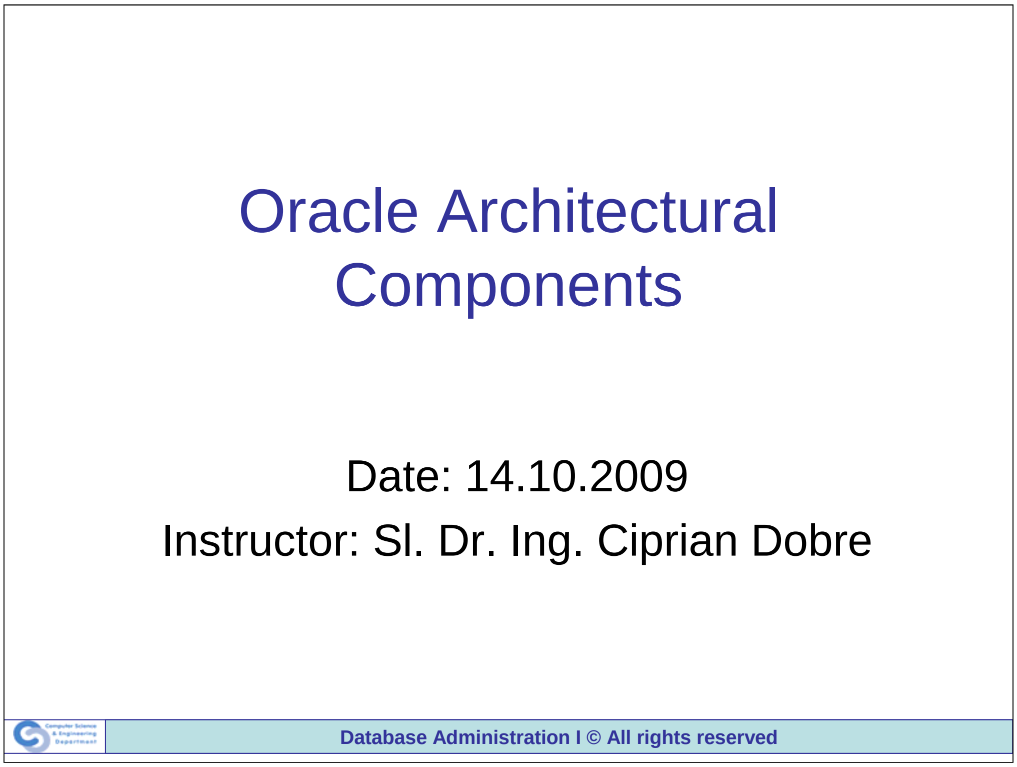
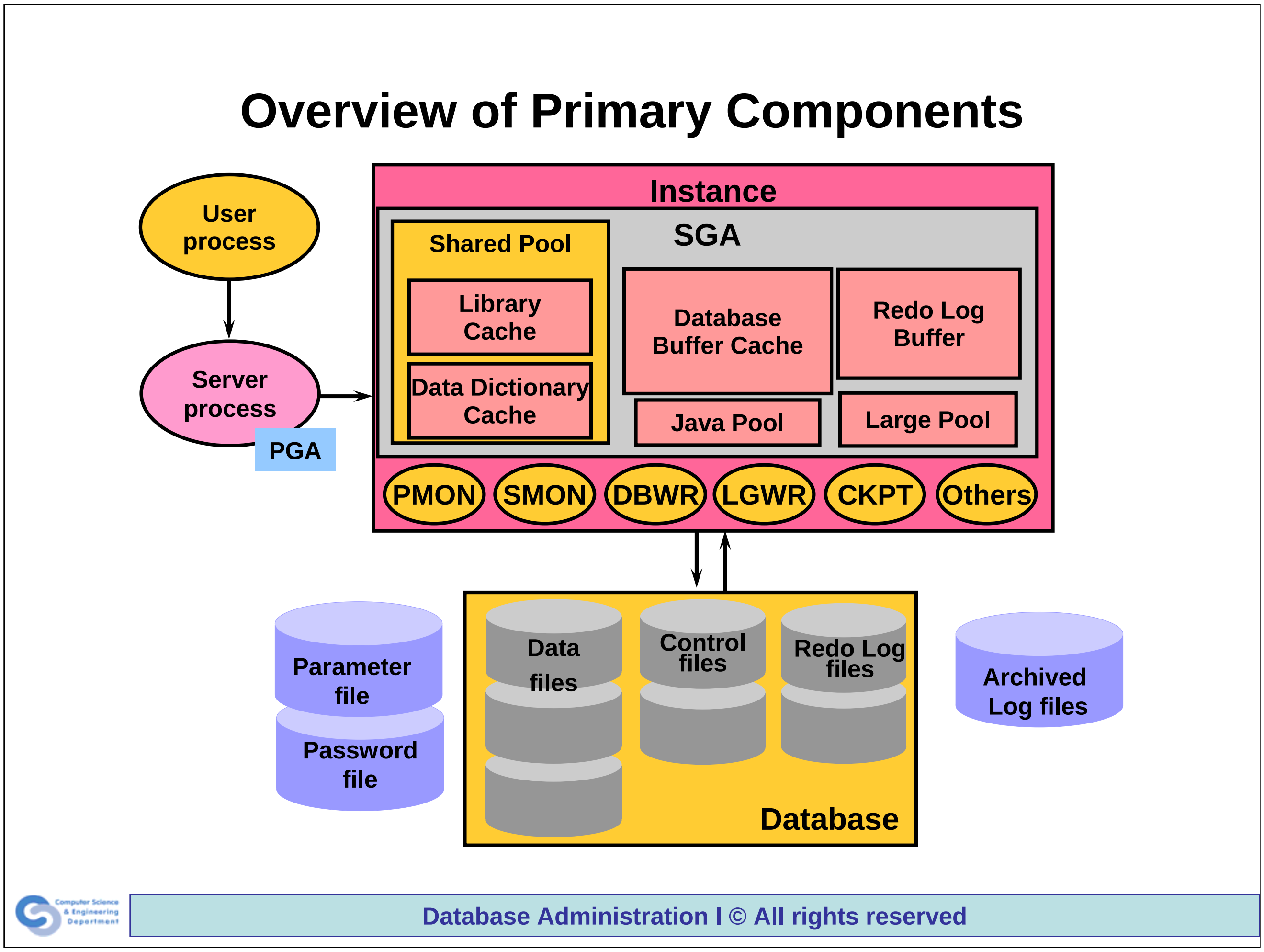


Oracle Architectural Components

Date: 14.10.2009

Instructor: Sl. Dr. Ing. Ciprian Dobre

Database Administration I © All rights reserved



Overview of Primary Components

User process

Server process

PGA

Instance

SGA

Shared Pool

Library Cache

Data Dictionary Cache

Database Buffer Cache

Redo Log Buffer

Java Pool

Large Pool

PMON

SMON

DBWR

LGWR

CKPT

Others

Parameter file

Password file

Data files

Control files

Redo Log files

Database

Archived Log files

Overview of Primary Components

The Oracle architecture includes a number of primary components, which are discussed further in this lesson.

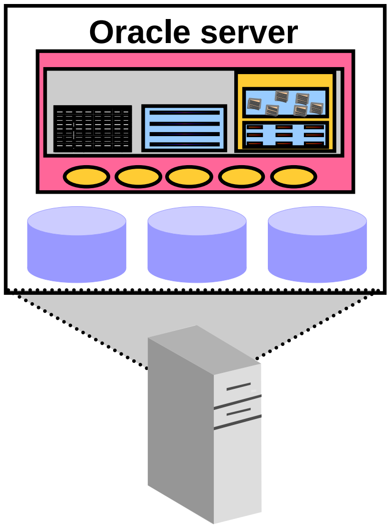
Oracle server: There are several files, processes, and memory structures in an Oracle server; however, not all of them are used when processing a SQL statement. Some are used to improve the performance of the database, to ensure that the database can be recovered in the event of a software or hardware error, or to perform other tasks necessary to maintain the database. The Oracle server consists of an Oracle instance and an Oracle database.

Oracle instance: An Oracle instance is the combination of the background processes and memory structures. The instance must be started to access the data in the database. Every time an instance is started, a System Global Area (SGA) is allocated and Oracle background processes are started. Background processes perform functions on behalf of the invoking process. They consolidate functions that would otherwise be handled by multiple Oracle programs running for each user. The background processes perform input/output (I/O) and monitor other Oracle processes to provide increased parallelism for better performance and reliability.

Oracle Server

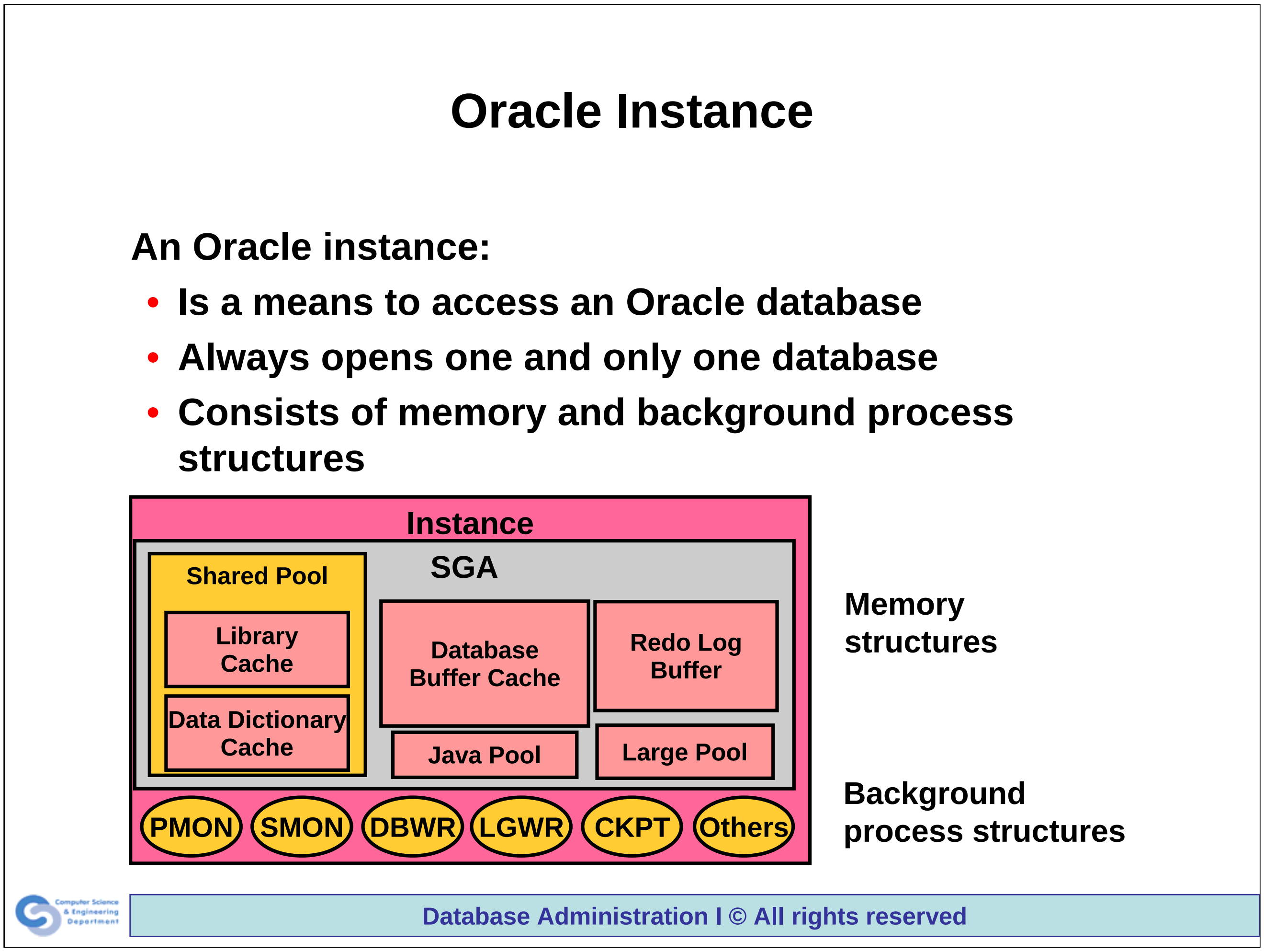
An Oracle server:

- Is a database management system that provides an open, comprehensive, integrated approach to information management
- Consists of an Oracle instance and an Oracle database



Oracle Server

The Oracle server is the key to information management. In general, an Oracle server must reliably manage a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this must be accomplished while delivering high performance. An Oracle server must also prevent unauthorized access and provide efficient solutions for failure recovery.



Oracle Instance

An Oracle instance:

- Is a means to access an Oracle database
- Always opens one and only one database
- Consists of memory and background process structures

Instance

SGA

Shared Pool

Library Cache

Data Dictionary Cache

Database Buffer Cache

Redo Log Buffer

Java Pool

Large Pool

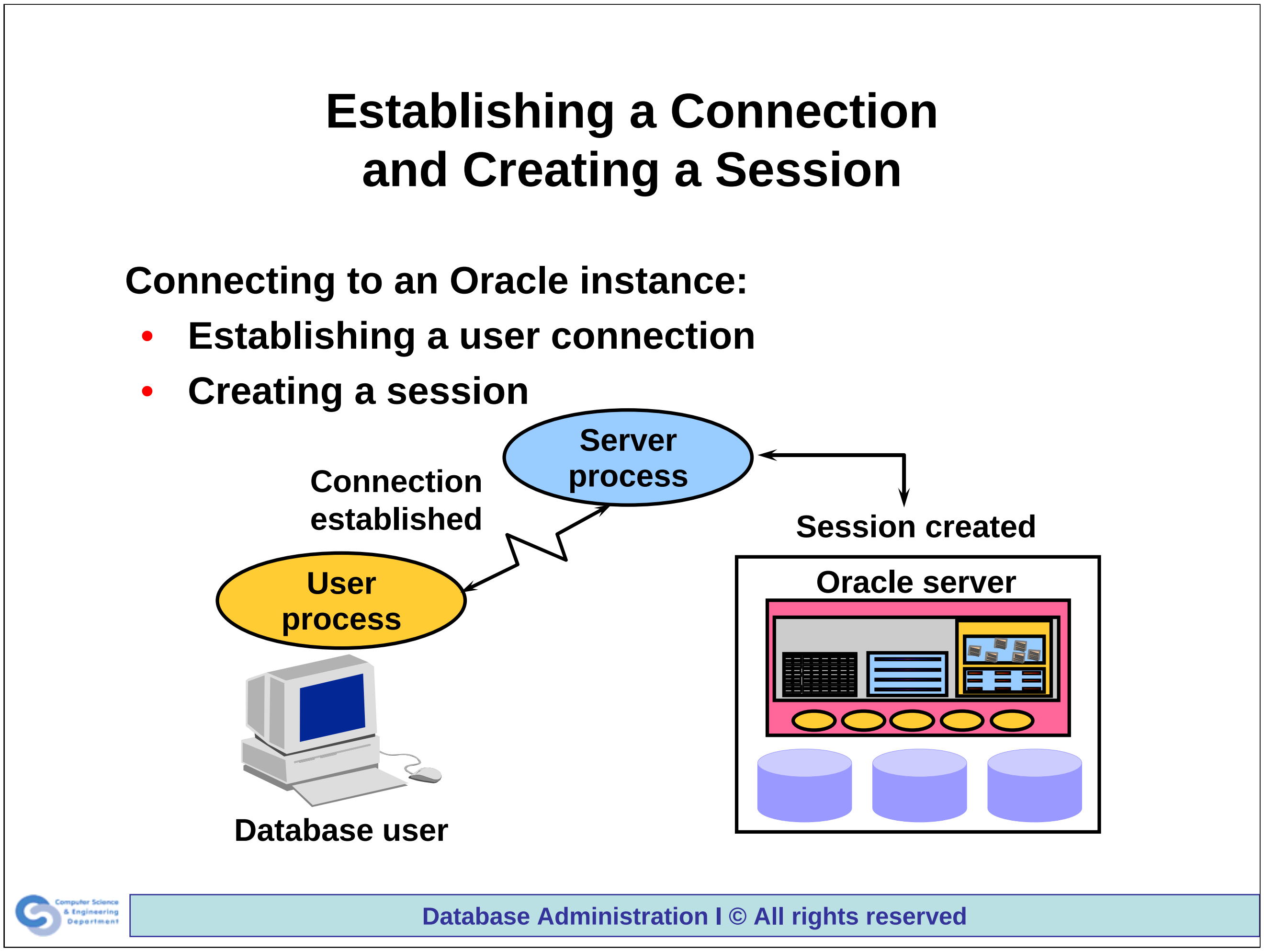
PMON SMON DBWR LGWR CKPT Others

Memory structures

Background process structures

Oracle Instance

An Oracle instance consists of the System Global Area (SGA) memory structure and the background processes that are used to manage a database. An instance is identified by using methods specific to each operating system. The instance can open and use only one database at a time.



