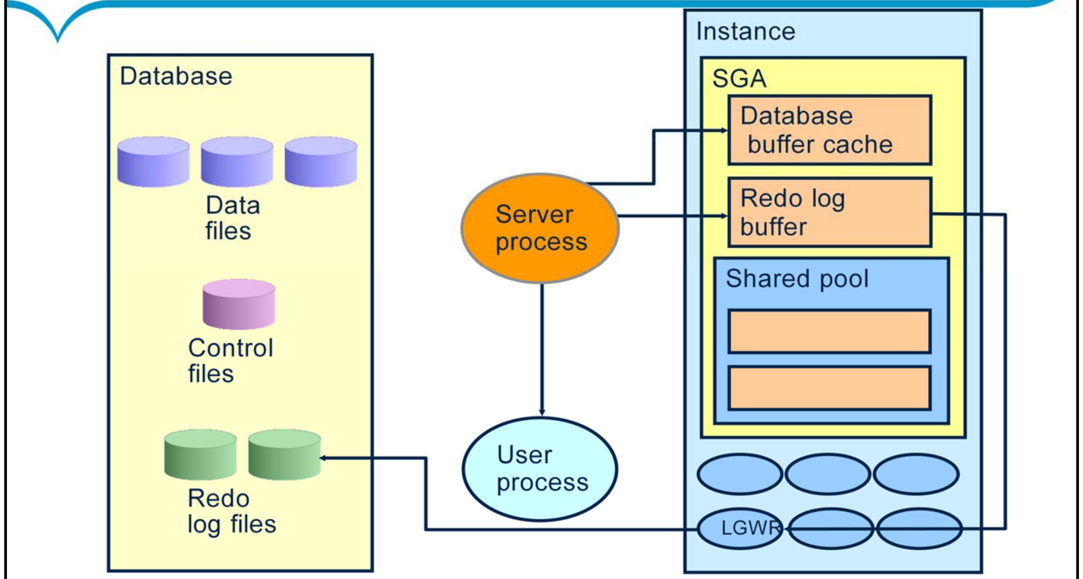
DML Execute Phase (continued)

Any changed blocks in the buffer cache are marked as dirty buffers; that is, the buffers are not the same as the corresponding blocks on the disk. These buffers are not immediately written to disk by the Database Writer (DBWR) process. When a transaction is committed, the changes made to the blocks are immediately recorded in the redo logs by the Log Writer process and are later written to disk by DBWR based on an internal algorithm.

The processing of a `DELETE` or `INSERT` command uses similar steps. The before image for a `DELETE` contains the column values in the deleted row, and the before image of an `INSERT` contains the row location information.

Until a transaction is committed, the changes made to the blocks are only recorded in memory structures and are not written immediately to disk. A computer failure that causes the loss of the SGA can also lose these changes.

COMMIT Processing



Fast `COMMIT`

The Oracle Database uses a Fast `COMMIT` mechanism that guarantees the committed changes can be recovered in case of instance failure.

System Change Number

Whenever a transaction commits, the Oracle Database assigns a commit system change number (SCN) to the transaction. The SCN is monotonically incremented and is unique within the database. It is used by the Oracle Database as an internal time stamp to synchronize data and to provide read consistency when data is retrieved from the data files. Using the SCN enables the Oracle Database to perform consistency checks without depending on the date and time of the operating system.

