2. Database Architecture

Abstract: Every DBA needs to be intimately familiar with the components and facilities required to set up correctly the database server. These components include logical, physical, and memory structures. This lesson discusses aspects related to database architecture concepts. In order to provide a better understanding, the theoretical aspects are exemplified by analogy with the real-world concepts proposed by the Oracle database system.

Contents

Objective:

- Explain the database architecture concepts
- Describe the Oracle architecture and its main components

2.1. Logical and Physical Storage Structures

A database system consists of two major components — the ***database*** and the ***instance***. The database represents the physical files that store data. The instance comprises the memory structures and background processes used to access data (from the physical database files). Each database has at least one instance associated with it. Most database systems allow for multiple instances to access a single database (in Oracle this is known as the Real Application Cluster configuration).

The database, which is a collection of data, is used to store and retrieve information. Any database consists of logical structures and physical structures. ***Logical structures*** represent the components seen in the database (such as tables, indexes, and so on), and ***physical structures*** represent the method of storage used internally (the physical files).

EXAMPLE – Logical and Physical Storage Structures in Oracle

The Oracle server maintains the logical structure and physical structure separately, so that the logical structures can be defined identically across different hardware and operating system platforms.

In Oracle the database is logically divided into smaller units to manage, store, and retrieve data efficiently.

1. Tablespaces The database is logically divided into smaller units at the highest level called tablespaces. A ***tablespace*** commonly groups related logical structures together. For example, you might group data specific to an application or a function together in one or more tablespaces. This logical division helps to administer a portion of the database without affecting the rest of it. Each database should have one or more tablespaces. When you create a database, Oracle creates the *SYSTEM* tablespace as a minimum requirement.

2. Blocks A ***block*** is the smallest unit of storage in Oracle. A block is usually a multiple of the operating system block size. A data block corresponds to a specific number of bytes of storage space. The block size is based on the parameter *DB_BLOCK_SIZE* and is determined when the database is created.

3. Extents An ***extent*** is the next level of logical grouping. It is a grouping of contiguous blocks, allocated in one chunk.

4. Segments A ***segment*** is a set of extents allocated for logical structures such as tables, indexes, clusters, and so on. Whenever a logical structure is created, Oracle allocates a segment, which contains at least one extent, which in turn has at least one block. A segment can be associated to only one tablespace. Figure 1 shows the relationship between tablespaces, segments, extents, and blocks.

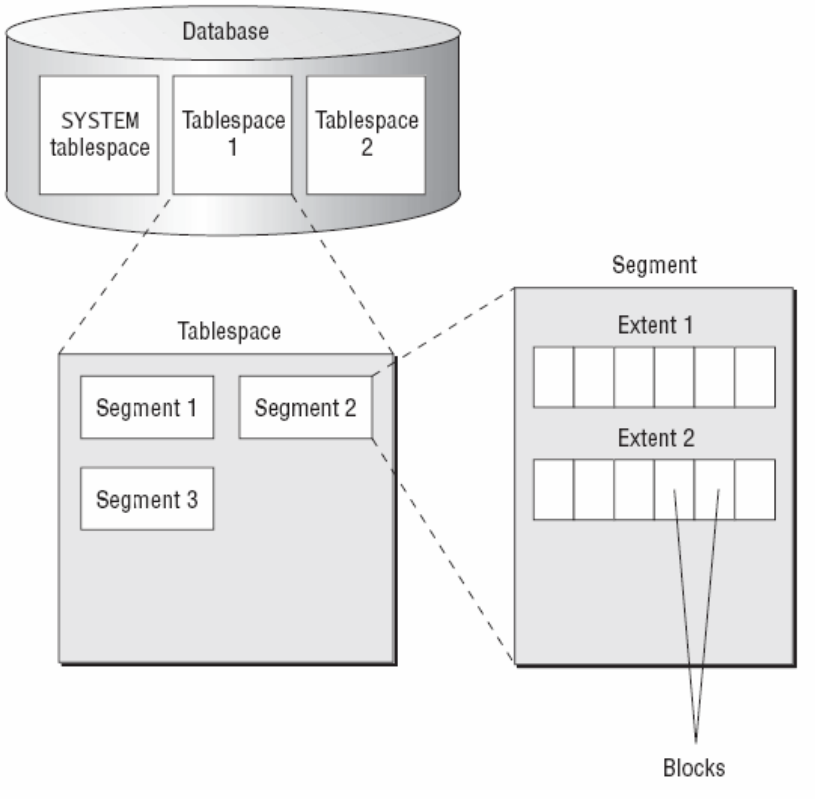

Figure 1. Logical Structures.

There are four types of segments:

1. Data segments Store the table (or cluster) data. A data segment is automatically allocated to a newly created table.

2. Index segments Store the index data. An index segment is automatically allocated to a newly create index.

3. Temporary segments Are created when Oracle needs a temporary work area, such as for sorting, during a query, and to complete execution of a SQL statement. These segments are freed when the execution completes.

4. Undo segments Store undo information. When you roll back the changes made to the database, undo records in the undo tablespace are used to undo the changes.

A schema is a logical structure that groups the database objects. A schema is not directly related to a tablespace or to any other logical storage structure. The objects that belong to a schema can reside in different tablespaces, and a tablespace can have objects that belong to multiple schemas. Schema objects include structures such as tables, indexes, synonyms, procedures, triggers, database links, and so on.

The physical database structure consists of three types of physical files:

- Data files
- Redo log files
- Control files

Figure 2 shows the physical structures and how the database is related to the memory structures and background processes. This figure also shows the relationship between tablespaces and data files.

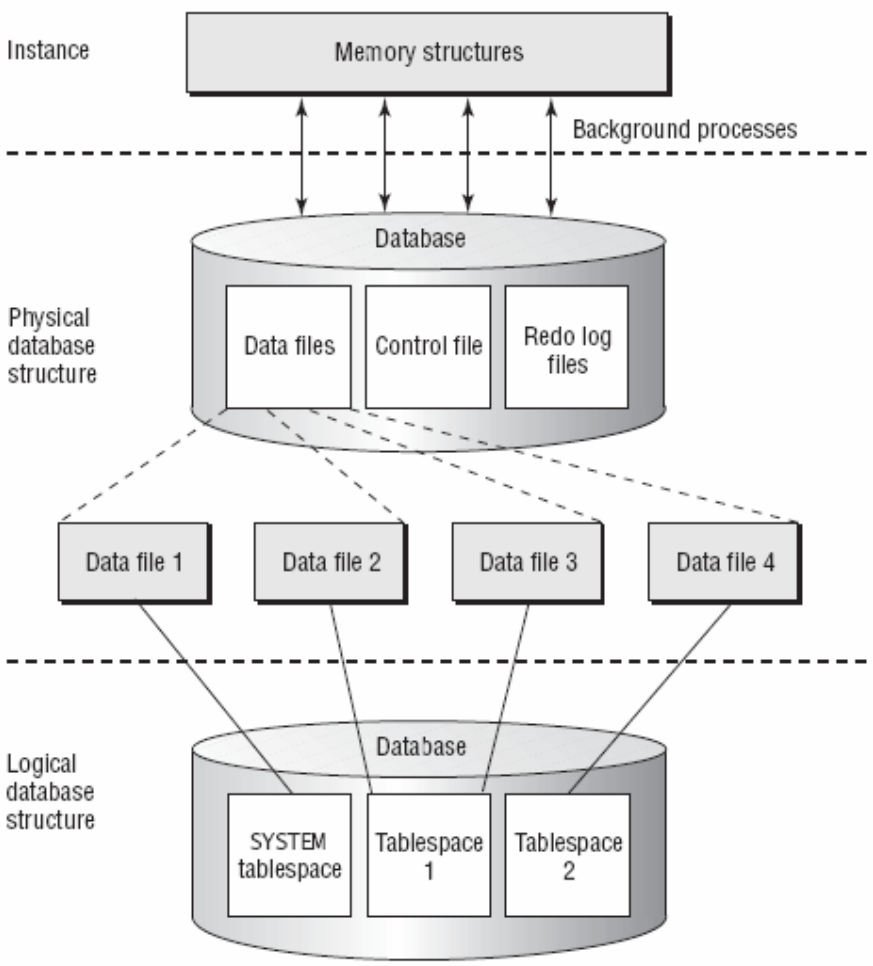

Figure 2. Oracle server physical structures.

Data files contain all the database data. Every Oracle database should have one or more data files. Each data file is associated with one and only one tablespace. A tablespace can consist of more than one data file.