Establishing a Connection and Creating a Session

Connecting to an Oracle instance:

- Establishing a user connection
- Creating a session

Establishing a Connection and Creating a Session

Before users can submit SQL statements to an Oracle database, they must connect to an instance.

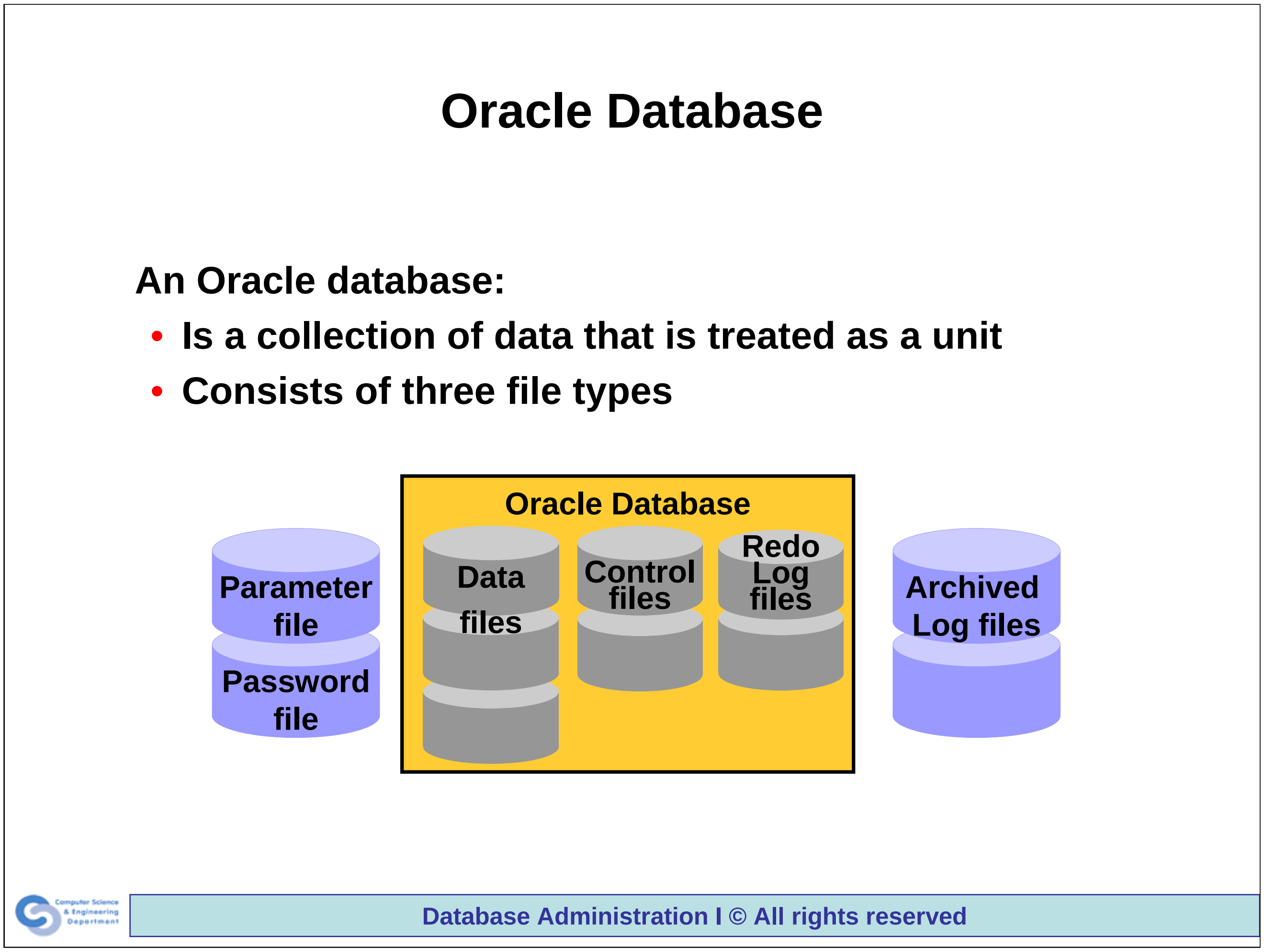
The user starts a tool such as SQL*Plus or runs an application developed using a tool such as Oracle Forms. This application or tool is executed as a user process.

In the most basic configuration, when a user logs on to the Oracle server, a process is created on the computer running the Oracle server. This process is called a 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 server. A database user can connect to an Oracle server in one of three ways:

The user logs on to the operating system 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.



Oracle Database

An Oracle database:

- Is a collection of data that is treated as a unit
- Consists of three file types

Oracle Database

The general purpose of a database is to store and retrieve related information. An Oracle database has a logical and a physical structure. The physical structure of the database is the set of operating system files in the database. An Oracle database consists of three file types.

Data files containing the actual data in the database

Online redo log files containing a record of changes made to the database to enable recovery of the data in case of failures

Control files containing information necessary to maintain and verify database integrity

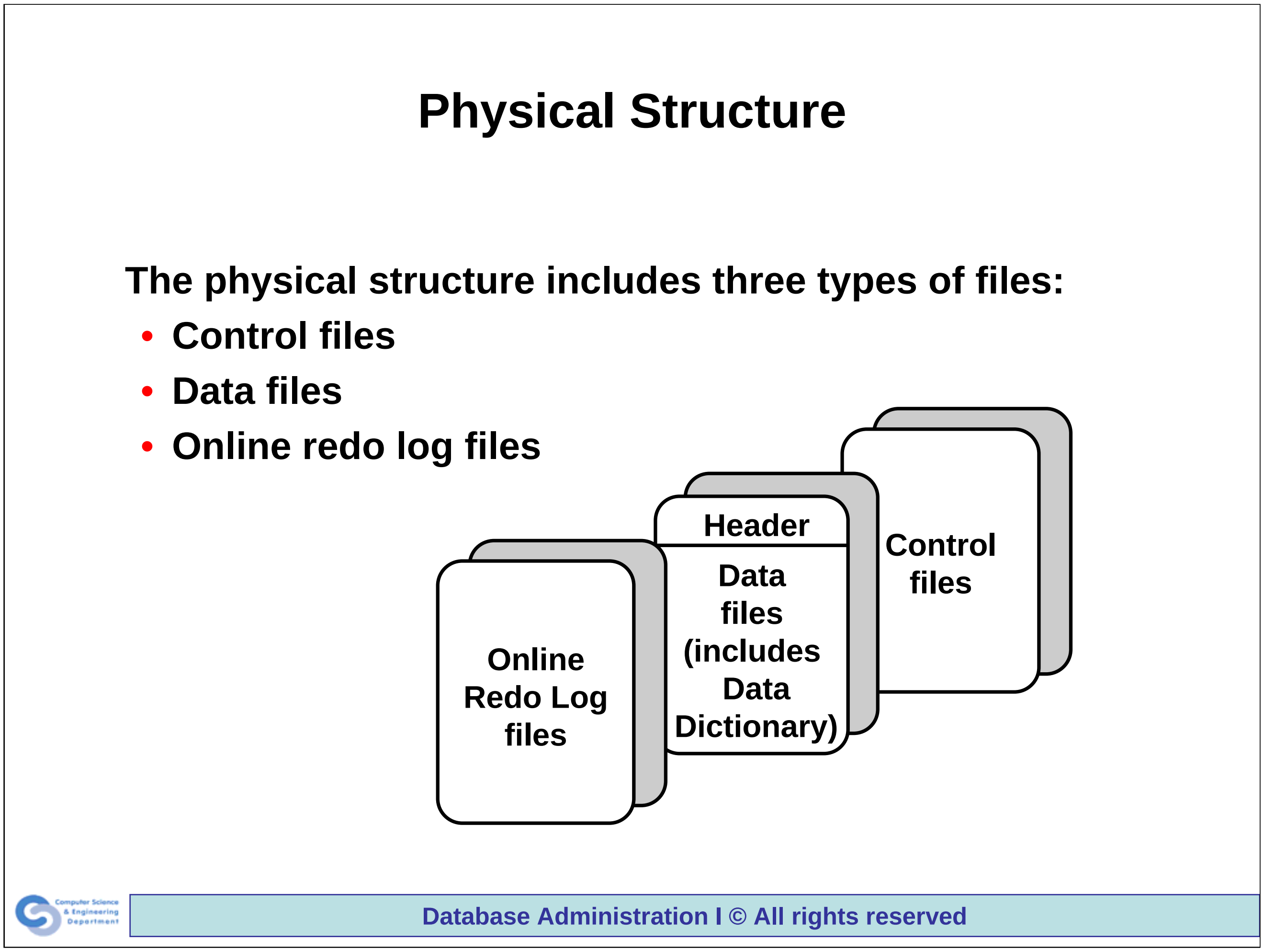
Other Key File Structures

The Oracle server also uses other files that are not part of the database:

The parameter file defines the characteristics of an Oracle instance. For example, it contains parameters that size some of the memory structures in the SGA.

The password file authenticates users privileged to start up and shut down an Oracle instance.

Archived redo log files are offline copies of the online redo log files that may be necessary to recover from media failures.



Physical Structure

The physical structure includes three types of files:

- Control files
- Data files
- Online redo log files

Online Redo Log files

Header

Data files (includes Data Dictionary)

Control files

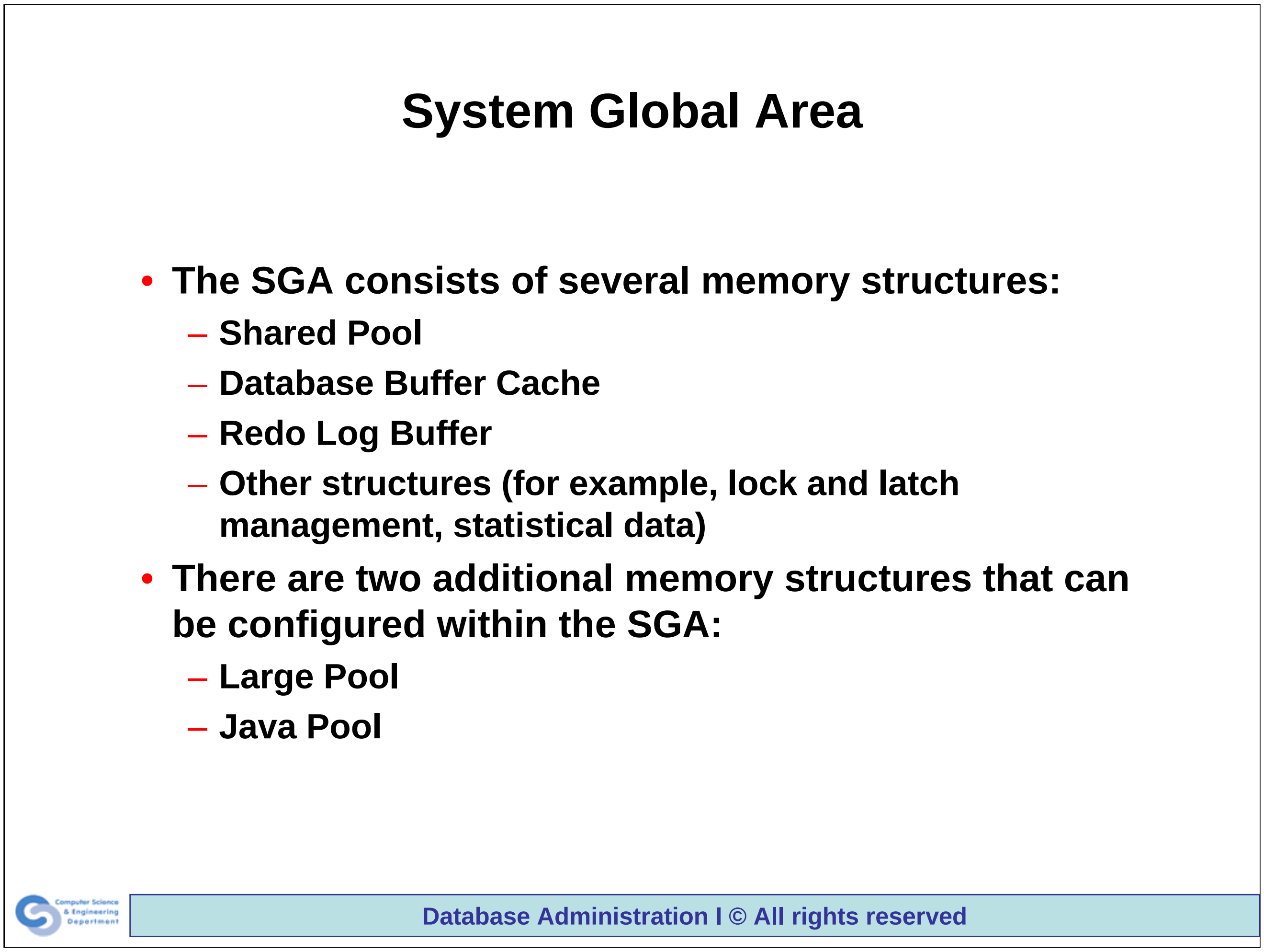
Physical Structure

The physical structure of an Oracle database includes three types of files: control files, data files, and online redo log files.