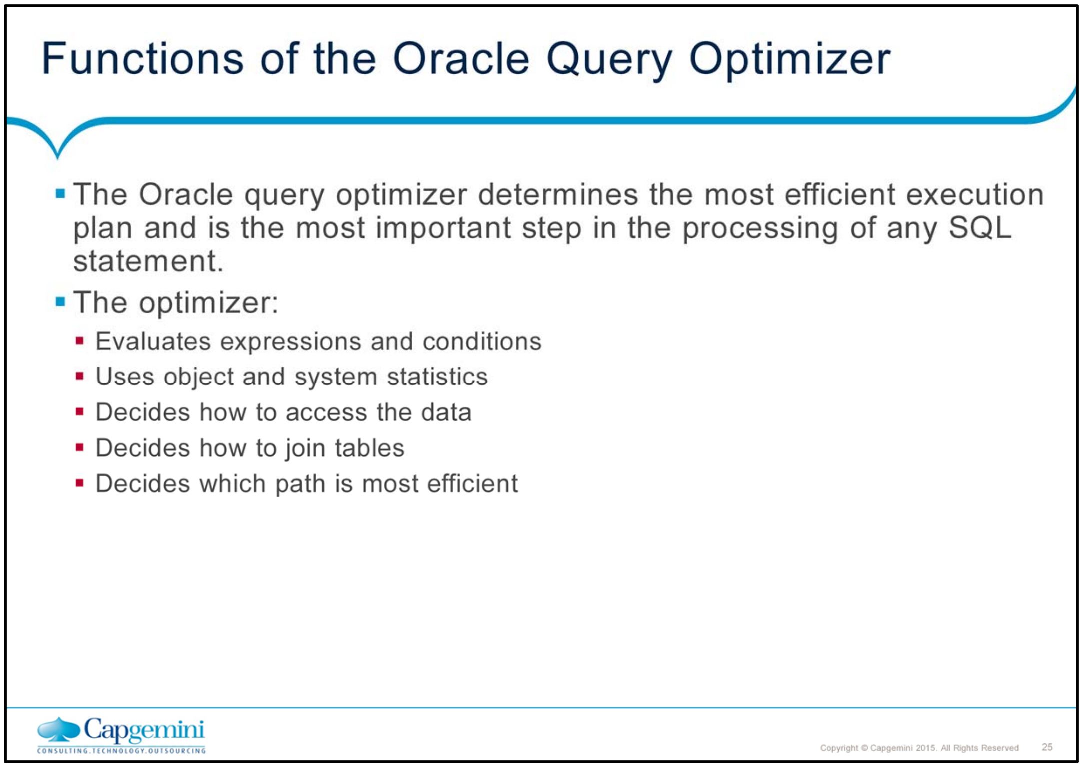
Fast `COMMIT` (continued)

System Change Number (continued)

When a `COMMIT` is issued, the following steps are performed:

- The server process places a commit record, along with the SCN, in the redo log buffer.
- The background Log Writer process (`LGWR`) performs a contiguous write of all the redo log buffer entries up to and including the commit record to the redo log files. After this point, the Oracle Database can guarantee that the changes will not be lost even if there is an instance failure.
- The server process provides feedback to the user process about the completion of the transaction.

`DBWR` eventually writes the actual changes back to disk based on its own internal timing mechanism.



Functions of the Oracle Query Optimizer

- The Oracle query optimizer determines the most efficient execution plan and is the most important step in the processing of any SQL statement.
- The optimizer:
  - Evaluates expressions and conditions
  - Uses object and system statistics
  - Decides how to access the data
  - Decides how to join tables
  - Decides which path is most efficient