Redo log files record all changes made to data. Every Oracle database should have two or more redo log files, because Oracle writes to the redo log files in a circular fashion. If a failure prevents a database change from being written to a data file, you can obtain the changes from the redo log files; therefore changes are never lost. Redo logs are critical for database operation and recovery from a failure. Oracle allows you to have multiple copies of the redo log files (preferably on different disks). This feature is known as *multiplexing* of redo logs, a process in which Oracle treats the redo log and its copies as a group identified with an integer, known as a redo log group.

Every Oracle database has at least one ***control file***. It maintains information about the physical structure of the database. The control file can be multiplexed, so that Oracle maintains multiple copies. It is critical to the database. The control file contains the database name and timestamp of database creation as well as the name and location of every data file and redo log file.

2.2. Application Architecture

The ***software architecture*** of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them. The software architecture discipline is centered on the idea of reducing complexity through abstraction and separation of concerns. To date there is still no agreement on the precise definition of the term "software architecture". Software architecture, also described as strategic design, is an activity concerned with global design constraints, such as programming paradigms, architectural styles, component-based software engineering standards, design principles, and law-governed regularities.

Client-server is a computing architecture which separates a client from a server, and is almost always implemented over a computer network. Each client or server connected to a network can also be referred to as a node. The most basic type of client-server architecture employs only two types of nodes: clients and servers. This type of architecture is sometimes referred to as **two-tier**. It allows devices to share files and resources.

Each instance of the client software can send data requests to one or more connected servers. In turn, the servers can accept these requests, process them, and return the requested information to the client. Although this concept can be applied for a variety of reasons to many different kinds of applications, the architecture remains fundamentally the same. These days, clients are most often web browsers, although that has not always been the case. Servers typically include web servers, database servers and mail servers.

Some designs are more sophisticated and consist of three different kinds of nodes: clients, application servers which process data for the clients, and database servers which store data for the application servers. This configuration is called a **three-tier architecture**, and is the

most commonly used type of client-server architecture. Designs that contain more than two tiers are referred to as **multi-tiered** or **n-tiered**.

The **multi-tier architecture** (or n-tier architecture) is a client-server architecture, originally designed by Jonathon Bolster of Hematites Corp, in which an application is executed by more than one distinct software agent. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of "multi-tier architecture" refers to three-tier architecture.

The advantages of n-tiered architectures are that they are far more scalable, since they balance and distribute the processing load among multiple, often redundant, specialized server nodes. This in turn improves overall system performance and reliability, since more of the processing load can be accommodated simultaneously.

Another type of network architecture is known as **peer-to-peer**, because each node or instance of the program can simultaneously act as both a client and a server, and because each has equivalent responsibilities and status. Peer-to-peer architectures are often abbreviated using the acronym P2P. Both client-server and P2P architectures are in wide usage today.

While classic Client-Server architecture requires one of communication endpoints to act as a server, which is much harder to implement, the **Client-Queue-Client architecture** allows all endpoints to be simple clients, while the server consists of some external software, which also acts as passive queue (one software instance passes its query to another instance to queue, e.g. database, and then this other instance pulls it from database, makes a response, passes it to database etc.). Such architecture allows to simplify software implementations many times. Peer-to-Peer architecture was originally based on Client-Queue-Client concept.

EXAMPLE – Application Architectures in Oracle

Client/Server Architecture

In the Oracle database system environment, the database application and the database are separated into two parts: a front-end or **client** portion, and a back-end or **server** portion—hence the term **client/server architecture**. The client runs the database application that accesses database information and interacts with a user through the keyboard, screen, and pointing device, such as a mouse. The server runs the Oracle software and handles the functions required for concurrent, shared data access to an Oracle database.

Although the client application and Oracle can be run on the same computer, greater efficiency can often be achieved when the client portions and server portion are run by different computers connected through a network.

Distributed processing is the use of more than one processor to perform the processing for an individual task. Examples of distributed processing in Oracle database systems appear in Figure 3.

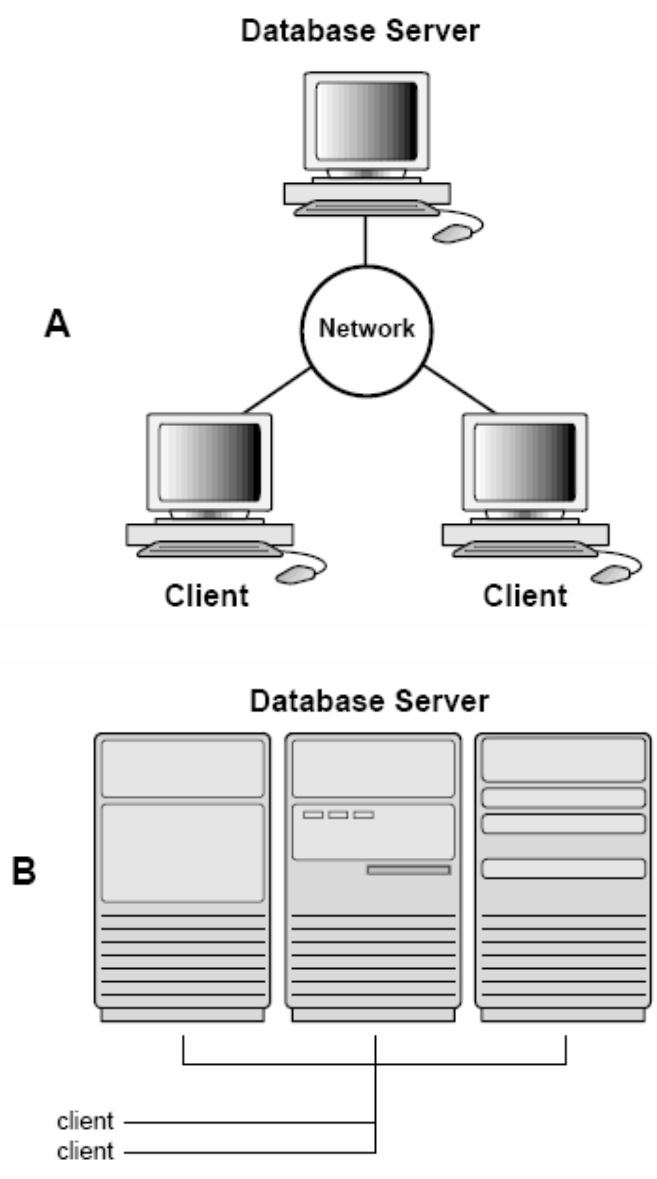

Figure 3. The Client/Server Architecture and Distributed Processing.

In Part A of the figure, the client and server are located on different computers, and these computers are connected through a network. The server and clients of an Oracle database system communicate through Oracle Net Services, Oracle's network interface.

In Part B of the figure, a single computer has more than one processor, and different processors separate the execution of the client application from Oracle.

Oracle client/server architecture in a distributed processing environment provides the following benefits:

- Client applications are not responsible for performing any data processing. Rather, they request input from users, request data from the server, and then analyze and present this data using the display capabilities of the client workstation or the terminal (for example, using graphics or spreadsheets).

- Client applications are not dependent on the physical location of the data. Even if the data is moved or distributed to other database servers, the application continues to function with little or no modification.

- Oracle exploits the multitasking and shared-memory facilities of its underlying operating system. As a result, it delivers the highest possible degree of concurrency, data integrity, and performance to its client applications.

- Client workstations or terminals can be optimized for the presentation of data (for example, by providing graphics and mouse support), and the server can be optimized for the processing and storage of data (for example, by having large amounts of memory and disk space).

- In networked environments, you can use inexpensive client workstations to access the remote data of the server effectively.

- If necessary, Oracle can be scaled as your system grows. You can add multiple servers to distribute the database processing load throughout the network (horizontally scaled), or you can move Oracle to a minicomputer or mainframe, to take advantage of a larger system's performance (vertically scaled). In either case, all data and applications are maintained with little or no modification, because Oracle is portable between systems.

- In networked environments, shared data is stored on the servers rather than on all computers in the system. This makes it easier and more efficient to manage concurrent access.

- In networked environments, client applications submit database requests to the server using SQL statements. After it is received, the SQL statement is processed by the server, and the results are returned to the client application. Network traffic is kept to a minimum, because only the requests and the results are shipped over the network.

Multitier Architecture

In a multitier architecture environment, an application server provides data for clients and serves as an interface between clients and database servers. This architecture is particularly important because of the prevalence of Internet use.

This architecture enables use of an application server to:

- Validate the credentials of a client, such as aWeb browser

- Connect to a database server
- Perform the requested operation

An example of a multitier architecture appears in Figure 4.