Memory Structure

Oracle’s memory structure consists of two memory areas known as:

- System Global Area (SGA): Allocated at instance start up, and is a fundamental component of an Oracle instance
- Program Global Area (PGA): Allocated when the server process is started



System Global Area

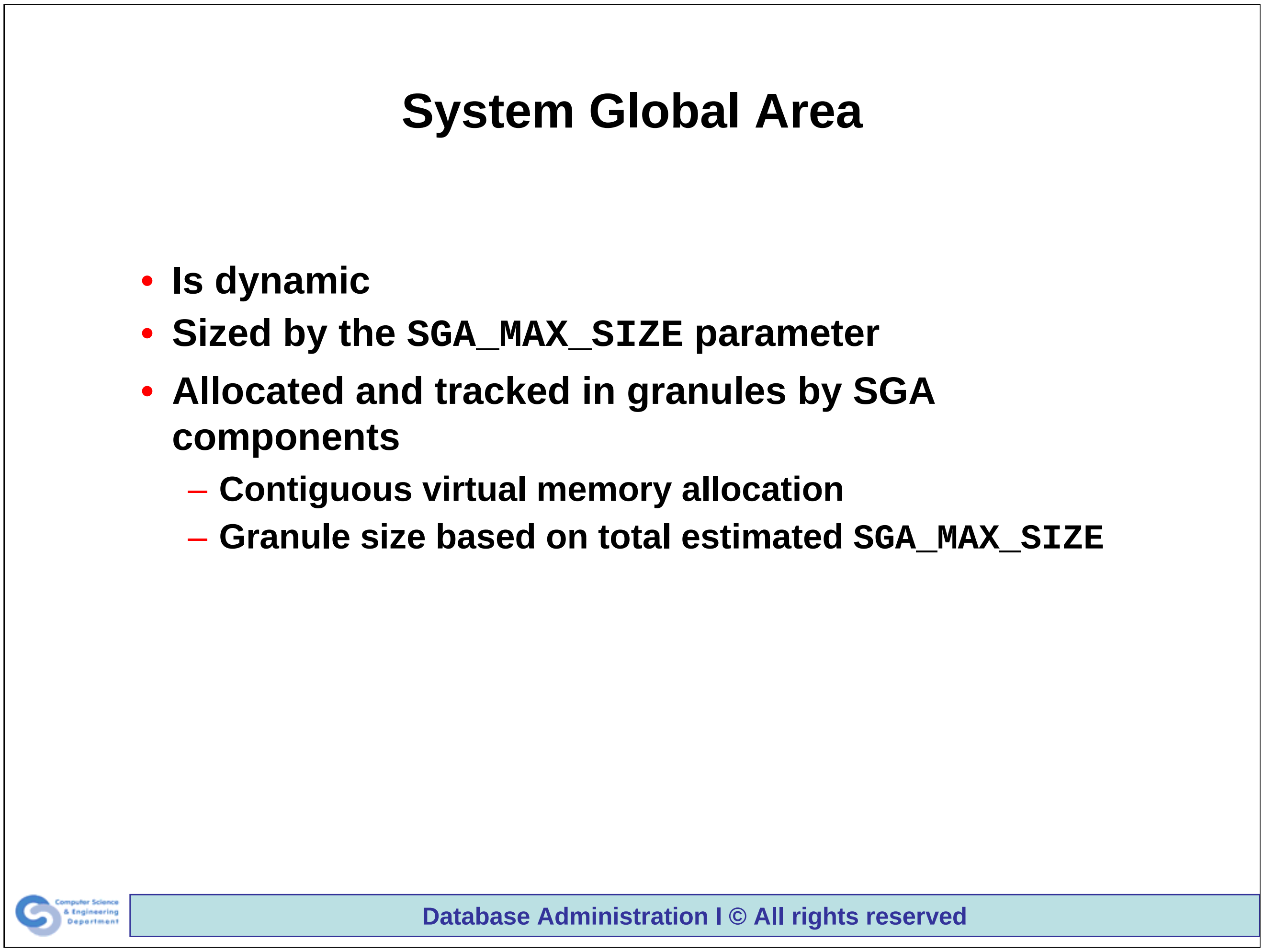
- The SGA consists of several memory structures:
  - Shared Pool
  - Database Buffer Cache
  - Redo Log Buffer
  - Other structures (for example, lock and latch management, statistical data)
- There are two additional memory structures that can be configured within the SGA:
  - Large Pool
  - Java Pool

System Global Area (SGA)

The SGA is also called the shared global area. It is used to store database information that is shared by database processes. It contains data and control information for the Oracle server and is allocated in the virtual memory of the computer where Oracle resides.

The following statement can be used to view SGA memory allocations:

```
SQL> SHOW SGA:
Total System Global Area      36437964  bytes
Fixed Size
          6543794   bytes
Variable Size                           19521536
          bytes
Database Buffers                        16777216
          bytes
Redo Buffers                             73728
          bytes
```



System Global Area

- Is dynamic
- Sized by the `SGA_MAX_SIZE` parameter
- Allocated and tracked in granules by SGA components
  - Contiguous virtual memory allocation
  - Granule size based on total estimated `SGA_MAX_SIZE`

Database Administration I © All rights reserved

System Global Area (continued)

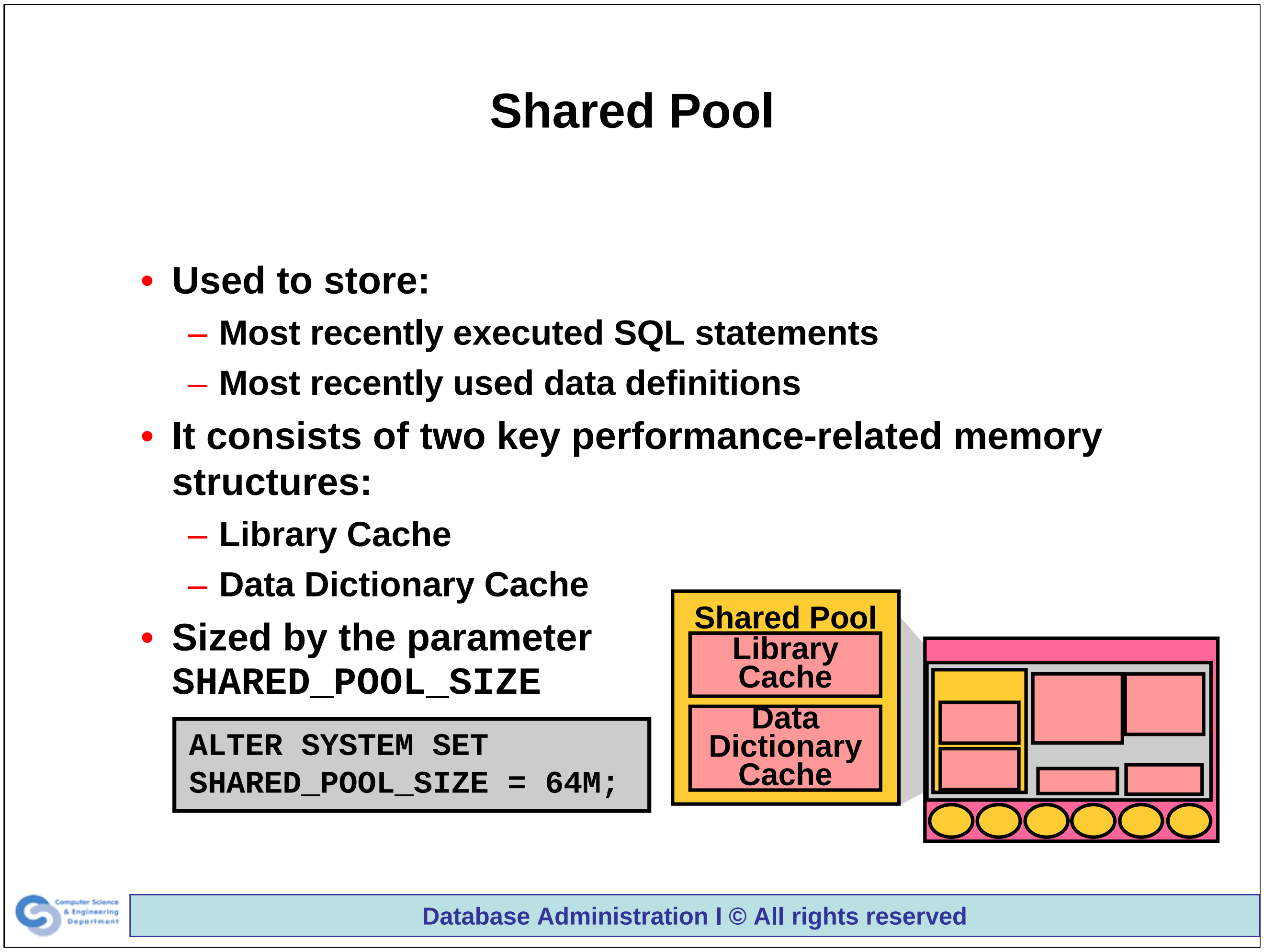
Unit of Allocation

A granule is a unit of contiguous virtual memory allocation. The size of a granule depends on the estimated total SGA size whose calculation is based on the value of the `SGA_MAX_SIZE` parameter.

4 MB if estimated SGA size is < 128 MB

16 MB otherwise

The components (Database Buffer Cache, Shared Pool, and Large Pool) are allowed to grow and shrink based on granule boundaries. At instance start up, the Oracle server allocates granule entries, one for each granule to support `SGA_MAX_SIZE` bytes of address space. As start up continues, each component acquires as many granules as it requires. The minimum SGA configuration is three granules (one granule for fixed SGA [includes redo buffers]; one granule for Database Buffer Cache; one granule for Shared Pool).



Shared Pool

- Used to store:
  - Most recently executed SQL statements
  - Most recently used data definitions
- It consists of two key performance-related memory structures:
  - Library Cache
  - Data Dictionary Cache
- Sized by the parameter `SHARED_POOL_SIZE`

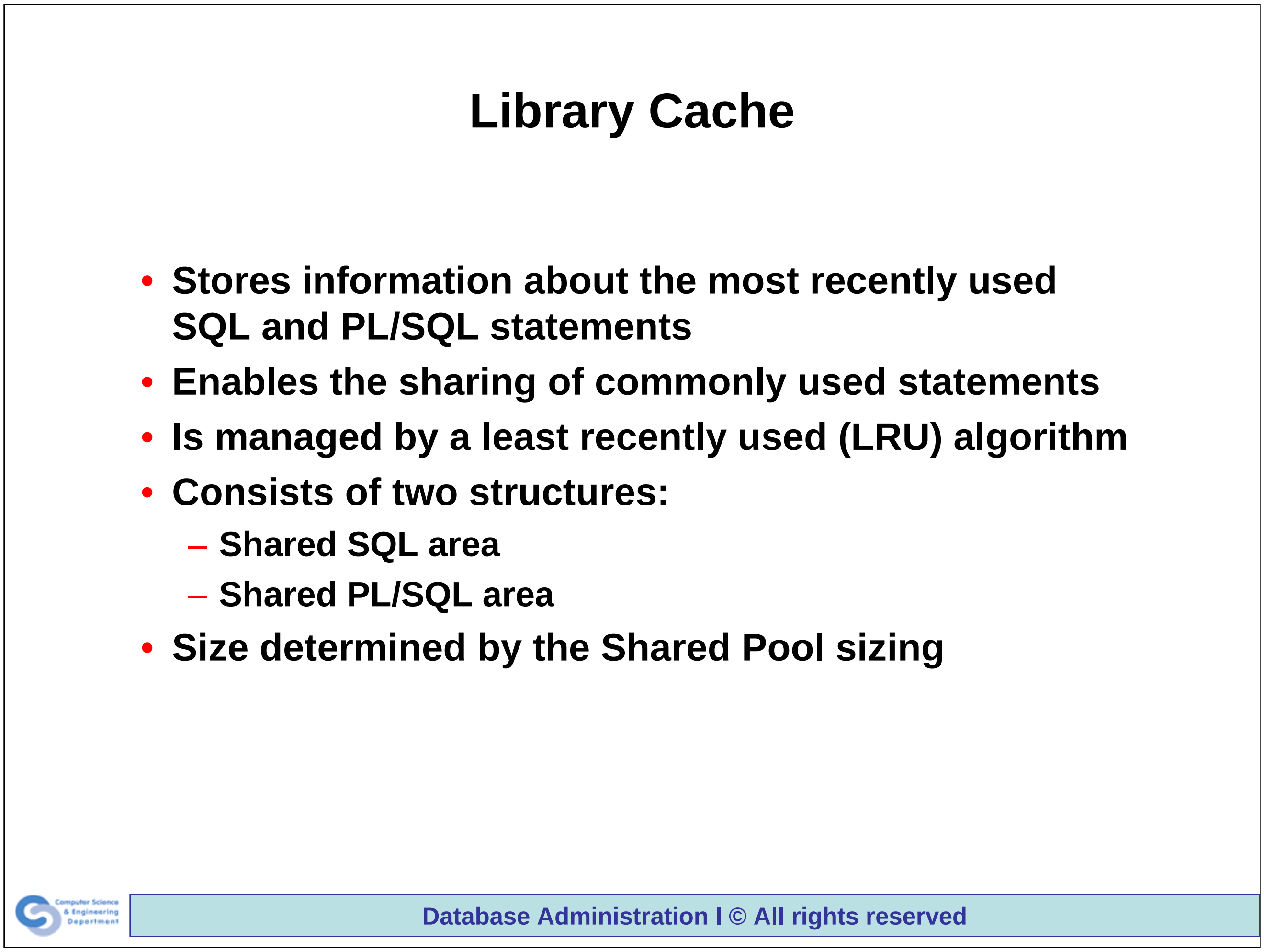
```
ALTER SYSTEM SET
SHARED_POOL_SIZE = 64M;
```

Shared Pool

The Shared Pool environment contains both fixed and variable structures. The fixed structures remain relatively the same size, whereas the variable structures grow and shrink based on user and program requirements. The actual sizing for the fixed and variable structures is based on an initialization parameter and the work of an Oracle internal algorithm.

Sizing the Shared Pool

Because the Shared Pool is used for objects that can be shared globally, such as reusable SQL execution plans, PL/SQL packages, procedures, functions, and cursor information, it must be sized to accommodate the needs of both the fixed and variable areas. Memory allocation for the Shared Pool is determined by the `SHARED_POOL_SIZE` initialization parameter. It can be dynamically resized using `ALTER SYSTEM SET`. After performance analysis, this can be adjusted but the total SGA size cannot exceed `SGA_MAX_SIZE`.