Functions of the Oracle Query Optimizer

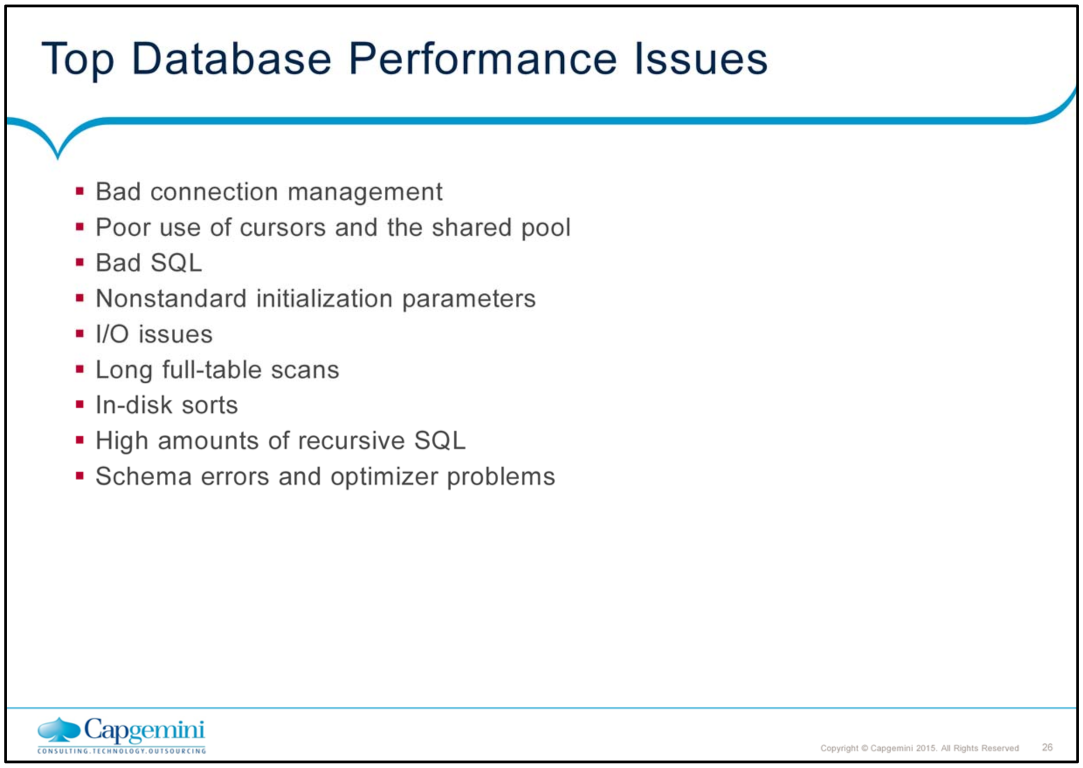
The optimizer is the part of the Oracle Database that creates the execution plan for a SQL statement. The determination of the execution plan is an important step in the processing of any SQL statement and can greatly affect execution time.

The execution plan is a series of operations that are performed in sequence to execute the statement. You have seen the various steps in executing a SQL statement in previous slides. The optimizer considers many factors related to the objects referenced and the conditions specified in the query. The information necessary to the optimizer includes:

Statistics gathered for the system (I/O, CPU, and so on) as well as schema objects (number of rows, index, and so on)
Information in the dictionary
•`WHERE` clause qualifiers
Hints supplied by the developer

When you use diagnostic tools such as Enterprise Manager, `EXPLAIN PLAN`, and SQL*Plus `AUTOTRACE`, you can see the execution plan that the optimizer chooses.

Note: In Oracle Database 10*g*, the optimizer has two names based on its functionality: the query optimizer and the Automatic Tuning Optimizer.



Top Database Performance Issues

- Bad connection management
- Poor use of cursors and the shared pool
- Bad SQL
- Nonstandard initialization parameters
- I/O issues
- Long full-table scans
- In-disk sorts
- High amounts of recursive SQL
- Schema errors and optimizer problems

Top Database Performance Issues

Bad connection management: The application connects and disconnects for each database interaction. This problem is common with stateless middleware in application servers. Additionally, simultaneous connections from the same client are also a waste of system and network resources

Poor use of cursors and the shared pool: Not reusing cursors results in repeated parses. If bind variables are not used, then there is hard parsing of all SQL statements. This has an order-of-magnitude impact in performance, and it is totally unscalable. Use cursors with bind variables that open the cursor and execute it many times. Be suspicious of applications generating dynamic SQL.

Bad SQL: Bad SQL is SQL that uses more resources than appropriate for the application requirement. This can be a decision support systems (DSS) query that runs for more than 24 hours or a query from an online application that takes more than a minute. SQL that consumes significant system resources should be investigated for potential improvement. ADDM identifies high-load SQL, and the SQL Tuning Advisor can be used to provide recommendations for improvement.

Top Database Performance Issues (continued)

Use of nonstandard initialization parameters: These might have been implemented based on poor advice or incorrect assumptions. Most systems will give acceptable performance using only the set of basic parameters. In particular, parameters associated with `SPIN_COUNT` on latches and undocumented optimizer features can cause a great deal of problems that can require considerable investigation.

I/O issues: If you configure your database to use multiple disks by disk space and not I/O bandwidth, then there will be excessive I/O to certain disks and little I/O to others. Frequently and simultaneously accessed objects (a table and its index) should be designed to be stored over different disks.