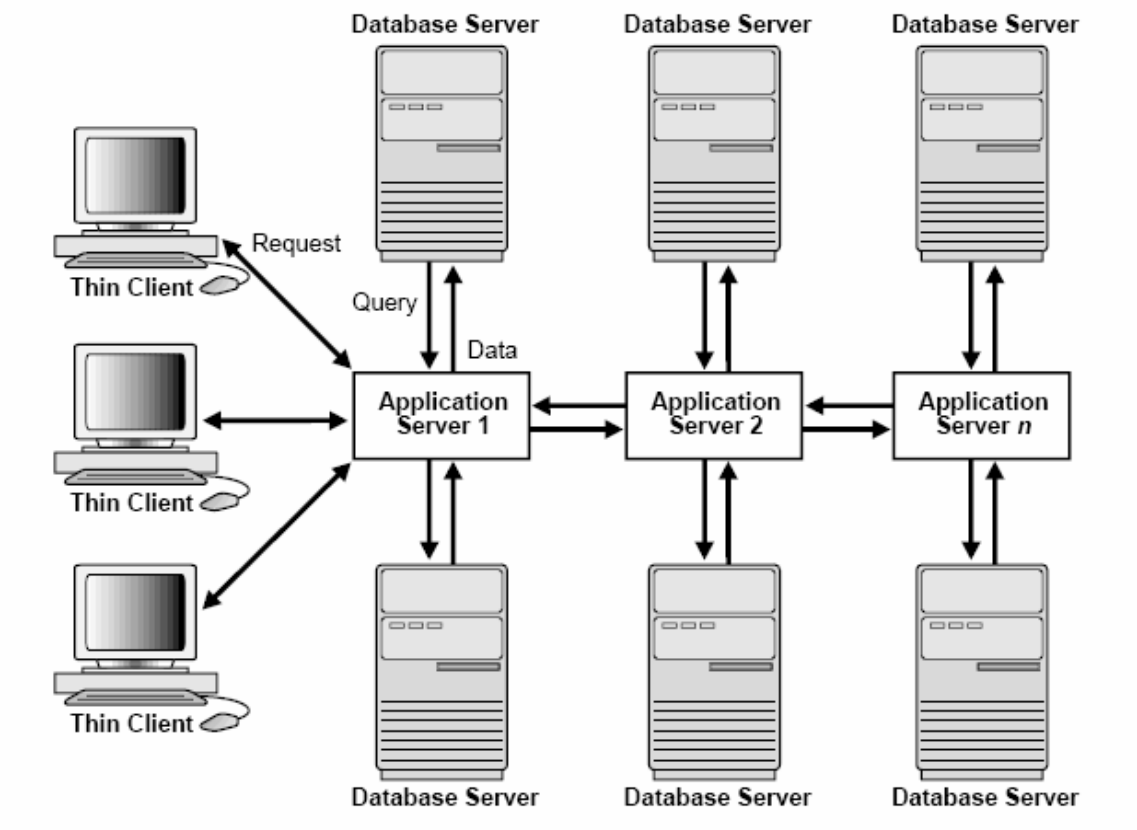

Figure 4. A Multitier Architecture Environment Example.

A **client** initiates a request for an operation to be performed on the database server. The client can be aWeb browser or other end-user process. In a multitier architecture, the client connects to the database server through one or more application servers.

An **application server** provides access to the data for the client. It serves as an interface between the client and one or more database servers, which provides an additional level of security. It can also perform some of the query processing for the client, thus removing some of the load from the database serve The application server assumes the identity of the client when it is performing operations on the database server for that client. The application server's privileges are restricted to prevent it from performing unneeded and unwanted operations during a client operation.

A **database server** provides the data requested by an application server on behalf of a client. The database server does all of the remaining query processing. The Oracle database server

can audit operations performed by the application server on behalf of individual clients as well as operations performed by the application server on its own behalf. For example, a client operation can be a request for information to be displayed on the client, whereas an application server operation can be a request for a connection to the database server.

Oracle Net Services

Oracle Net Services provides enterprise-wide connectivity solutions in distributed, heterogeneous computing environments. Oracle Net Services enables a network session from a client application to an Oracle database.

Oracle Net Services uses the communication protocols or application programmatic interfaces (APIs) supported by a wide range of networks to provide a distributed database and distributed processing for Oracle.

Oracle's support of industry network protocols provides an interface between Oracle processes running on the database server and the user processes of Oracle applications running on other computers of the network. The Oracle protocols take SQL statements from the interface of the Oracle applications and package them for transmission to Oracle through one of the supported industry-standard higher level protocols or programmatic interfaces. The protocols also take replies from Oracle and package them for transmission to the applications through the same higher level communications mechanism. This is all done independently of the network operating system.

Depending on the operation system that runs Oracle, the Oracle Net Services software of the database server could include the driver software and start an additional Oracle background process.

When an instance starts, a **listener process** establishes a communication pathway to Oracle. When a user process makes a connection request, the listener determines whether it should use a shared server dispatcher process or a dedicated server process and establishes an appropriate connection. The listener also establishes a communication pathway between databases. When multiple databases or instances run on one computer, as in Real Application Clusters, **service names** enable instances to register automatically with other listeners on the same machine. A service name can identify multiple instances, and an instance can belong to multiple services. Clients connecting to a service do not have to specify which instance they require.

Dynamic service registration reduces the administrative overhead for multiple databases or instances. Information about the services to which the listener forwards client requests is registered with the listener. Service information can by dynamically registered with the listener through a feature called **service registration** or statically configured in the *listener.ora* file.

Service registration relies on the **PMON process**—an instance background process—to register instance information with a listener, as well as the current state and load of the instance and shared server dispatchers. The registered information enables the listener to

forward client connection requests to the appropriate service handler. Service registration does not require configuration in the *listener.ora* file.

The initialization parameter SERVICE_NAMES identifies which database services an instance belongs to. On startup, each instance registers with the listeners of other instances belonging to the same services. During database operations, the instances of each service pass information about CPU use and current connection counts to all of the listeners in the same services. This enables dynamic load balancing and connection failover.

2.3. Memory Architecture

The **memory structures** are used to cache application data, data dictionary information, Structured Query Language (SQL) commands, PL/SQL and Java program units, transaction information, data required for execution of individual database requests, and other control information.

EXAMPLE – The memory architecture of an Oracle instance

Memory structures are allocated to the Oracle instance when the instance is started. The two major memory structures are known as the ***System Global Area*** (also called the **Shared Global Area**) and the ***Program Global Area*** (also called the **Private Global Area** or the **Process Global Area**). Figure 5 illustrates the various memory structures in Oracle.

The ***System Global Area*** (**SGA**) is a shared memory area. All users of the database share the information maintained in this area. The SGA and the background processes constitute an Oracle instance. Oracle allocates memory for the SGA when an Oracle instance is started and de-allocates it when the instance is shut down. The information stored in the SGA is divided into multiple memory structures that are allocated space when the instance is started. These memory structures are dynamic in Oracle9i, in which the total size cannot exceed the value specified in the initialization parameter *SGA_MAX_SIZE*.

Memory in the SGA is allocated in units of contiguous memory called *granules*. The size of a granule depends on the parameter *SGA_MAX_SIZE*; if the SGA size is less than 128MB, each granule is 4MB; otherwise, each granule is 16MB.

A minimum of three granules are allocated for the SGA: one for the fixed part of the SGA (redo buffers, locking information, database state information), one for the buffer cache, and one for the shared pool (library cache and data dictionary cache).