Library Cache

- Stores information about the most recently used SQL and PL/SQL statements
- Enables the sharing of commonly used statements
- Is managed by a least recently used (LRU) algorithm
- Consists of two structures:
  - Shared SQL area
  - Shared PL/SQL area
- Size determined by the Shared Pool sizing

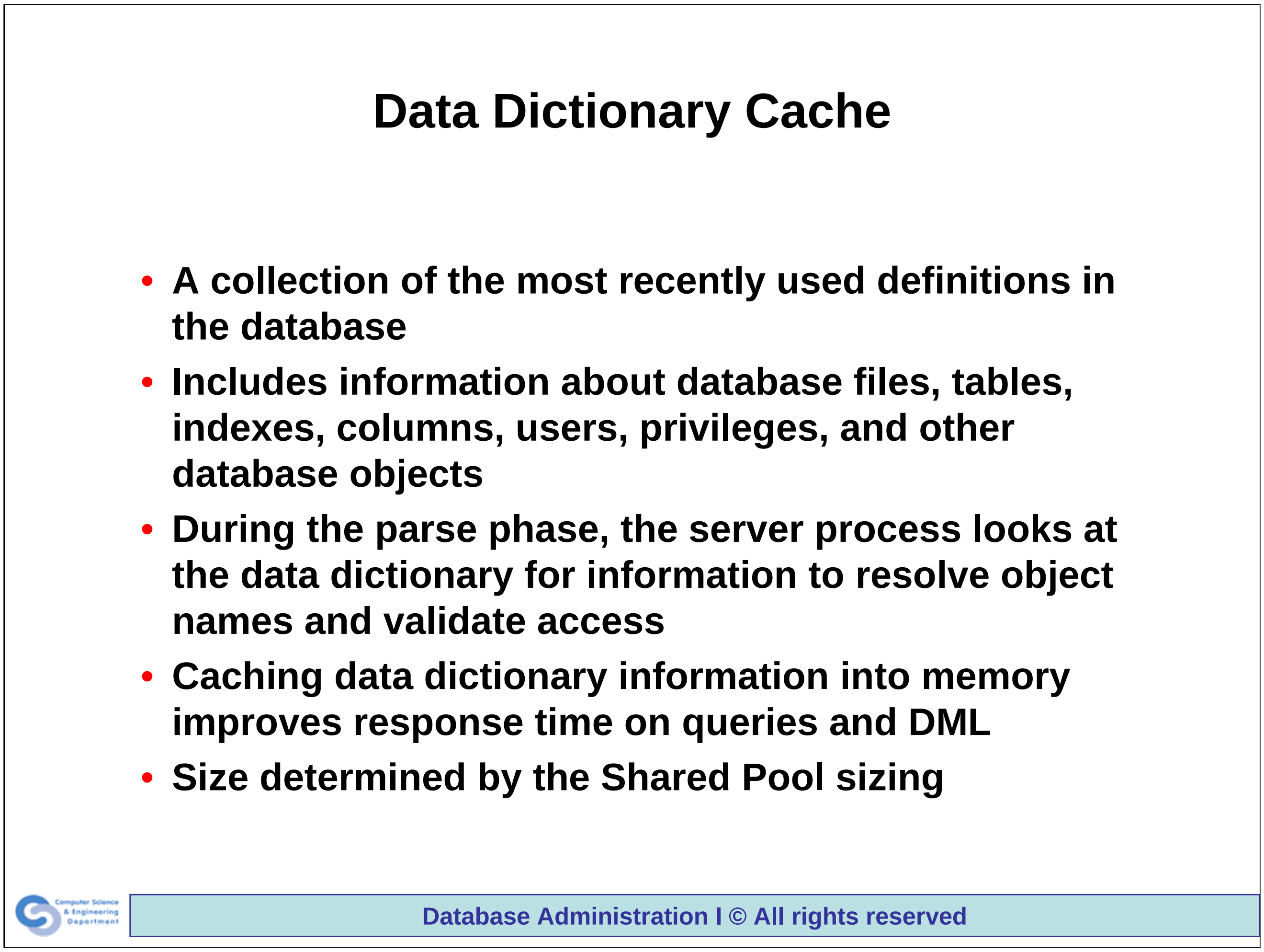
Library Cache

The Library Cache size is based on the sizing defined for the Shared Pool. Memory is allocated when a statement is parsed or a program unit is called. If the size of the Shared Pool is too small, statements are continually reloaded into the Library Cache, which affects performance. The Library Cache is managed by an LRU algorithm. As the cache fills, less recently used execution paths and parse trees are removed from the Library Cache to make room for the new entries. If the SQL or PL/SQL statements are not reused, they eventually are aged out.

The Library Cache consists of two structures:

Shared SQL: The Shared SQL stores and shares the execution plan and parse tree for SQL statements run against the database. The second time that an identical SQL statement is run, it is able to take advantage of the parse information available in the shared SQL to expedite its execution. To ensure that SQL statements use a shared SQL area whenever possible, the text, schema, and bind variables must be exactly the same.

Shared PL/SQL: The Shared PL/SQL area stores and shares the most recently executed PL/SQL statements. Parsed and compiled program units and procedures (functions, packages, and triggers) are stored in this area.



Data Dictionary Cache

- A collection of the most recently used definitions in the database
- Includes information about database files, tables, indexes, columns, users, privileges, and other database objects
- During the parse phase, the server process looks at the data dictionary for information to resolve object names and validate access
- Caching data dictionary information into memory improves response time on queries and DML
- Size determined by the Shared Pool sizing

Data Dictionary Cache

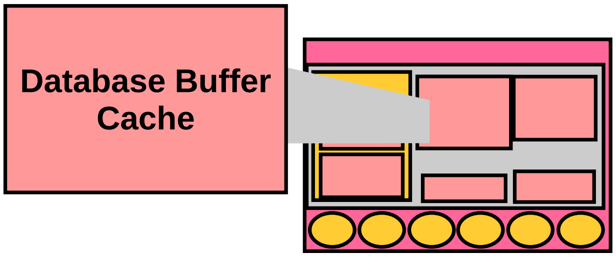
The Data Dictionary Cache is also referred to as the dictionary cache or row cache. Information about the database (user account data, data file names, segment names, extent locations, table descriptions, and user privileges) is stored in the data dictionary tables. When this information is needed by the server, the data dictionary tables are read, and the data that is returned is stored in the Data Dictionary Cache.

Sizing the Data Dictionary

The overall size is dependent on the size of the Shared Pool size and is managed internally by the database. If the Data Dictionary Cache is too small, then the database has to query the data dictionary tables repeatedly for information needed by the server. These queries are called recursive calls and are slower than the direct queries on the Data Dictionary Cache because direct queries do not use SQL.

Database Buffer Cache

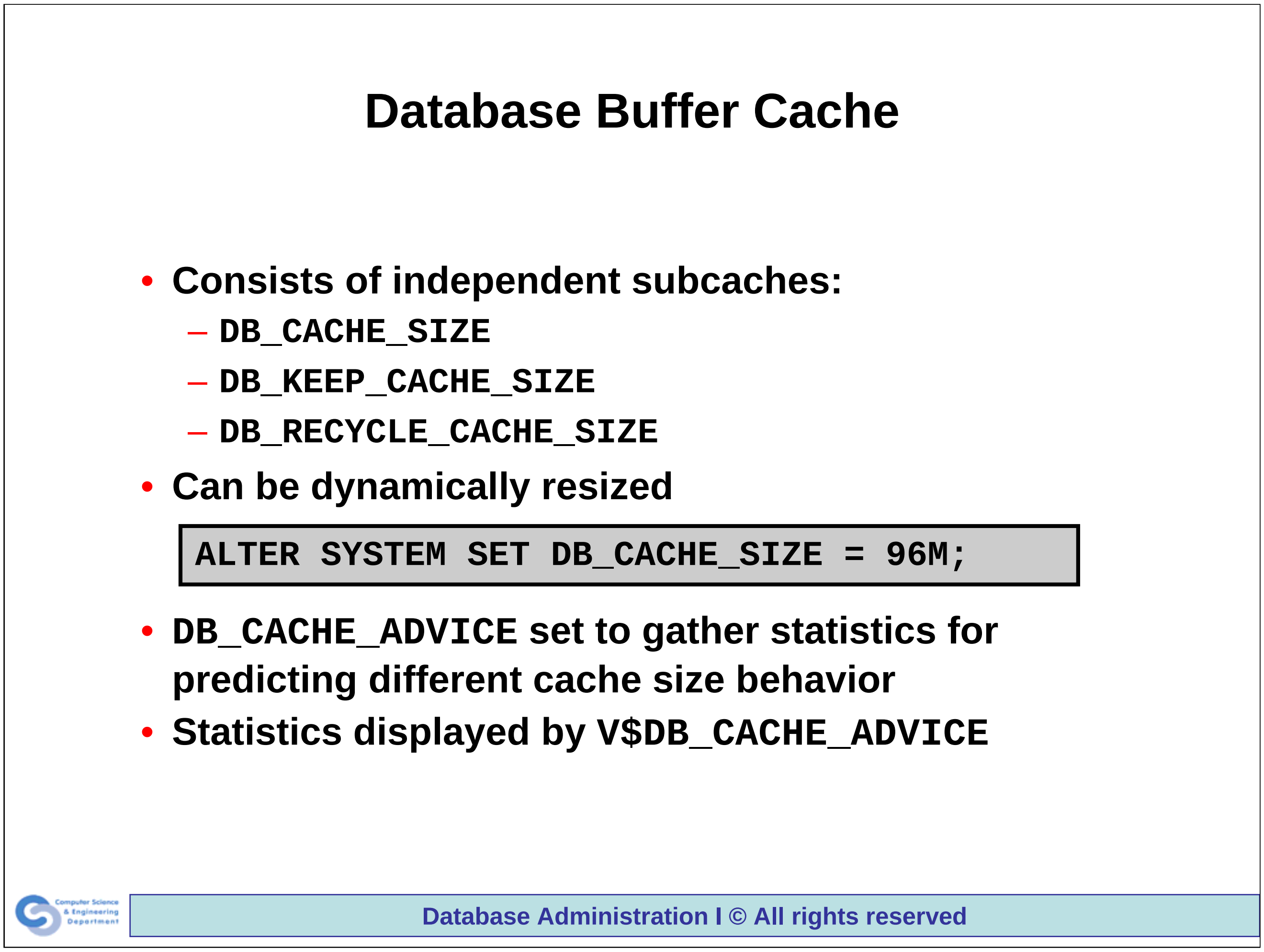
- Stores copies of data blocks that have been retrieved from the data files
- Enables great performance gains when you obtain and update data
- Managed through an LRU algorithm
- `DB_BLOCK_SIZE` determines primary block size



Database Buffer Cache

Database Buffer Cache

When a query is processed, the Oracle server process looks in the Database Buffer Cache for any blocks it needs. If the block is not found in the Database Buffer Cache, the server process reads the block from the datafile and places a copy in the Database Buffer Cache. Because subsequent requests for the same block may find the block in memory, the requests may not require physical reads. The Oracle server uses an LRU algorithm to age out buffers that have not been accessed recently to make room for new blocks in the Database Buffer Cache.



Database Buffer Cache

- Consists of independent subcaches:
  - DB_CACHE_SIZE
  - DB_KEEP_CACHE_SIZE
  - DB_RECYCLE_CACHE_SIZE
- Can be dynamically resized

```
ALTER SYSTEM SET DB_CACHE_SIZE = 96M;
```

- DB_CACHE_ADVICE set to gather statistics for predicting different cache size behavior
- Statistics displayed by V$DB_CACHE_ADVICE

Database Administration I © All rights reserved

Database Buffer Cache

Sizing the Database Buffer Cache

The size of each buffer in the Database Buffer Cache is equal to the size of an Oracle block, and it is specified by the DB_BLOCK_SIZE parameter. The Database Buffer Cache consists of independent subcaches for buffer pools and for multiple block sizes. The parameter DB_BLOCK_SIZE determines the primary block size, which is used for the SYSTEM tablespace.

Three parameters define the sizes of the Database Buffer Caches:

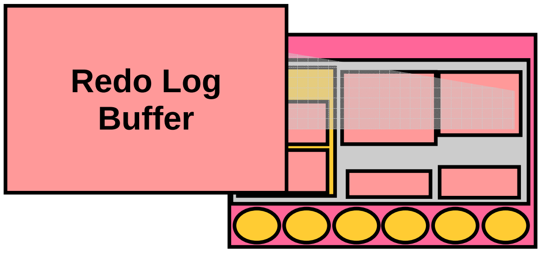
DB_CACHE_SIZE: Sizes the default buffer cache only; it always exists and cannot be set to zero

DB_KEEP_CACHE_SIZE: Sizes the keep buffer cache, which is used to retain blocks in memory that are likely to be reused

DB_RECYCLE_CACHE_SIZE: Sizes the recycle buffer cache, which is used to eliminate blocks from memory that have little chance of being reused

Redo Log Buffer

- Records all changes made to the database data blocks
- Primary purpose is recovery
- Changes recorded within are called redo entries
- Redo entries contain information to reconstruct or redo changes
- Size defined by `LOG_BUFFER`

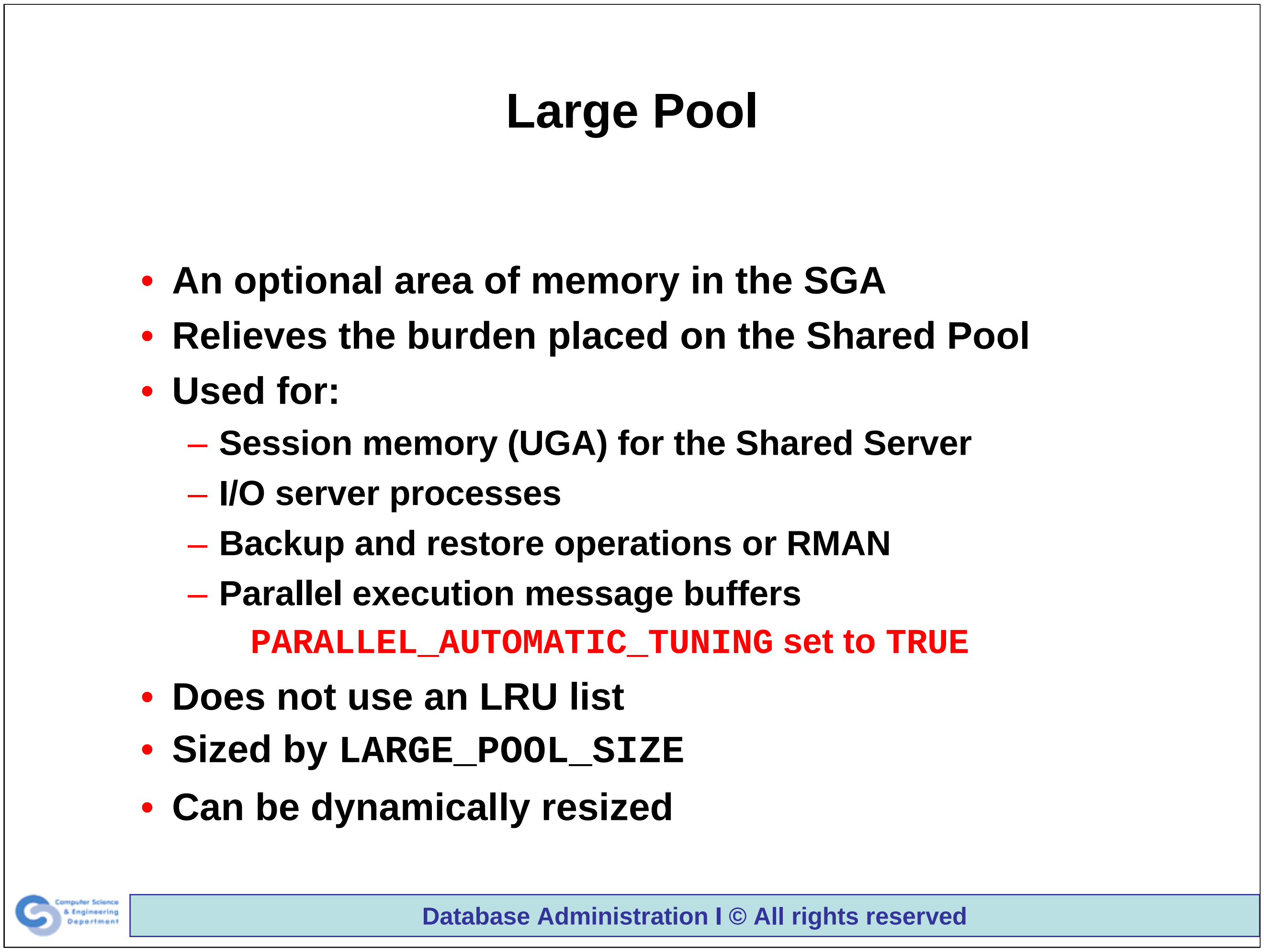


Redo Log Buffer

The Redo Log Buffer is a circular buffer that contains changes made to datafile blocks. This information is stored in redo entries. Redo entries contain the information necessary to re-create the data prior to the change made by `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, or `DROP` operations.