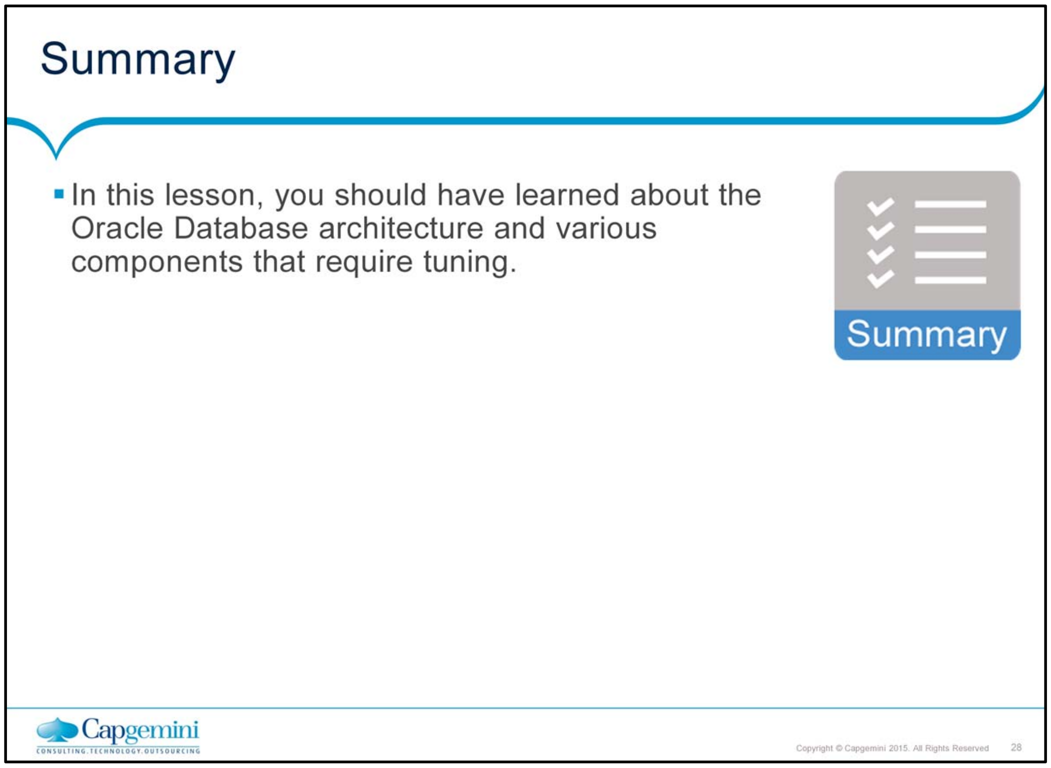
Long full-table scans: Long full-table scans for high-volume or interactive online operations could indicate poor transaction design, missing indexes, or poor SQL optimization.

In-disk sorting: In-disk sorts for online operations could indicate poor transaction design, missing indexes, or poor SQL optimization. Disk sorts, by nature, are I/O-intensive and unscalable.

High amounts of recursive SQL: Large amounts of recursive SQL executed by SYS could indicate space management activities, such as extent allocations, taking place. This is unscalable and impacts user response time. Recursive SQL executed under another user ID is probably SQL and PL/SQL, and this is not a problem.

Schema errors and optimizer problems: In many cases, an application uses too many resources because the schema owning the tables has not been successfully migrated from the development environment or from an older implementation. Examples of this are missing indexes or incorrect statistics. These errors can lead to suboptimal execution plans and poor interactive user performance. When migrating applications of known performance, you should export the schema statistics to maintain plan stability by using the `DBMS_STATS` package.

Likewise, optimizer parameters set in the initialization parameter file can override proven optimal execution plans. For these reasons, schemas, schema statistics, and optimizer settings should be managed together as a group to ensure consistency of performance.



Summary

- In this lesson, you should have learned about the Oracle Database architecture and various components that require tuning.

Summary

In this lesson, you have seen a brief overview of the Oracle Database architecture and different components that are involved in the tuning process.