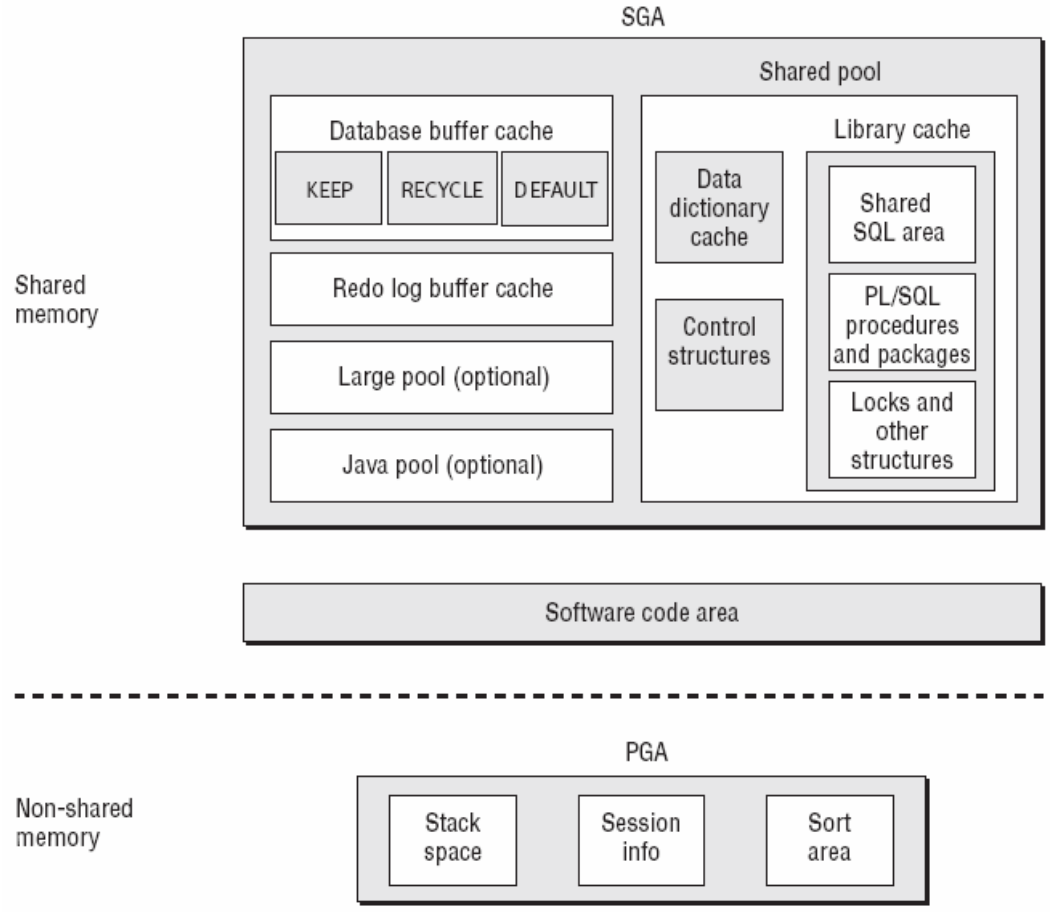

Figure 5. Oracle memory structures.

The components of the SGA are the database buffer cache, the redo log buffer, the shared pool, the large pool and the Java pool.

The ***database (DB) buffer cache*** is the area of memory that caches the database data, holding blocks from the data files that have been read recently. The DB buffer cache is shared among all the users connected to the database. There are three types of buffers:

- **Dirty buffers** Dirty buffers are the buffer blocks that need to be written to the data files. The data in these buffers has changed and has not yet been written to the disk.
- **Free buffers** Free buffers do not contain any data or are free to be overwritten. When Oracle reads data from disk, free buffers hold this data.
- **Pinned buffers** Pinned buffers are the buffers that are currently being accessed or explicitly retained for future use (for example, the *KEEP* buffer pool).

Oracle maintains two lists to manage the buffer cache. The *write list* (dirty buffer list) contains the buffers that are modified and need to be written to the disk (the dirty buffers).

The *least recently used* (LRU) list contains free buffers, pinned buffers, and the dirty buffers that have not yet been moved to the write list. Consider the LRU list as a queue of blocks, in which the most recently accessed blocks are always in the front (known as the most recently used, or MRU, end of the list; the other end, where the least recently accessed blocks are, is the LRU end). The least-used blocks are thrown out of the list when new blocks are accessed and added to the list.

When an Oracle process accesses a buffer, it moves the buffer to the MRU end of the list so that the most frequently accessed data is available in the buffers. When new data buffers are moved to the LRU list, they are copied to the MRU end of the list, pushing out the buffers from the LRU end. An exception to this procedure occurs when a full table is scanned; in this case, the blocks are written to the LRU end of the list. When an Oracle process requests data, it searches the data in the buffer cache, and if it finds data, the result is a cache hit. If it cannot find the data, the result is a cache miss, and data then needs to be copied from disk to the buffer.

Before reading a data block into the cache, the process must first find a free buffer. The server process on behalf of the user process searches either until it finds a free buffer or until it has searched the threshold limit of buffers. If the server process finds a dirty buffer as it searches the LRU list, it moves that buffer to the write list and continues to search. When the process finds a free buffer, it reads the data block from the disk into the buffer and moves the buffer to the MRU end of the LRU list. If an Oracle server process searches the threshold limit of buffers without finding a free buffer, the process stops searching and signals the DBWn background process to write some of the dirty buffers to disk. The DBWn process and other background processes are discussed in the next section.

NOTE: Oracle9i allows for three independent sub-caches within the database buffer cache: The *KEEP* buffer pool retains the data blocks in memory; they are not aged out. The *RECYCLE* buffer pool removes the buffers from memory as soon as they are not needed. The *DEFAULT* buffer pool contains the blocks that are not assigned to the other pools. The parameters *DB_CACHE_SIZE*, *DB_KEEP_CACHE_SIZE*, and *DB_RECYCLE_CACHE_SIZE* determine the size of the buffer cache. *DB_RECYCLE_CACHE_SIZE* determines the size to be allocated to the *RECYCLE* pool, and *DB_KEEP_CACHE_SIZE* determines the size to be allocated to the *KEEP* pool, in the buffer cache. The *DB_BLOCK_SIZE* parameter defines the size of an Oracle block in the buffer cache. When creating or altering tables and indexes, you can specify the *BUFFER_POOL* in the *STORAGE* clause.

To allow the DBA (database administrator) to size the components of the buffer cache efficiently, Oracle9i provides the “buffer cache advisory feature” to maintain the statistics associated with different cache sizes. This feature does incur a small performance hit on both memory and CPU, however.

NOTE: You use the parameter *DB_CACHE_ADVICE* to enable or disable statistics collection; its values can be *ON*, *OFF*, or *READY*. The view *V$DB_CACHE_ADVICE* is available for displaying the cache statistic

The ***redo log buffer*** is a circular buffer in the SGA that holds information about the changes made to the database data. The changes are known as redo entries or change vectors and are

used to redo the changes in case of a failure. Changes are made to the database through *INSERT*, *UPDATE*, *DELETE*, *CREATE*, *ALTER*, or *DROP* commands.

NOTE: The parameter *LOG_BUFFER* determines the size of the redo log buffer cache.

The ***shared pool*** portion of the SGA holds information such as SQL, PL/SQL procedures and packages, the data dictionary, locks, character set information, security attributes, and so on. The shared pool consists of the library cache and the data dictionary cache.

The ***library cache*** contains the shared SQL areas, private SQL areas, PL/SQL procedures and packages, and control structures such as locks and library cache handles.

The shared SQL area is used for maintaining recently executed SQL commands and their execution plans. Oracle divides each SQL statement that it executes into a shared SQL area and a private SQL area. When two users are executing the same SQL, the information in the shared SQL area is used for both. The shared SQL area contains the parse tree and execution plan, whereas the private SQL area contains values for the bind variables (persistent area) and runtime buffers (runtime area). Oracle creates the runtime area as the first step of an execute request. For *INSERT*, *UPDATE*, and *DELETE* statements, Oracle frees the runtime area after the statement has been executed. For queries, Oracle frees the runtime area only after all rows have been fetched or the query has been canceled.

Oracle processes PL/SQL program units the same way it processes SQL statements. When a PL/SQL program unit is executed, the code is moved to the shared PL/SQL area, and the individual SQL commands within the program unit are moved to the shared SQL area. Again, the shared program units are maintained in memory with an LRU algorithm. If another process requires the same program unit, Oracle can omit disk I/O and compilation, and the code that resides in memory will be executed.

The instance maintains the third area of the library cache for internal use. Various locks, latches, and other control structures reside here, and any server processes that require this information can freely access it.

The ***data dictionary*** is a collection of database tables and views containing metadata about the database, its structures, its privileges, and its users. Oracle accesses the data dictionary frequently during the parsing of SQL statements. The data dictionary cache holds the most recently used database dictionary information. The data dictionary cache is also known as the row cache because it holds data as rows instead of buffers (which hold entire blocks of data).

NOTE: The parameter *SHARED_POOL_SIZE* determines the size of the shared pool and can be dynamically altered.

The ***large pool*** is an optional area in the SGA that the DBA can configure to provide large memory allocations for specific database operations such as an Oracle backup or restore. The large pool allows Oracle to request large memory allocations from a separate pool to prevent contention from other applications for the same memory. The large pool does not have an LRU list.

NOTE: The parameter *LARGE_POOL_SIZE* specifies the size of the large pool and can be dynamically altered.

The ***Java pool*** is another optional area in the SGA that the DBA can be configured to provide memory for Java operations, just as the shared pool is provided for processing SQL and PL/SQL commands.

NOTE: The parameter *JAVA_POOL_SIZE* determines the size of the Java pool, and can be dynamically altered.