Sizing the Redo Log Buffer

The size of the Redo Log Buffer is defined by the `LOG_BUFFER` initialization parameter.



Large Pool

- An optional area of memory in the SGA
- Relieves the burden placed on the Shared Pool
- Used for:
  - Session memory (UGA) for the Shared Server
  - I/O server processes
  - Backup and restore operations or RMAN
  - Parallel execution message buffers

    `PARALLEL_AUTOMATIC_TUNING` set to `TRUE`
- Does not use an LRU list
- Sized by `LARGE_POOL_SIZE`
- Can be dynamically resized

Large Pool

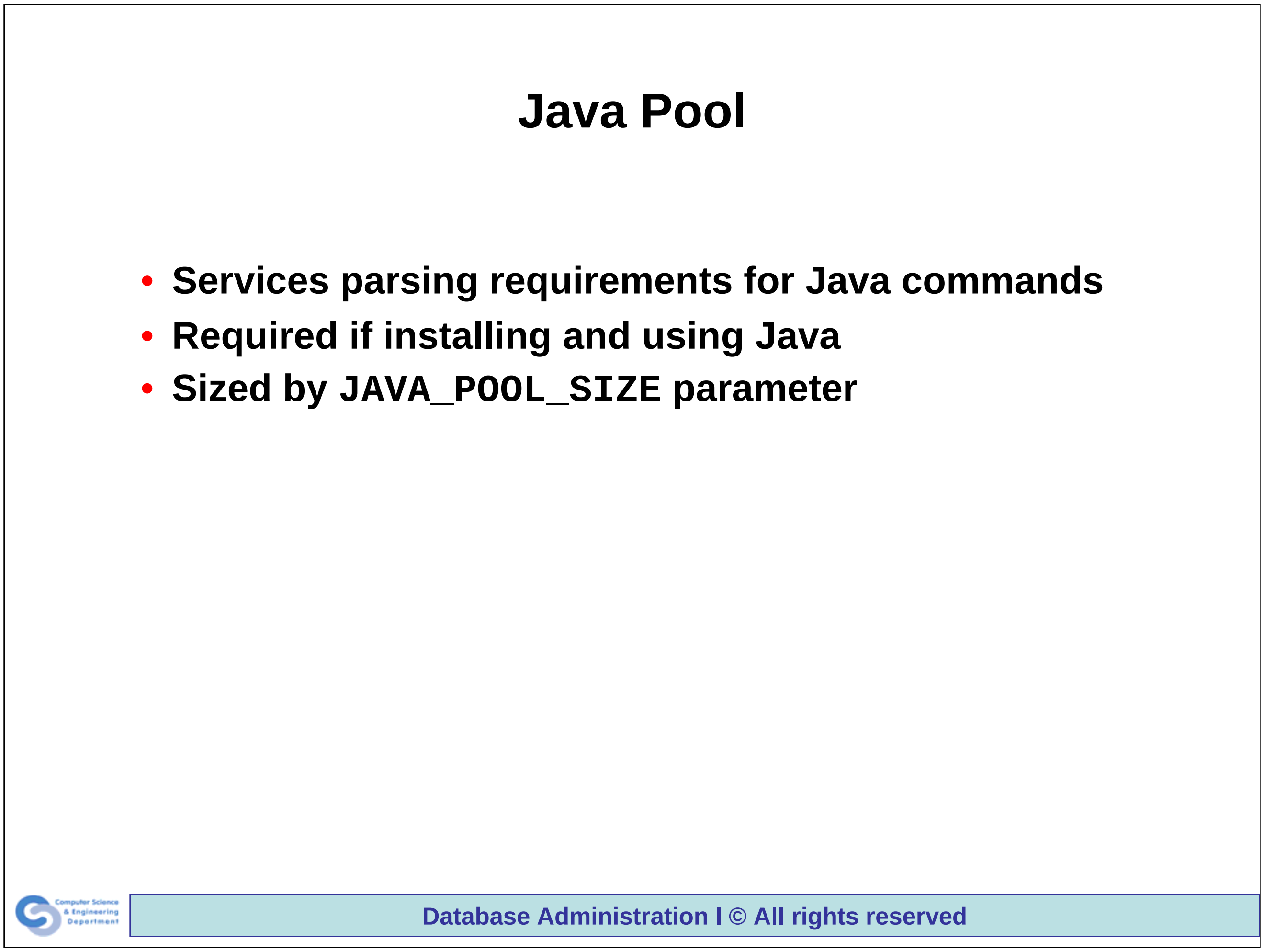
By allocating session memory from the Large Pool for Shared Server, Oracle XA, or parallel query buffers, Oracle can use the Shared Pool primarily for caching Shared SQL statements. Thus relieving the burden on areas within the Shared Pool. The Shared Pool does not have to give up memory for caching SQL parse trees in favor of Shared Server session information, I/O, and backup and recover processes. The performance gain is from the reduction of overhead from increasing and shrinkage of the shared SQL cache.

Backup and Restore

Recovery Manager (RMAN) uses the Large Pool when the `BACKUP_DISK_IO=`*n* and `BACKUP_TAPE_IO_SLAVE=TRUE` parameters are set. If the Large Pool is configured but is not large enough, the allocation of memory from the Large Pool fails. RMAN writes an error message to the alert log file and does not use I/O slaves for backup or restore.

Parallel Execution

The Large Pool is used if `PARALLEL_AUTOMATIC_TUNING` is set to `TRUE`, otherwise, these buffers are allocated to the Shared Pool.

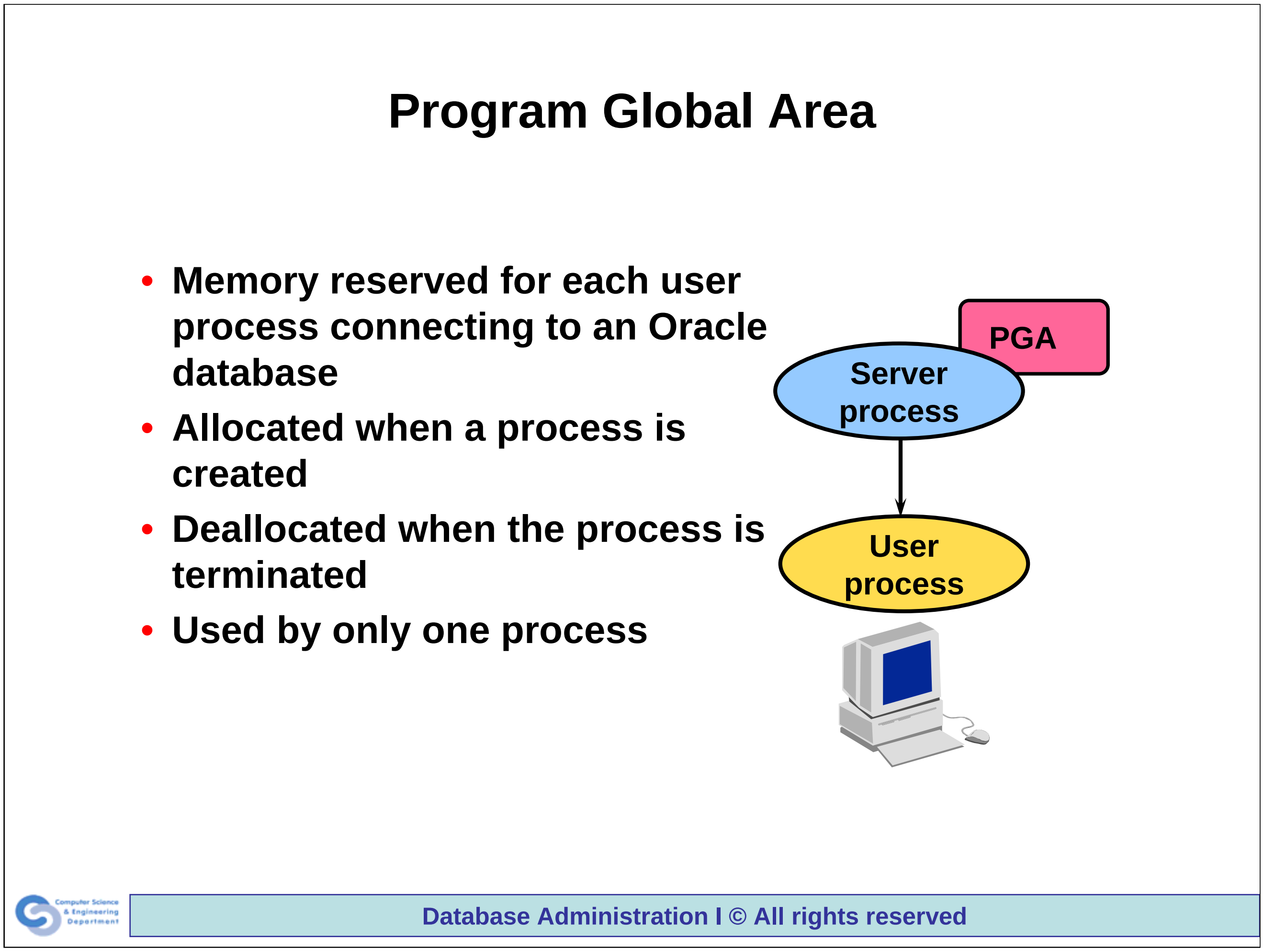


Java Pool

- Services parsing requirements for Java commands
- Required if installing and using Java
- Sized by `JAVA_POOL_SIZE` parameter

Java Pool

The Java Pool is an optional setting but is required if you are installing and using Java. Its size is set, in bytes, using the `JAVA_POOL_SIZE` parameter. In Oracle9*i*, the default size of the Java Pool is 24 MB.



Program Global Area

- Memory reserved for each user process connecting to an Oracle database
- Allocated when a process is created
- Deallocated when the process is terminated
- Used by only one process

PGA

Server process

User process

Program Global Area (PGA)

The Program Global Area or Process Global Area (PGA) is a memory region that contains data and control information for a single server process or a single background process. The PGA is allocated when a process is created and deallocated when the process is terminated. In contrast to the SGA, which is shared by several processes, the PGA is an area that is used by only one process.

Contents of PGA

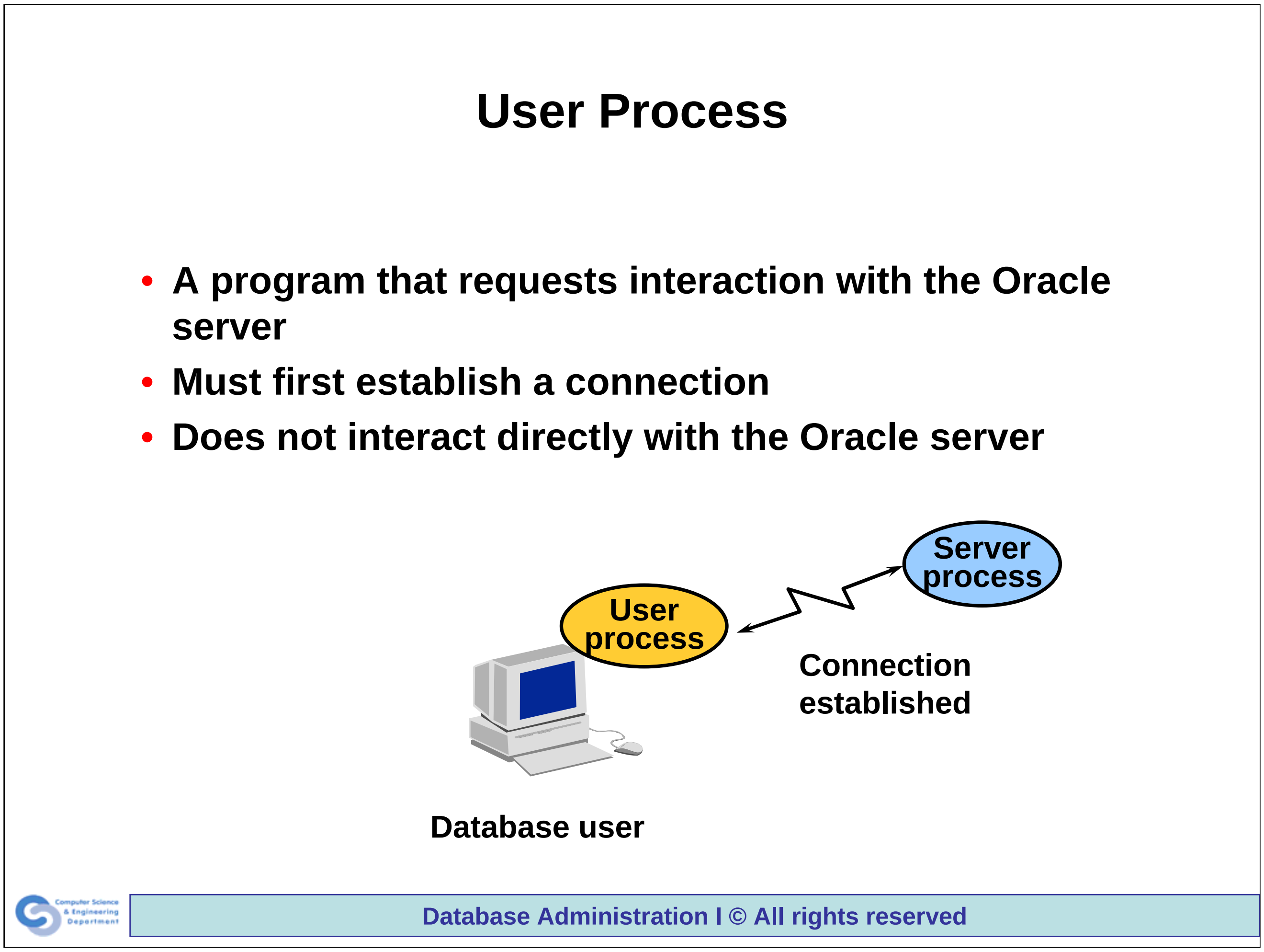
The contents of the PGA memory varies, depending whether the instance is running in a dedicated server or shared server configuration. Generally the PGA memory includes these components:

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 divided into two areas:

Process Structure

Oracle takes advantage of various types of processes:

- User process: Started at the time a database user requests connection to the Oracle server
- Server process: Connects to the Oracle instance and is started when a user establishes a session
- Background processes: Started when an Oracle instance is started



User Process

- A program that requests interaction with the Oracle server
- Must first establish a connection
- Does not interact directly with the Oracle server

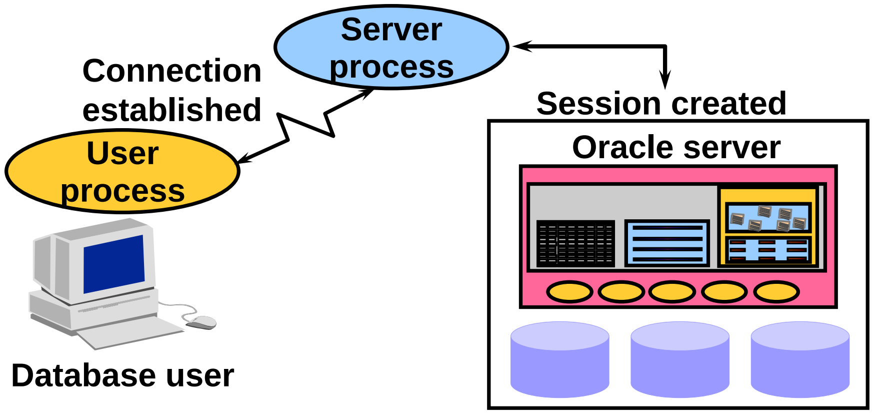
Server process
User process
Connection established
Database user

User Process

A database user who needs to request information from the database must first make a connection with the Oracle server. The connection is requested using a database interface tool, such as SQL*Plus, and beginning the user process. The user process does not interact directly with the Oracle server. Rather it generates calls through the user program interface (UPI), which creates a session and starts a server process.

Server Process

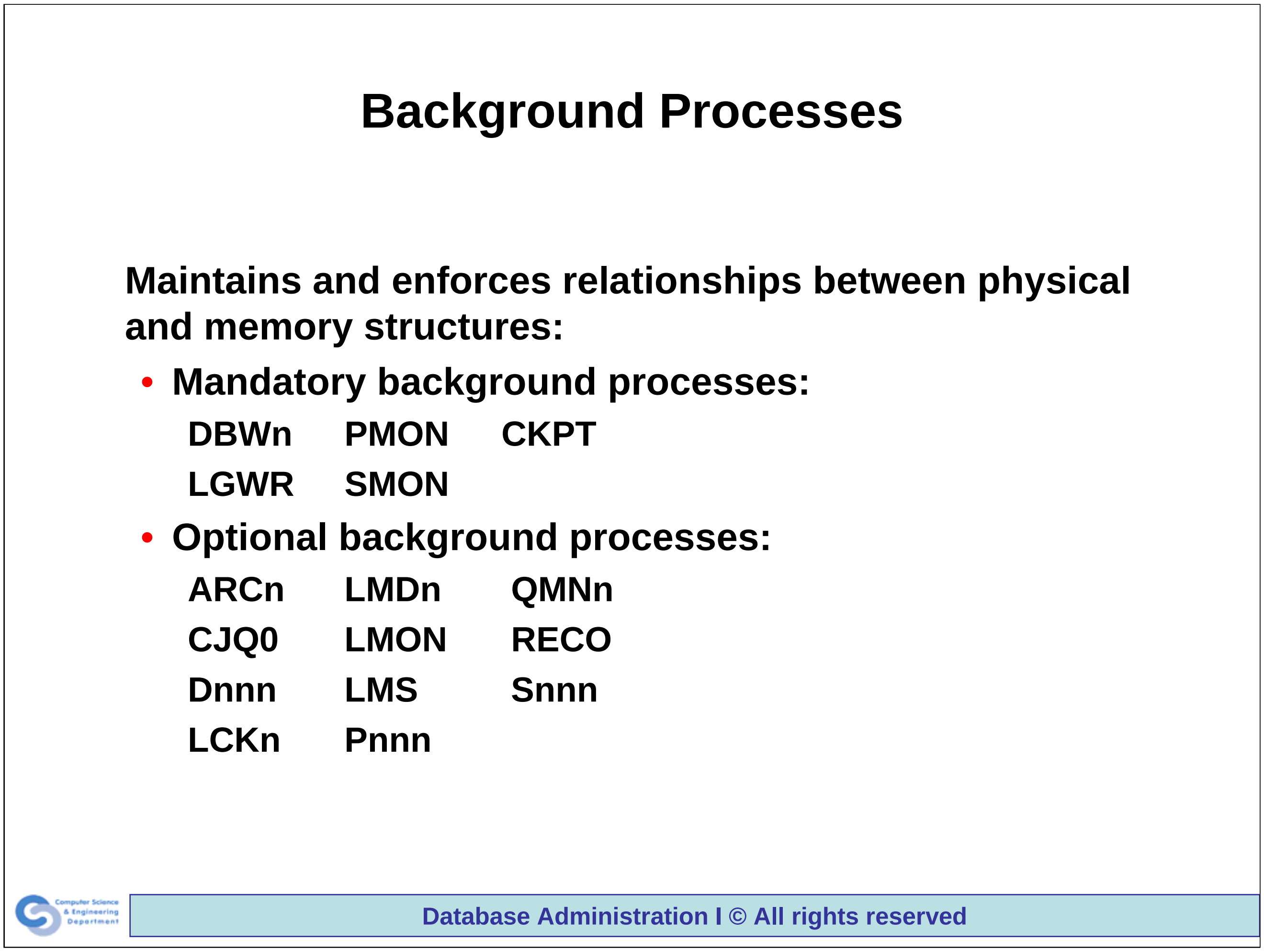
- A program that directly interacts with the Oracle server
- Fulfills calls generated and returns results
- Can be dedicated or shared server



Server Process

Once a user has established a connection, a server process is started to handle the user processes requests. A server process can be either a dedicated server process or a shared server process. In a dedicated server environment, the server process handles the request of a single user process. Once a user process disconnects, the server process is terminated. In a shared server environment, the server process handles the request of several user processes. The server process communicates with the Oracle server using the Oracle Program Interface (OPI).

Note: Allocation of server process in a dedicated environment versus a shared environment is covered in further detail in the *Oracle9i Database Performance Tuning* course.



Background Processes

Maintains and enforces relationships between physical and memory structures:

- **Mandatory background processes:**

  DBWn
  LGWR
  PMON
  SMON
  CKPT

- **Optional background processes:**

  ARCn
  CJQ0
  Dnnn
  LCKn
  LMDn
  LMON
  LMS
  Pnnn
  QMNn
  RECO
  Snnn

Background Processes

The Oracle architecture has five mandatory background processes that are discussed further in this lesson. In addition to the mandatory list, Oracle has many optional background process that are started when their option is being used. These optional processes are not within the scope of this course, with the exception of the background process, ARCn. Following is a list of some optional background processes:

ARCn: Archiver

CJQ0: Coordinator Job Queue background process

Dnnn: Dispatcher

LCKn: RAC Lock Manager–Instance Locks

LMDn: RAC DLM Monitor–Remote Locks

LMON: RAC DLM Monitor–Global Locks

LMS: RAC Global Cache Service

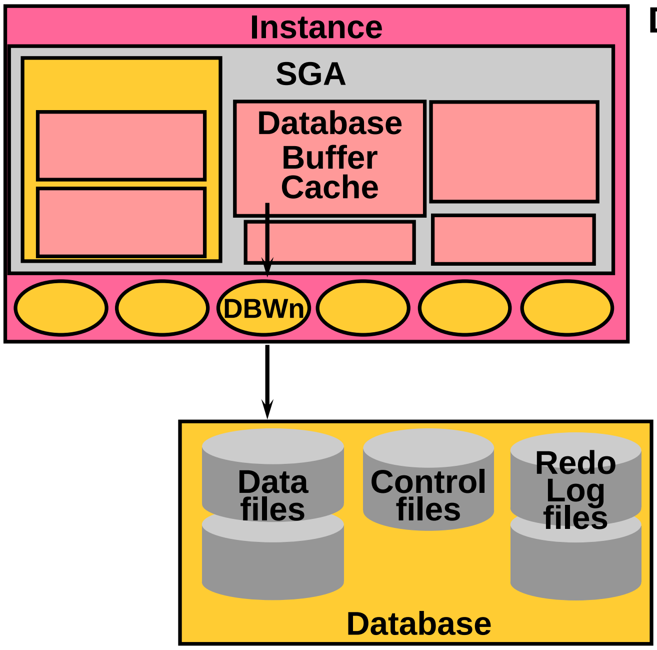
Pnnn: Parallel Query Slaves

QMNn: Advanced Queuing

RECO: Recoverer

Snnn: Shared Server

Database Writer (DBWn)



DBWn writes when:

- Checkpoint occurs
- Dirty buffers reach threshold
- There are no free buffers
- Timeout occurs
- RAC ping request is made
- Tablespace `OFFLINE`
- Tablespace `READ ONLY`
- Table `DROP` or `TRUNCATE`
- Tablespace `BEGIN BACKUP`

Database Writer (DBWn)

The server process records changes to undo and data blocks in the Database Buffer Cache. DBWn writes the dirty buffers from the Database Buffer Cache to the data files. It ensures that a sufficient number of free buffers (buffers that can be overwritten when server processes need to read in blocks from the data files) are available in the Database Buffer Cache. Database performance is improved because server processes make changes only in the Database Buffer Cache.

DBWn defers writing to the data files until one of the following events occurs:

Incremental or normal checkpoint

The number of dirty buffers reaches a threshold value

A process scans a specified number of blocks when scanning for free buffers and cannot find any

Timeout occurs

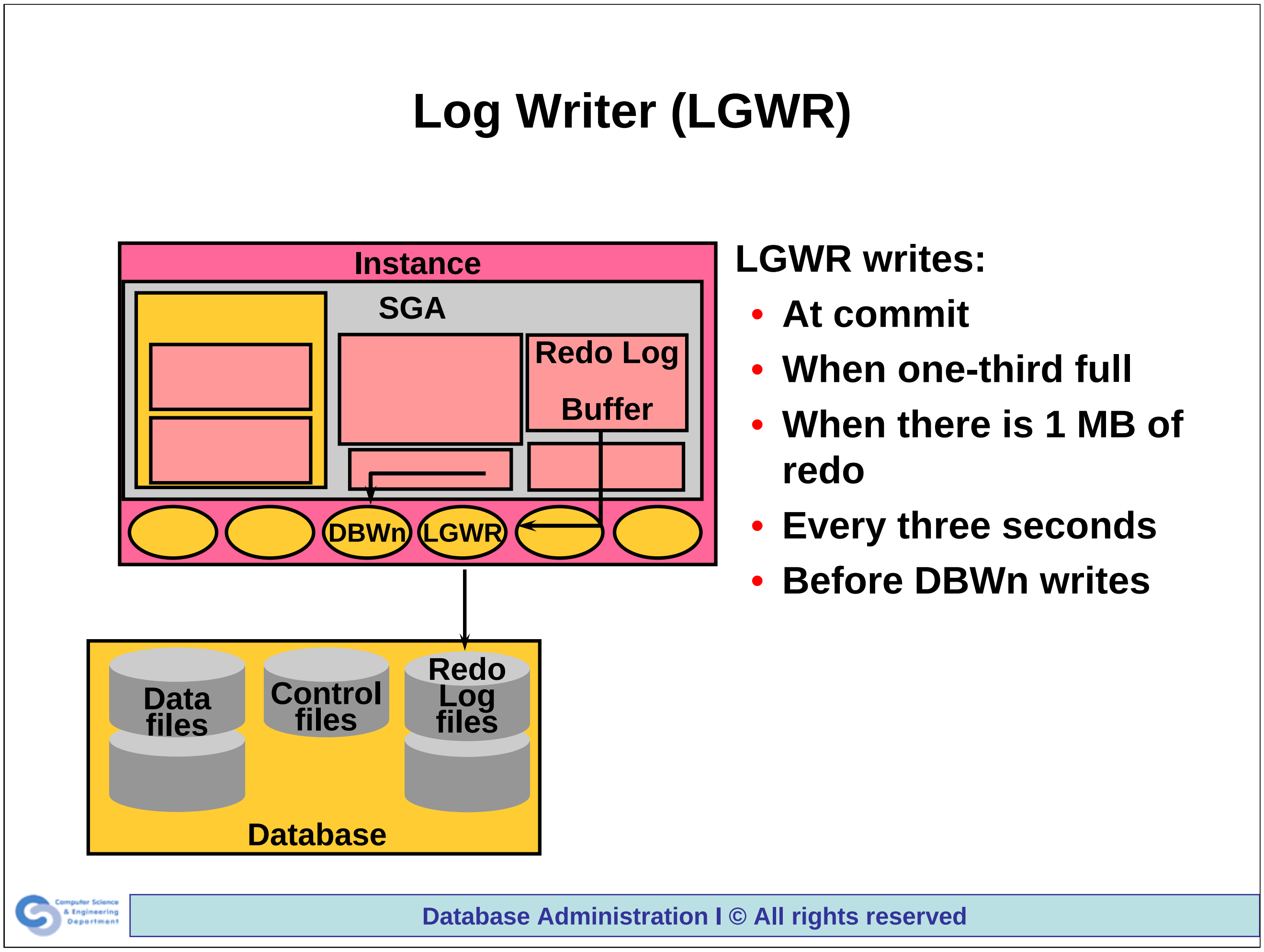
A ping request in Real Application Clusters (RAC) environment

Placing a normal or temporary tablespace offline

Placing a tablespace in read-only mode

Dropping or truncating a table

`ALTER TABLESPACE` tablespace name `BEGIN BACKUP`



Log Writer (LGWR)

LGWR writes:

- **At commit**
- **When one-third full**
- **When there is 1 MB of redo**
- **Every three seconds**
- **Before DBWn writes**

Database Administration I © All rights reserved

Log Writer (LGWR)

LGWR performs sequential writes from the Redo Log Buffer to the online redo log file under the following situations:

When a transaction commits

When the Redo Log Buffer is one-third full

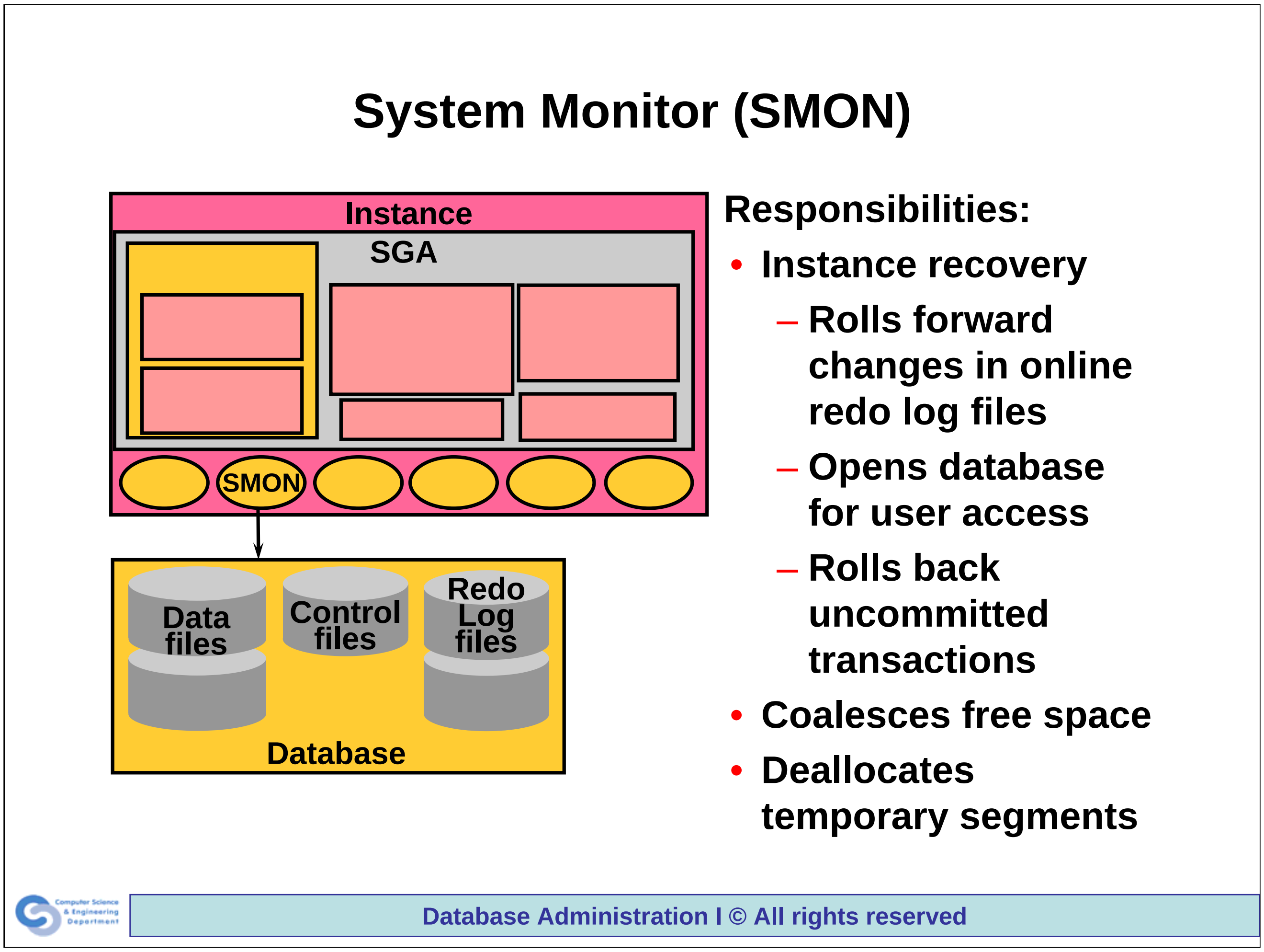
When there is more than 1 MB of changes recorded in the Redo Log Buffer

Before DBWn writes modified blocks in the Database Buffer Cache to the data files

Every three seconds

Because the redo is needed for recovery, LGWR confirms the commit operation only after the redo is written to disk.

LGWR can also call on DBWn to write to the data files.



System Monitor (SMON)

Responsibilities:

- Instance recovery
  - Rolls forward changes in online redo log files
  - Opens database for user access
  - Rolls back uncommitted transactions
- Coalesces free space
- Deallocates temporary segments

Database Administration I © All rights reserved

System Monitor (SMON)

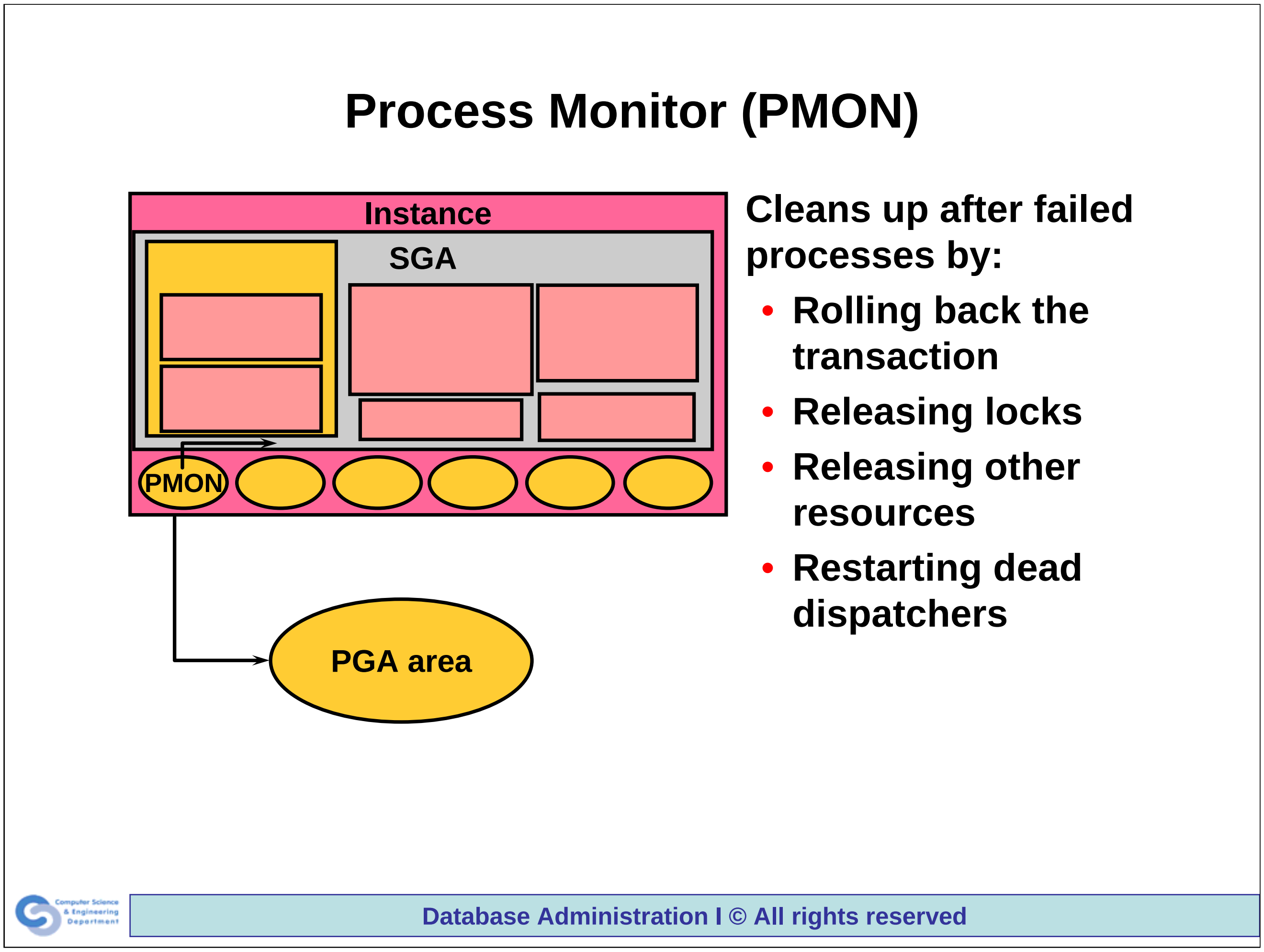
If the Oracle instance fails, any information in the SGA that has not been written to disk is lost. For example, the failure of the operating system causes an instance failure. After the loss of the instance, the background process SMON automatically performs instance recovery when the database is reopened. Instance recovery consists of the following steps:

1. Rolling forward to recover data that has not been recorded in the data files but that has been recorded in the online redo log file. This data has not been written to disk because of the loss of the SGA during instance failure. During this process, SMON reads the online redo log files and applies the changes recorded in the online redo log file to the data blocks. Because all committed transactions have been written to the online redo log files, this process completely recovers these transactions.

2. Opening the database so that users can log on. Any data that is not locked by unrecovered transactions is immediately available.

3. Rolling back uncommitted transactions. They are rolled back by SMON or by the individual server processes as they access locked data.

SMON also performs some space maintenance functions:



Process Monitor (PMON)

Cleans up after failed processes by:

- Rolling back the transaction
- Releasing locks
- Releasing other resources
- Restarting dead dispatchers

Process Monitor (PMON)

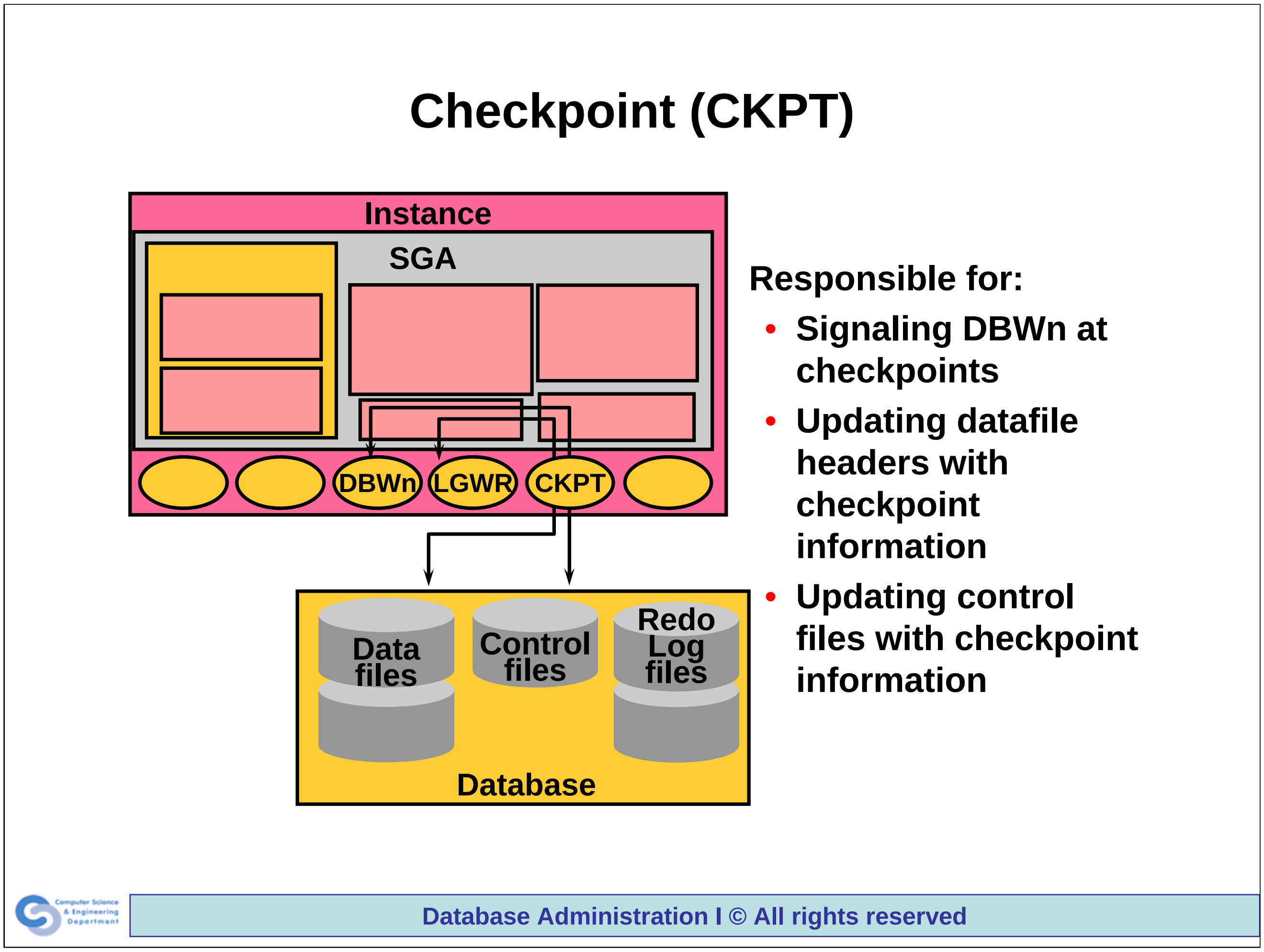
The background process PMON cleans up after failed processes by:

Rolling back the user’s current transaction

Releasing all currently held table or row locks

Freeing other resources currently reserved by the user

Restarts dead dispatchers



Checkpoint (CKPT)

Responsible for:

- Signaling DBWn at checkpoints
- Updating datafile headers with checkpoint information
- Updating control files with checkpoint information

Checkpoint (CKPT)

Every three seconds the CKPT process stores data in the control file to identify that place in the online redo log file where recovery is to begin, which is called a checkpoint. The purpose of a checkpoint is to ensure that all of the buffers in the Database Buffer Cache that were modified prior to a point in time have been written to the data files. This point in time (called the checkpoint position) is where database recovery is to begin in the event of an instance failure. DBWn will already have written all of the buffers in the Database Buffer Cache that were modified prior to that point in time. Prior to Oracle9*i*, this was done at the end of the online redo log file. In the event of a log switch CKPT also writes this checkpoint information to the headers of the data files.

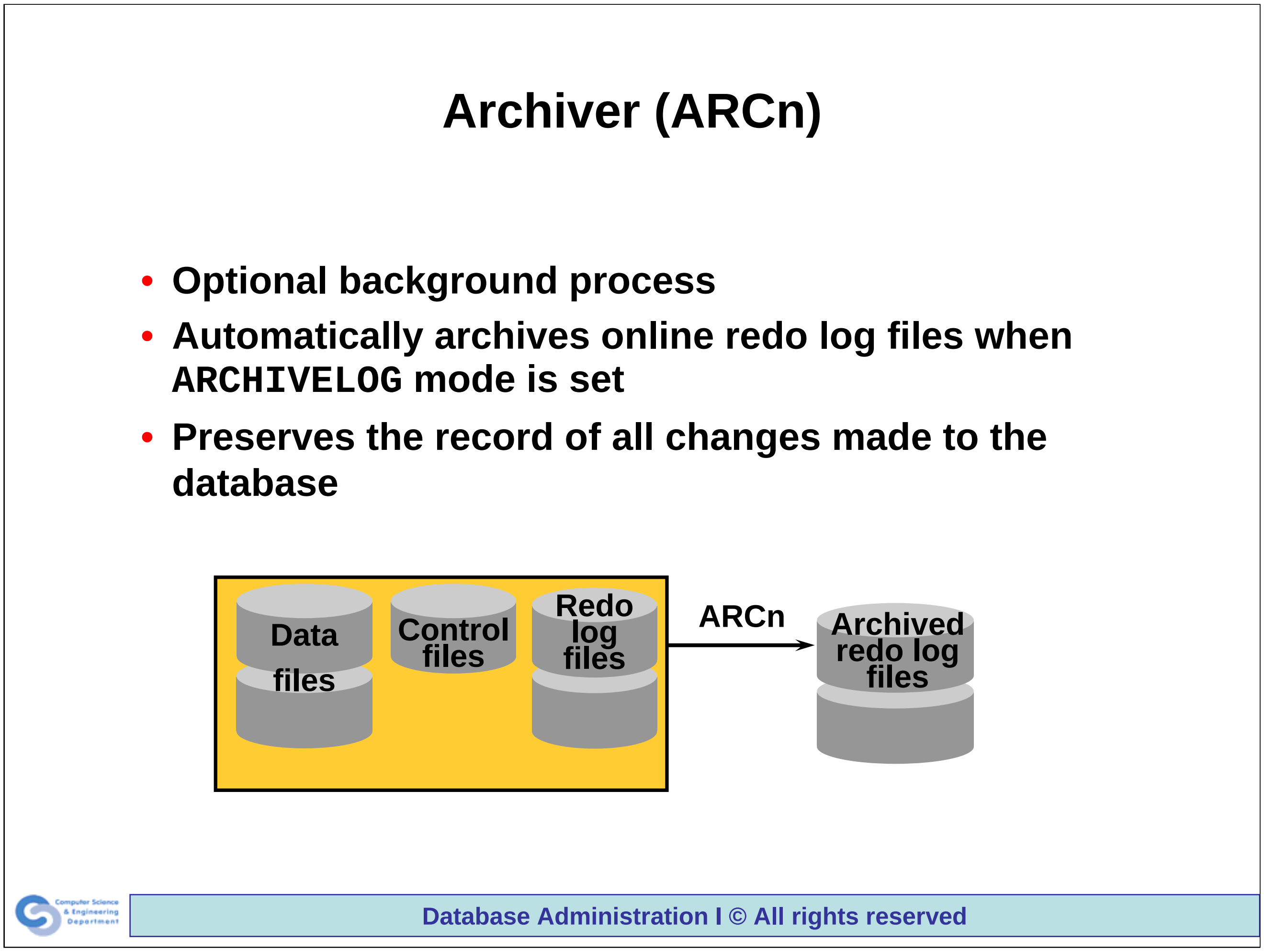
Checkpoints are initiated for the following reasons:

To ensure that modified data blocks in memory are written to disk regularly so that data is not lost in case of a system or database failure.