The ***Program Global Area*** (***PGA***) is the area in the memory that contains the data and process information for one process, and this area is non-shared memory. The contents of the PGA depend on the server configuration. For a dedicated server configuration (one dedicated server process for each connection to the database), the PGA holds stack space and session information. For shared server configurations, in which user connections are pooled through a dispatcher, the PGA contains the stack space information, and the session information is in the SGA. *Stack space* is the memory allocated to hold variables, arrays, and other information that belongs to the session. A PGA is allocated for each server process and de-allocated when the process is completed. Unlike the SGA that is shared by several processes, the PGA provides sort space, session information, stack space, and cursor information for a single server process.

The memory area that Oracle uses to sort data is known as the ***sort area***, and it uses memory from the PGA for a dedicated server connection. For shared server configurations, the sort area is allocated from the SGA. Shared and dedicated server configurations are discussed later in this chapter. Sort area size can grow depending on the need; you use the *SORT_AREA_SIZE* parameter to set the maximum size. The parameter *SORT_AREA_RETAINED_SIZE* determines the size to which the sort area is reduced after the sort operation. The memory released from the sort area is kept with the server process; it is not released to the operating system.

If the data to be sorted does not fit into the memory area defined by *SORT_AREA_SIZE*, Oracle divides the data into smaller pieces that do fit and sorts these individually. These individual sorts are called *runs*, and the sorted data is held in the user's temporary tablespace using temporary segments. When all the individual sorts are complete, Oracle merges these runs to produce the final result. Oracle sorts the result set if the query contains a *DISTINCT*, *ORDER BY*, or *GROUP BY* operator or any set operators (*UNION*, *INTERSECT*, *MINUS*).

Managing *SORT_AREA_SIZE* in a large enterprise environment may be challenging. The trick is trying to maximize performance without using up too many system resources. Oracle9i provides an automatic method for managing PGA memory. The two key initialization parameters used to automate PGA memory management are *PGA_AGGREGATE_TARGET* and *WORKAREA_SIZE_POLICY*. The value for *PGA_AGGREGATE_TARGET* specifies the total amount of memory that can be used by all server processes, and the value for *WORKAREA_SIZE_POLICY* is either *MANUAL* or *AUTO*.

Software code areas are the portions of memory that store the code that is being executed. Software code areas are mostly static in size and depend on the operating system. These areas

are read-only and can be shared (if the operating system allows), so multiple copies of the same code are not kept in memory. Some Oracle tools and utilities (such as SQL*Forms and SQL*Plus) can be installed as shared, but some cannot. Multiple instances of Oracle can use the same Oracle code area with different databases if they are running on the same computer.

2.4. Process Architecture

A ***process*** is a mechanism used in the operating system to execute a series of tasks. The term 'process' was introduced historically as an abstraction to capture the notion of dynamic execution. In modern systems, particularly since the advent of window-based interfaces there is often a need for a single program (or application) to allow several activities to be in progress at the same time. For example, you may wish to work on the draft of an email message but pause to incorporate and read new email coming in, then reply to an urgent message before resuming your original draft.

A process is a "thread of control" or a mechanism in an operating system that can run a series of steps. (Some operating systems use the terms job or task.) A process normally has its own private memory area in which it runs.

EXAMPLE – Processes in Oracle

All connected Oracle users must run two modules of code to access an Oracle database instance.

- Application or Oracle tool: A database user runs a database application (such as a precompiler program) or an Oracle tool (such as SQL*Plus), which issues SQL statements to an Oracle database.
- Oracle server code: Each user has some Oracle server code executing on his or her behalf, which interprets and processes the application's SQL statements.

These code modules are run by processes. The processes in an Oracle system can be categorized into two major groups:

- User processes run the application or Oracle tool code.
- Oracle processes run the Oracle server code. They include server processes and background processes.

The process structure varies for different Oracle configurations, depending on the operating system and the choice of Oracle options. The code for connected users can be configured as a dedicated server or a shared server. With dedicated server, for each user, the database

application is run by a different process (a user process) than the one that runs the Oracle server code (a dedicated server process). With shared server, the database application is run by a different process (a user process) than the one that runs the Oracle server code. Each server process that runs Oracle server code (a shared server process) can serve multiple user processes.

Figure 6 illustrates a dedicated server configuration. Each connected user has a separate user process, and several background processes run Oracle.

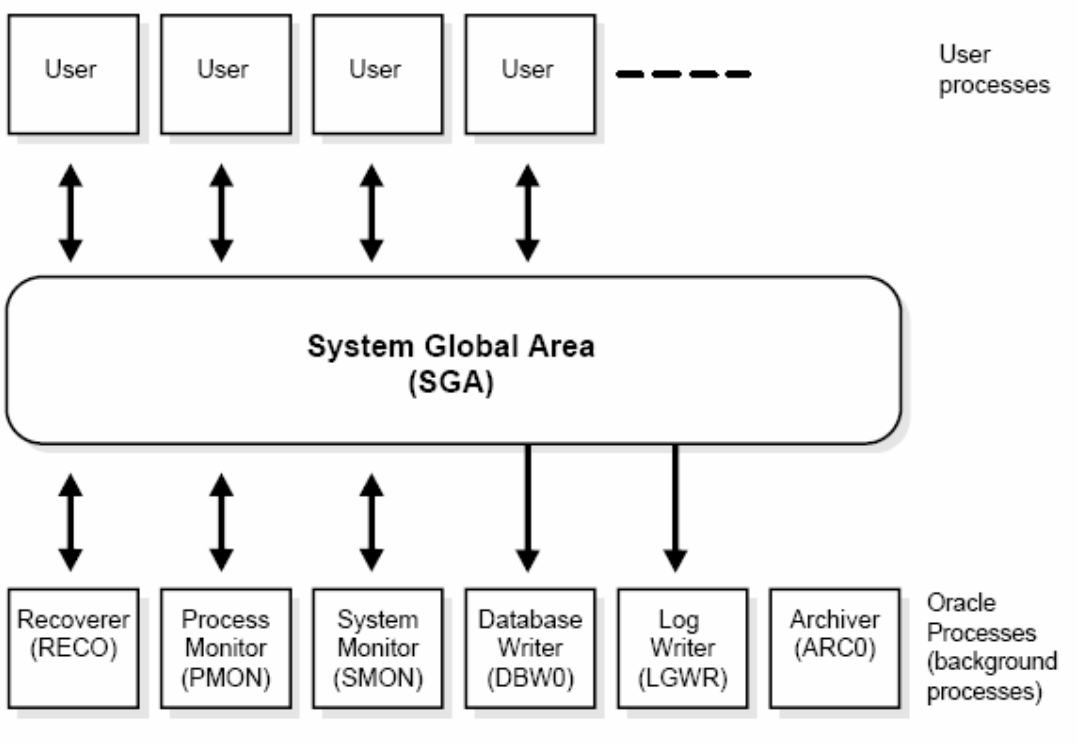

Figure 6. An Oracle Instance.

When a user runs an application program or an Oracle tool (such as Enterprise Manager or SQL*Plus), Oracle creates a user process to run the user's application.

Connection and **session** are closely related to **user process** but are very different in meaning.

A **connection** is a communication pathway between a user process and an Oracle instance. A communication pathway is established using available interprocess communication mechanisms (on a computer that runs both the user process and Oracle) or network software (when different computers run the database application and Oracle, and communicate through a network).

A **session** is a specific connection of a user to an Oracle instance through a user process. For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time the user connects

until the time the user disconnects or exits the database application.

Multiple sessions can be created and exist concurrently for a single Oracle user using the same username. For example, a user with the username/password of SCOTT/TIGER can connect to the same Oracle instance several times.

In configurations without the shared server, Oracle creates a server process on behalf of each user session. However, with the shared server, many user sessions can share a single server process.

There are two types of processes that run the Oracle server code (**server processes** and **background processes**).

Oracle creates **server processes** to handle the requests of user processes connected to the instance. In some situations when the application and Oracle operate on the same machine, it is possible to combine the user process and corresponding server process into a single process to reduce system overhead. However, when the application and Oracle operate on different machines, a user process always communicates with Oracle through a separate server process.

Server processes (or the server portion of combined user/server processes) created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application
- Read necessary data blocks from datafiles on disk into the shared database buffers of the SGA, if the blocks are not already present in the SGA
- Return results in such a way that the application can process the information

To maximize performance and accommodate many users, a multiprocess Oracle system uses some additional Oracle processes called **background processes**. An Oracle instance can have many background processes; not all are always present. The background processes in an Oracle instance are presented in Figure 7.