To reduce the time required for instance recovery. Only the online redo log file entries following the last checkpoint need to be processed for recovery to occur.

To ensure that all committed data has been written to the data files during shut down.

Checkpoint information written by CKPT includes checkpoint position, system change number, location in the online redo log file to begin recovery, information about logs, and so on.



Archiver (ARCn)

- Optional background process
- Automatically archives online redo log files when `ARCHIVELOG` mode is set
- Preserves the record of all changes made to the database

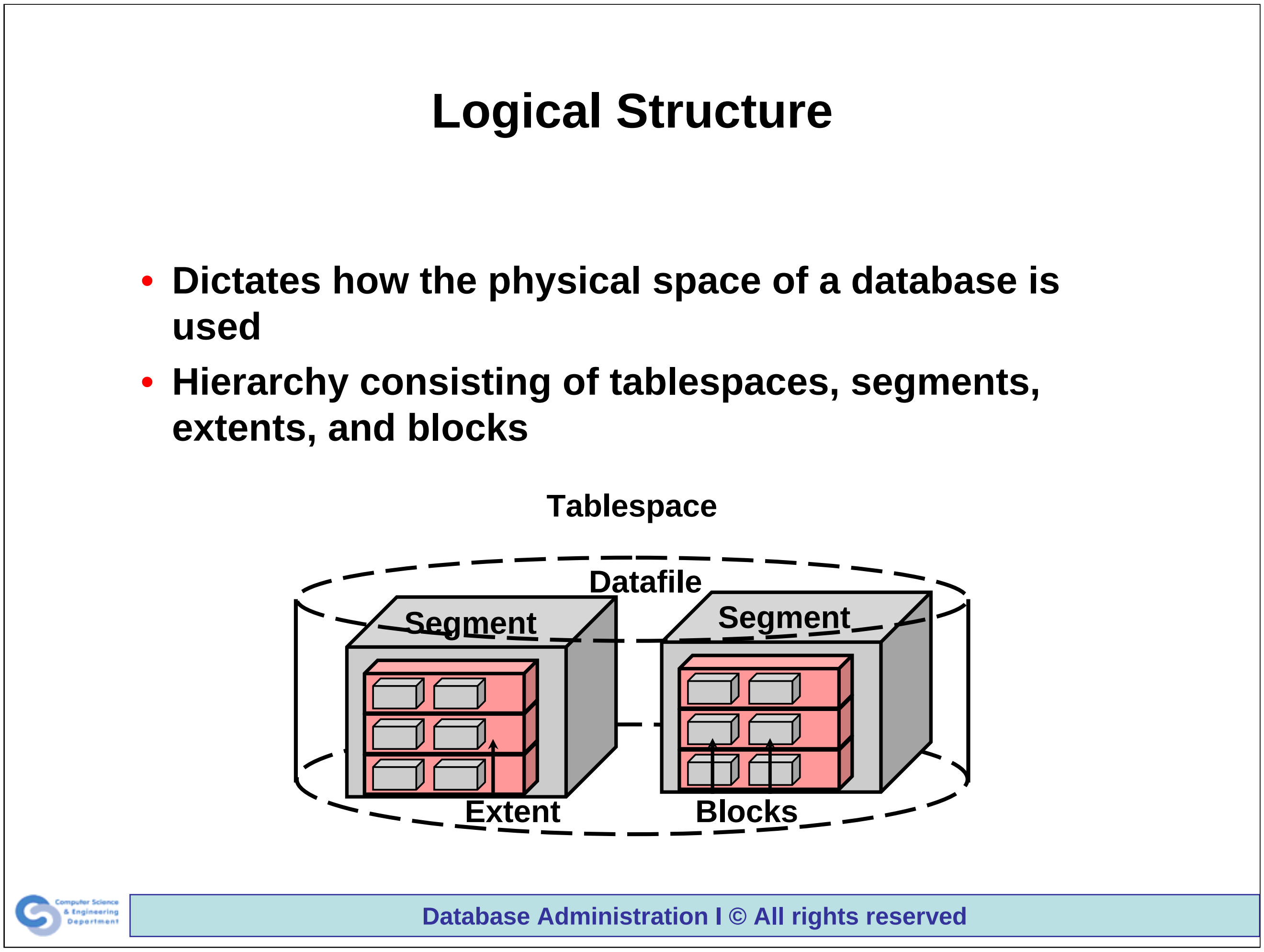
Archiver (ARCn)

ARCn is an optional background process, however, it is crucial to recovering a database after the loss of a disk. As online redo log files get filled, the Oracle server begins writing to the next online redo log file. The process of switching from one online redo log file to another is called a log switch. The ARCn process initiates backing up, or archiving, of the filled log group at every log switch. It automatically archives the online redo log file before the log can be reused, so all of the changes made to the database are preserved. This enables recovery of the database to the point of failure even if a disk drive is damaged.

Archiving Online Redo Log Files

One of the important decisions that a DBA has to make is whether to configure the database to operate in `ARCHIVELOG` or in `NOARCHIVELOG` mode.

`NOARCHIVELOG` mode: In `NOARCHIVELOG` mode, the online redo log files are overwritten each time a log switch occurs. LGWR does not overwrite an online redo log file group until the checkpoint for that group is complete. This ensures that committed data can be recovered if there is an instance crash. During the instance crash, only the SGA is lost. There is no loss of disks, only memory. For example, an operating system crash causes an instance crash.



Logical Structure

- Dictates how the physical space of a database is used
- Hierarchy consisting of tablespaces, segments, extents, and blocks

Logical Structure

A logical structure hierarchy exists as follows:

An Oracle database contains at least one tablespace.

A tablespace contains one or more segments.

A segment is made up of extents.

An extent is made up of logical blocks.

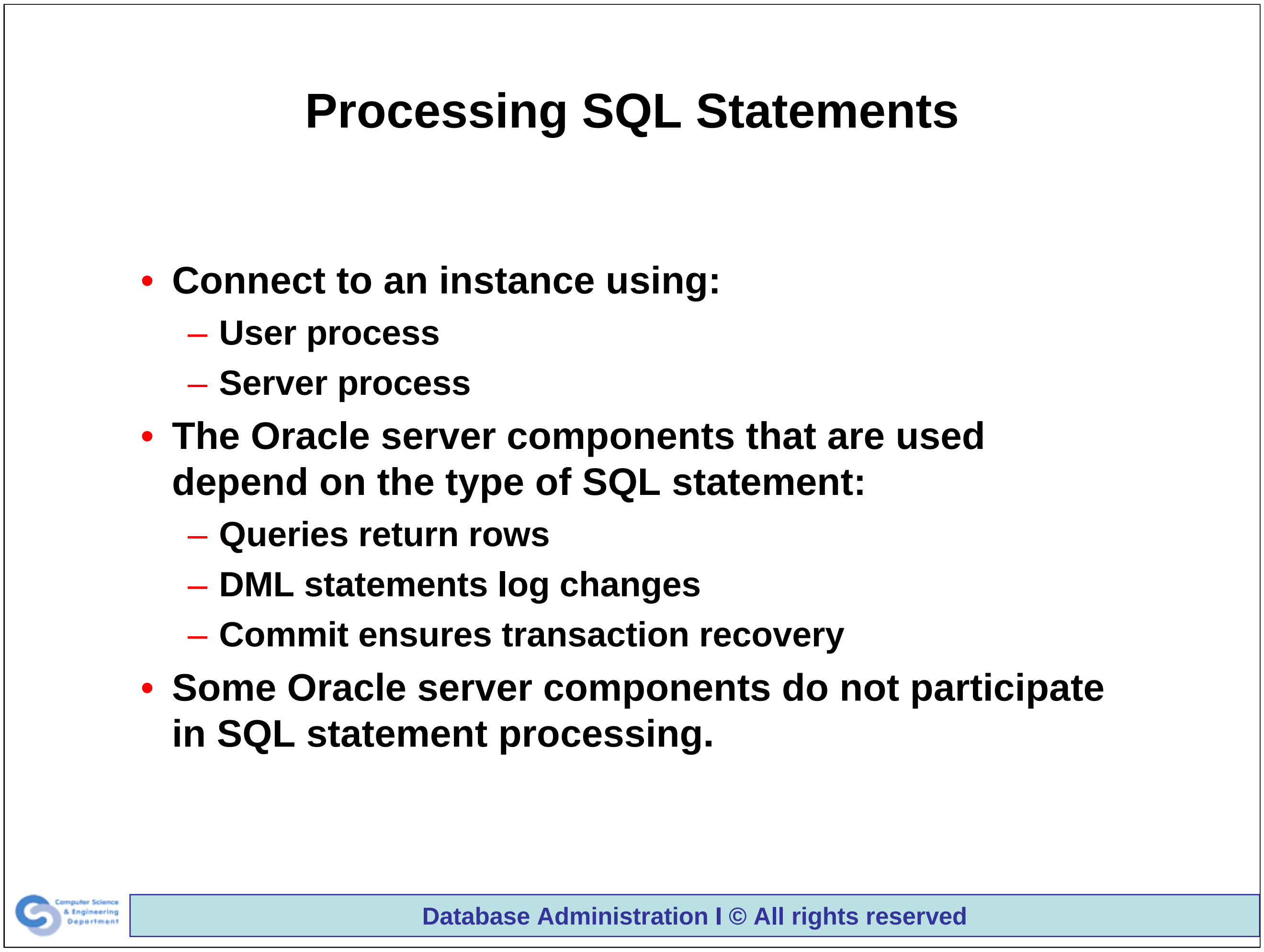
A block is the smallest unit for read and write operations.

The Oracle database architecture includes logical and physical structures that make up the database.

The physical structure includes the control files, online redo log files, and data files that make up the database.

The logical structure includes tablespaces, segments, extents, and data blocks.

The Oracle server enables fine-grained control of disk space use through tablespace and logical storage structures, including segments, extents, and data blocks.



Processing SQL Statements

- Connect to an instance using:
  - User process
  - Server process
- The Oracle server components that are used depend on the type of SQL statement:
  - Queries return rows
  - DML statements log changes
  - Commit ensures transaction recovery
- Some Oracle server components do not participate in SQL statement processing.

Processing SQL Statements

Processing a Query

Parse:

Search for identical statement

Check syntax, object names, and privileges

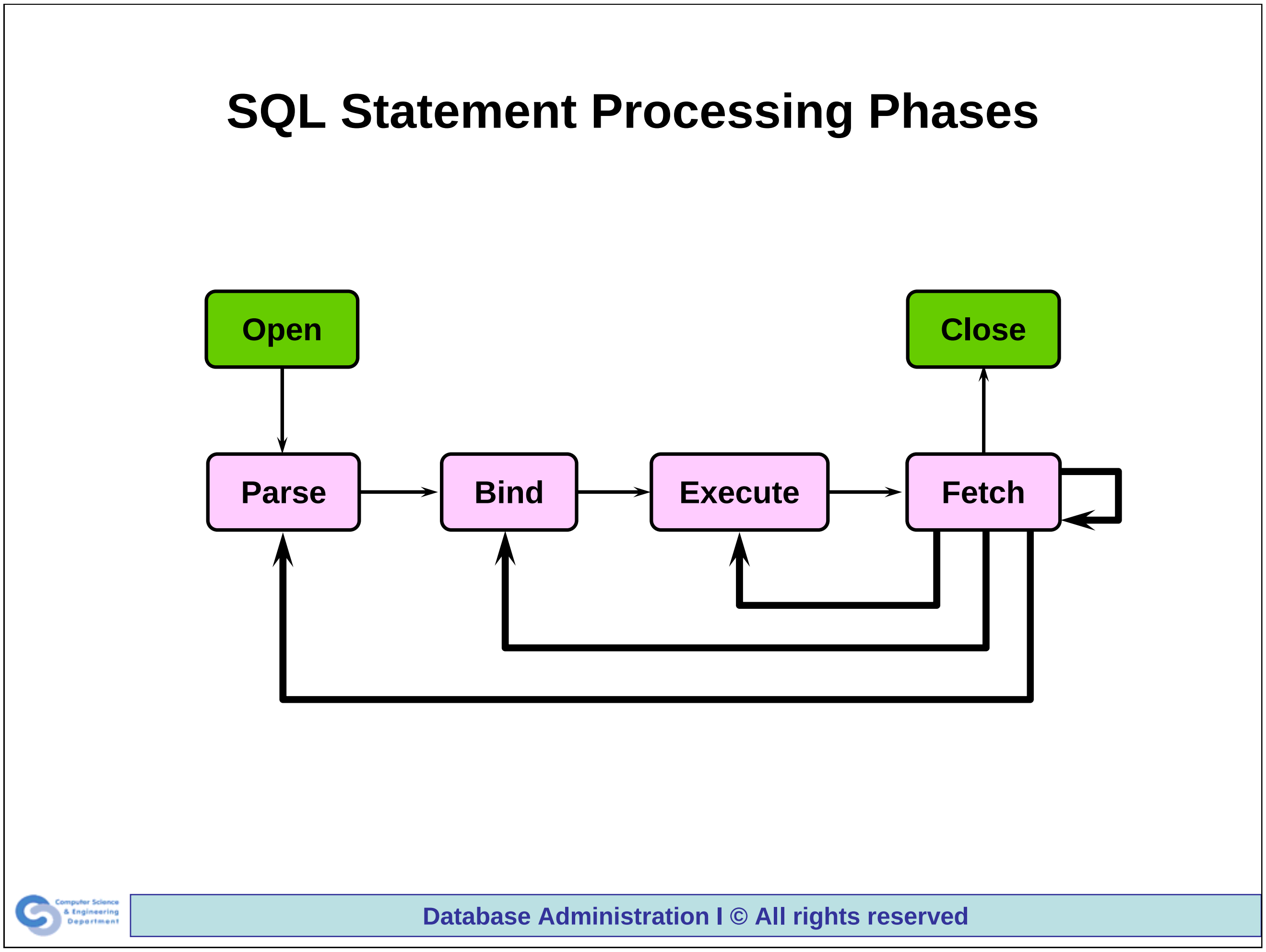
Lock objects used during parse

Create and store execution plan

Bind: Obtain values for variables

Execute: Process statement

Fetch: Return rows to user process



SQL Statement Processing Phases

Open

Close

Parse

Bind

Execute

Fetch

Processing Phases

The four most important phases in SQL statement processing are 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.