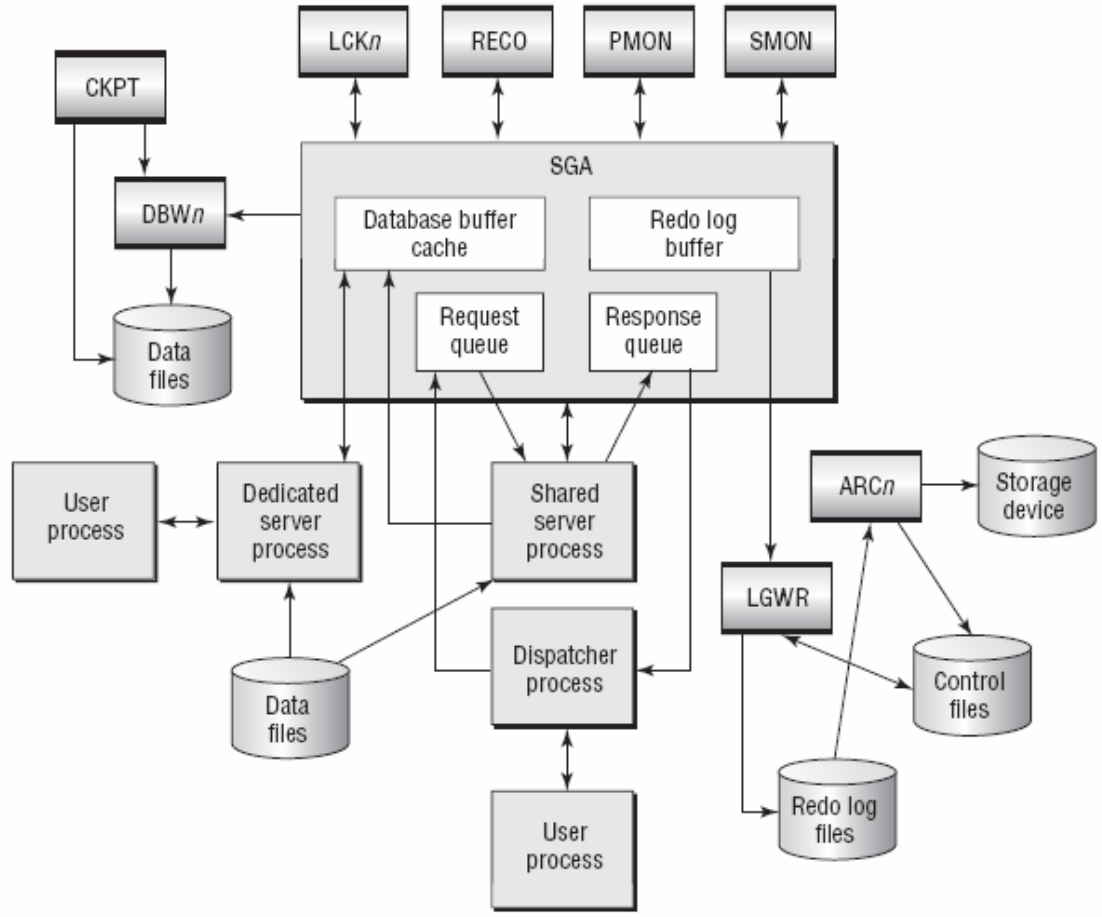

Figure 7. The Oracle background processes.

The purpose of the **database writer process** (**DBWn**) is to write the contents of the dirty buffers to the data file. By default, Oracle starts one database writer process (*DBW0*) when the instance starts; for multi-user and busy systems, you can have nine more database writer processes (*DBW1* through *DBW9*) to improve performance. The parameter *DB_WRITER_PROCESSES* determines the additional number of database writer processes to be started.

The *DBWn* process writes the modified buffer blocks to disk, so more free buffers are available in the buffer cache. Writes are always performed in bulk to reduce disk contention; the number of blocks written in each I/O is operating system dependent. The *DBWn* process initiates writing to data files under these circumstances:

- When the server process cannot find a clean buffer after searching the set threshold of buffers, it initiates the *DBWn* process to write dirty buffers to the disk, so that some buffers are freed.

- When a checkpoint occurs, *DBWn* periodically writes buffers to disk.
- When a timeout occurs.
- When you change a tablespace to read-only.
- When you place a tablespace offline.
- When you drop or truncate a table.
- When you place a tablespace in BACKUP mode.

NOTE: Writes to the data file(s) are independent of the corresponding *COMMIT* performed in the SQL code.

The **log writer process** (**LGWR**) writes the blocks in the redo log buffer in the SGA to the online redo log files. The redo log buffer is circular. When the LGWR writes log buffers to the disk, Oracle server processes can write new entries in the redo log buffer. LGWR writes the entries to the disk fast enough to ensure that room is available for the server process to write log information. The log writer process writes the buffers to the disk under the following circumstances:

- When a user transaction issues a *COMMIT*
- When the redo log buffer is one-third full
- When the DBWn process writes dirty buffers to disk
- Every three seconds
- When there is one megabyte of redo records

LGWR writes simultaneously to the multiplexed online redo log files. Even if one of the log files in the group is damaged, LGWR continues writing to the available file. LGWR writes to the redo logs sequentially so that transactions can be applied in order in the event of a failure.

NOTE: By writing the committed transaction to the redo log files, the change to the database is never lost (that is, it can be recovered if a failure occurs).

Checkpoints help to reduce the time required for instance recovery. A **checkpoint** is an event that flushes the modified data from the buffer cache to the disk and updates the control file and data files. The **checkpoint process** (**CKPT**) updates the headers of data files and control files; the DBWn process writes the actual blocks to the file.

If checkpoints occur too frequently, disk contention becomes a problem with the data file updates. If checkpoints occur too infrequently, the time required to recover a failed database can be significantly longer. Checkpoints occur automatically when an online redo log file fills

(log switch). A log switch occurs when Oracle finishes writing one file and starts the next file.

The **system monitor process** (**SMON**) performs instance or crash recovery at database start-up by using the online redo log files. SMON is also responsible for cleaning up temporary segments in the tablespaces that are no longer used and for coalescing the contiguous free space in the tablespaces. If any dead transactions were skipped during crash and instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online. SMON wakes up regularly to check whether it is needed. Other processes can call SMON if they detect a need for SMON to wake up.

NOTE: SMON coalesces the contiguous free space in a tablespace only if its default *PCTINCREASE* value is set to a nonzero value.

The **process monitor process** (**PMON**) cleans up failed user processes and frees up all the resources used by the failed process. It resets the status of the active transaction table and removes the process ID from the list of active processes. It reclaims all resources held by the user and releases all locks on tables and rows held by the user. PMON wakes up periodically to check whether it is needed.

NOTE: DBWn, LGWR, CKPT, SMON, and PMON processes are the default processes associated with all instances.

When the Oracle database is running in *ARCHIVELOG* mode, the online redo log files are copied to another location before they are overwritten. You can use these archived log files to recover the database. When the database is in *ARCHIVELOG* mode, you can recover the database up to the point of failure. The **archiver process** (**ARCn**) performs the archiving function. Oracle9i can have as many as 10 *ARCn* processes (*ARC0* through *ARC9*). The *LGWR* process starts new *ARCn* processes whenever the current number of *ARCn* processes is insufficient to handle the workload. The *ARCn* process is enabled only if the database is in *ARCHIVELOG* mode and automatic archiving is enabled (parameter *LOG_ARCHIVE_START = TRUE*).

The **recoverer process** (**RECO**) is used with distributed transactions to resolve failures. The RECO process is present only if the instance permits distributed transactions and if the *DISTRIBUTED_TRANSACTIONS* parameter is set to a nonzero value. If this initialization parameter is zero, RECO is not created during instance start-up. This process attempts to access databases involved in in-doubt transactions and resolves the transactions. A transaction is in doubt when you change data in multiple databases and a failure occurs before you save the changes. The failure can be the result of a server crash or a network problem.

LCKn processes (LCK0 through LCK9) are used in the Real Application Cluster environment, for inter-instance locking. The Real Application Cluster option lets you mount the same database for multiple instances.

The **queue monitor process** is used for Oracle Advanced Queuing, which monitors the message queues. You can configure as many as 10 queue monitor processes (QMN0 through QMN9). Oracle Advanced Queuing provides an infrastructure for distributed applications to communicate asynchronously using messages. Oracle Advanced Queuing stores messages in

queues for deferred retrieval and processing by the Oracle Server. The parameter *AQ_TM_PROCESSES* specifies the number of queue monitor processes.

NOTE: Failure of an SNP process or a QMN process does not cause the instance to crash; Oracle restarts the failed process. If any other background process fails, the Oracle instance fails.

Dispatcher processes are part of the shared server architecture. They minimize the resource needs by handling multiple connections to the database using a limited number of server processes. You can create multiple dispatcher processes for a single database instance; you must create at least one dispatcher for each network protocol used with Oracle.

Shared server processes provide the same functionality as the dedicated server processes, except that shared server processes are not associated with a specific user process. You create shared server processes to manage connections to the database in a shared server configuration. The number of shared server processes that you can create ranges between the values of the parameters *SHARED_SERVERS* and *MAX_SHARED_SERVERS*.

Each server and background process can write to an associated **trace file**. When a process detects an internal error, it dumps information about the error to its trace file. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle support.

All filenames of trace files associated with a background process contain the name of the process that generated the trace file. The one exception to this is trace files generated by job queue processes (*Jnnn*).

Additional information in trace files can provide guidance for tuning applications or an instance. Background processes always write this information to a trace file when appropriate.

Oracle uses the alert file to keep a record of these events as an alternative to displaying the information on an operator's console. (Many systems also display this information on the console.) If an administrative operation is successful, a message is written in the alert file as "completed" along with a time stamp.

EXAMPLE – Processing SQL Statements

Structured Query Language (SQL) is used to manipulate and retrieve data in an Oracle database. Data Manipulation Language (DML) statements query or manipulate data in existing database objects. *SELECT*, *INSERT*, *UPDATE*, *DELETE*, *EXPLAIN PLAN*, and *LOCK TABLE* are *DML* statements.

These are the most commonly used statements in the database. SQL statements are processed in either two or three steps, or stages. Each SQL statement passed to the server process from the user process goes through parse and execute phases. In the case of queries (*SELECT* statements), an additional phase of fetch is done to retrieve the rows.

Parse Parsing is one of the first stages in processing any SQL statement. When an application or tool issues a SQL statement, it makes a parse call to Oracle, which does the following:

- Checks the statement for syntax correctness and validates the table names and column names against the dictionary
- Determines whether the user has privileges to execute the statement
- Determines the optimal execution plan for the statement
- Finds a shared SQL area for the statement

If there is an existing SQL area with the parsed representation of the statement in the library cache, Oracle uses this parsed representation and executes the statement immediately. If not, Oracle generates the parsed representation of the statement, allocates a shared SQL area for the statement in the library cache, and stores its parsed representation there. Oracle's parse operation allocates a shared SQL area for the statement, which allows the statement to be executed any number of times without parsing repeatedly.

Execute Oracle executes the parsed statement in the execute stage. For *UPDATE* and *DELETE* statements, Oracle locks the rows that are affected, so that no other process is making changes to the rows until the transaction is completed. Oracle also looks for data blocks in the data buffer cache. If it finds them, the execution will be faster; if not, Oracle has to read the data blocks from the physical data file to the buffer cache. If the statement is a *SELECT* or an *INSERT*, no rows need to be locked because no data is being changed.

Fetch The fetch operation follows the execution of a SQL *SELECT* command. After the execution completes, the rows identified during the execution stage are returned to the user process. The rows are ordered (sorted) if requested by the query. The results are always in a tabular format; rows may be fetched (retrieved) one row at a time or in groups (array processing).

Figure 8 shows the steps required in processing a *SELECT* query.

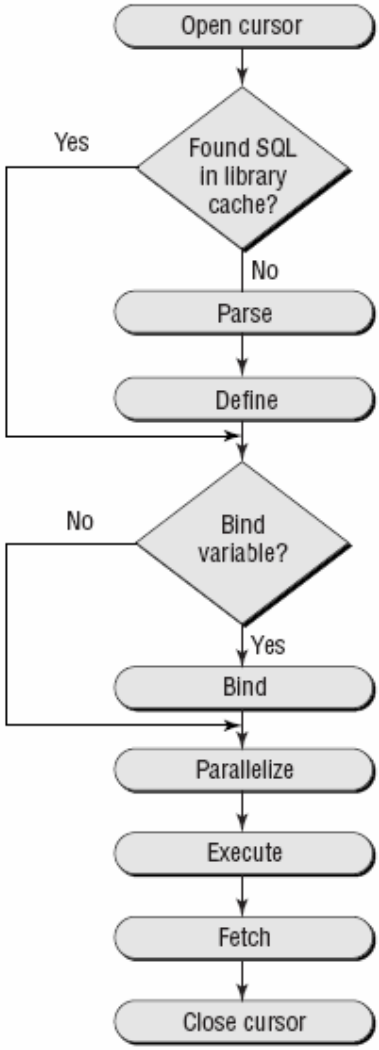

Figure 8. Query processing stages.

1. Create a cursor. The cursor may be explicit or implicit.

2. Parse the statement.

3. Define output: specify location, type, and data type of the result set, and convert the data type if necessary. This step is required only when the results are fetched to variables.

4. Bind variables; if the query is using any variables, Oracle should know the value for the variables.

5. See whether the query can be run in parallel, that is, whether multiple server processes can be working to complete the query.

6. Execute the query.

7. Fetch the rows.

8. Close the cursor.

Figure 9 shows the steps required to process one of the DML statements *INSERT*, *UPDATE*, or *DELETE*.

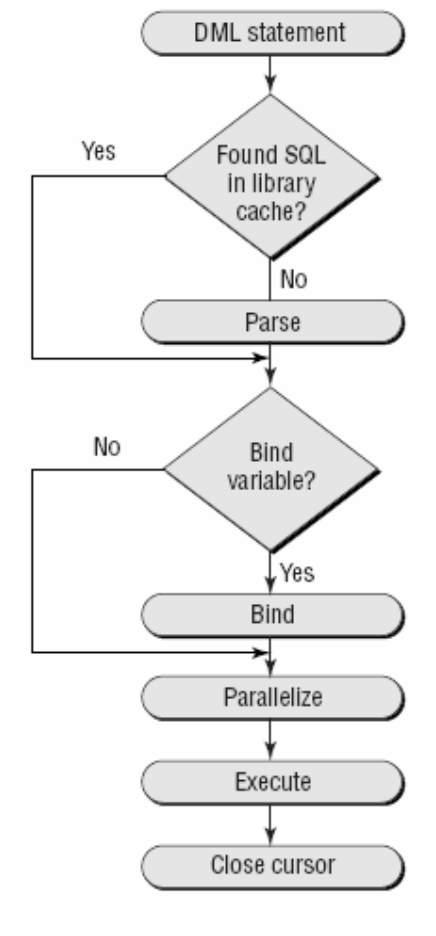

Figure 9. DML processing stages.

1. Create a cursor; Oracle creates an implicit cursor.

2. Parse the statement.

3. Bind variables; if the statement is using any variables, Oracle should know the value for the variables.

4. See whether the statement can be run in parallel (multiple server processes working

to complete the work).

5. Execute the statement.

6. Inform the user that the statement execution is complete.

7. Close the cursor.

After seeing how the server process processes a query and other DML statements we now look at the steps in processing a *COMMIT* or a *ROLLBACK*. Let's look at an important mechanism that Oracle uses for recovery—the *system change number* (SCN).

When a transaction commits, Oracle assigns a unique number that defines the database state at a precise moment in time, acting as an internal timestamp.

The SCN is a serial number, unique and always increasing. SCNs provide a read-consistent view of the database. The database is always recovered based on the SCN. The SCN is also used to provide a read-consistent view of the data. When a query reaches the execution stage, the current SCN is determined; only the blocks with an SCN less than or equal to this SCN are read—for changed blocks (with a higher SCN), data is read from the rollback segments.

The SCN is recorded in the control file, data file headers, block headers, and redo log files. The redo log file has a low SCN (the lowest change number stored in the log file) and high SCN (the highest change number in the log file—assigned when the file is closed, before opening the next redo log file). The SCN value is stored in every data file header, which is updated whenever a checkpoint is done. The control file records the SCN number for each data file that is taken offline.

Oracle commits a transaction when you do the following:

- Issue a *COMMIT* command
- Execute a DDL statement
- Disconnect from Oracle

The following are the steps for processing a *COMMIT*, that is, making the changes to the database permanent:

1. The server process generates an SCN number and is assigned to the rollback segment; then it marks in the rollback segment that the transaction is committed.

2. The LGWR process writes the redo log buffers to the online redo log files along with the SCN number.

3. The server process releases locks held on rows and tables.

4. The user is notified that the *COMMIT* is complete.

5. The server process marks the transaction as complete.

NOTE: Oracle defers writes to the data file to reduce disk I/O. The DBWn process writes the changed blocks to the data file independent of any *COMMIT*. By writing the change vectors and the SCN number to the redo log files, Oracle ensures that the committed changes are never lost. This process is known as the fast commit—writes to redo log files are faster than writing the blocks to data files.

If the transaction has not been committed, it can be rolled back; that is, the state of the database tables for the session are restored to their original values.

Oracle rolls back a transaction when:

- You issue a ROLLBACK command.
- The server process terminates abnormally.
- The DBA kills the session.

The following steps are used in processing a ROLLBACK, that is, in undoing the changes to the database:

1. The server process undoes all changes made in the transaction by using the undo segment entries.

2. The server process releases all locks held on tables and rows.

3. The server process marks the transaction as complete.

References

[1] Bob Bryla, "*Oracle 9i DBA Jump Start*", Sybex, 2003.

[2] Michael J. Hernandez, "Proiectarea Bazelor de Date", Teora, 2003.

[3] Oracle9i Database Concepts, Oracle, 2002.

Exam Essentials

Identify the three types of database files that constitute the database. Briefly describe the purpose and key differences between control files, data files, and redo log files. Describe other essential files that are needed to start up the database but are not considered a part of the database.

Explain and categorize the SGA memory structures. Identify the SGA areas along with the sub-components contained within each of these areas. Be able to place a database-related object (for example, a SQL statement or a data file block) into its appropriate SGA area.

Understand the steps involved in processing a SQL statement. Understand which server components do and do not participate in processing SQL, and understand the steps required when a DML statement is executed.

Enumerate and explain the primary (required) background processes. Identify each process with its primary purpose, its interaction with other background processes, and when the background process is active.

Understand the purpose of the PGA. List the components of the PGA, as well as understand the conditions under which PGA components are stored in the SGA.

Identify the initialization parameters related to the SGA and buffer pool sizing. Understand which parameters can be dynamically altered and which parameters are optional.

Review Questions

1. Which component is not part of the Oracle instance?

A. System Global Area

B. Process monitor

C. Control file

D. Shared pool

Correct answer: C. The Oracle instance consists of memory structures and background processes. The Oracle database consists of the physical components such as data files, redo log files, and the control file. The System Global Area and shared pool are memory structures. The process monitor is a background process.

2. Which background process and associated database component guarantee that committed data is saved even when the changes have not been recorded in the data files?

A. DBWn and database buffer cache

B. LGWR and online redo log file

C. CKPT and control file

D. DBWn and archived redo log file

Correct answer: B. The LGWR process writes the redo log buffer entries when a COMMIT occurs. The redo log buffer holds information on the changes made to the database. The DBWn process writes dirty buffers to the data file, but it is independent of the COMMIT. The dirty buffers can be written to the disk before or after a COMMIT. Writing the committed changes to the online redo log file ensures that the changes are never lost in case of a failure.

3. What is the maximum number of database writer processes allowed in an Oracle instance?

A. 1

B. 10

C. 256

D. Limit specified by an operating system parameter

Correct answer: B. By default, every Oracle instance has one database writer process—DBW0. Additional processes can be started by setting the initialization parameter DB_WRITER_PROCESSES(DBW1 through DBW9).

4. Which background process is not started by default when you start up the Oracle instance?

A. DBWn

B. LGWR

C. CKPT

D. ARCn

Correct answer: D. ARCn is the archiver process, which is started only when the LOG_ARCHIVE_START initialization parameter is set to TRUE. DBWn, LGWR, CKPT, SMON, and PMON are the default processes associated with all instances.

5. Choose the correct hierarchy, from largest to smallest, from this list of logical database structures.

A. Database, tablespace, extent, segment, block

B. Database, tablespace, segment, extent, block

C. Database, segment, tablespace, extent, block

D. Database, extent, tablespace, segment, block

Correct answer: B. The first level of logical database structure is the tablespace. A tablespace may have segments, segments have one or more extents, and extents have one or more contiguous blocks.

6. Which component of the SGA contains the parsed SQL code?

A. Buffer cache

B. Dictionary cache

C. Library cache

D. Parse cache

Correct answer: C. The library cache contains the parsed SQL code. If a query is executed again before it is aged out of the library cache, Oracle will use the parsed code and execution plan from the library cache. The buffer cache has data blocks that are cached. The dictionary cache caches data dictionary information. There is no SGA component named parse cache.

7. Julie, one of the database analysts, is complaining that her queries are taking longer and longer to complete, although they seem to produce the correct results. The DBA suspects that the buffer cache is not sized correctly and is causing delays due to data blocks not being available in memory. Which initialization parameter should the DBA use to monitor the usage of the buffer cache?

A. BUFFER_POOL_ADVICE

B. DB_CACHE_ADVICE

C. DB_CACHE_SIZE

D. SHARED_POOL_SIZE

Correct answer: B. The parameter DB_CACHE_ADVICE can be set to YES to enable cache usage monitoring. DB_CACHE_SIZE and SHARED_POOL_SIZE are sizing parameters for SGA structures; the parameter BUFFER_POOL_ADVICE does not exist.

8. Which background process is responsible for writing the dirty buffers to the database files?

A. DBWn

B. SMON

C. LGWR

D. CKPT

E. PMON

Correct answer: A. The DBWn process writes the dirty buffers to the data files under two circumstances—when a checkpoint occurs or when the server process searches the buffer cache for a set threshold.

9. Which component in the SGA has the dictionary cache?

A. Buffer cache

B. Library cache

C. Shared pool

D. Program Global Area

E. Large pool

Correct answer: C. The shared pool has three components: the library cache, the dictionary cache, and the control structures.

10. When a server process is terminated abnormally, which background process is responsible for releasing the locks held by the user?

A. DBWn

B. LGWR

C. SMON

D. PMON

Correct answer: D. PMON, or the process monitor, is responsible for cleaning up failed user processes. It reclaims all resources held by the user and releases all locks on tables and rows held by the user.

11. What is a dirty buffer?

A. Data buffer that is being accessed

B. Data buffer that is changed but is not written to the disk

C. Data buffer that is free

D. Data buffer that is changed and written to the disk

Correct answer: B. Dirty buffers are the buffer blocks that need to be written to the data files. The data in these buffers has changed and is not yet written to the disk. A block waiting to be written to disk is on the dirty list and cannot be overwritten.

12. If you are updating one row in a table using the ROWID in the WHERE clause (assume that the row is not already in the buffer cache), what will be the minimum amount of information read to the database buffer cache?

A. The entire table is copied to the database buffer cache.

B. The extent is copied to the database buffer cache.

C. The block is copied to the database buffer cache.

D. The row is copied to the database buffer cache.

Correct answer: C. The block is the smallest unit that can be copied to the buffer cache.

13. What happens next when a server process is not able to find enough free buffers to copy the blocks from disk?

A. Signals the CKPT process to clean up the dirty buffers

B. Signals the SMON process to clean up the dirty buffers

C. Signals the CKPT process to initiate a checkpoint

D. Signals the DBWn process to write the dirty buffers to disk

Correct answer: D. To reduce disk I/O contention, the DBWn process does not write the changed buffers immediately to the disk. They are written only when the dirty buffers reach a threshold, when there are not enough free buffers available, or when the checkpoint occurs.

14. Which memory structures are shared? Choose two.

A. Sort area

B. Program Global Area

C. Library cache

D. Large pool

Correct answer: C and D. The sort area is allocated to the server process as part of the PGA. The PGA is allocated when the server process starts and is deallocated when the server process completes. The library cache and the large pool are part of the SGA and are shared. The SGA is created when the instance starts.

15. Which of the following initialization parameters does NOT determine the size of the buffer cache?

A. DB_KEEP_CACHE_SIZE

B. DB_CACHE_SIZE

C. DB_BLOCK_SIZE

D. DB_RECYCLE_CACHE_SIZE

Correct answer: C. The parameter DB_BLOCK_SIZE does not change the size of the buffer cache. It changes only the size of each Oracle block written to and read from disk.

16. Which memory structure records all database changes made to the instance?

A. Database buffer

B. Dictionary cache

C. Redo log buffer

D. Library cache

Correct answer: C. The redo log buffer keeps track of all changes made to the database before writing them to the redo log files. The database buffer contains the data blocks that are read from the data files, and are most recently used. The dictionary cache holds the most recently used data dictionary information. The library cache holds the parsed SQL statements and PL/SQL code.

17. What is the minimum number of online redo log files required in a database?

A. One

B. Two

C. Four

D. Zero

Correct answer: B. There should be at least two redo log files in a database. The LGWR process writes to the redo log files in a circular manner, so there should be at least two files.

18. When are the system change numbers assigned?

A. When a transaction begins

B. When a transaction ends abnormally

C. When a checkpoint occurs

D. When a COMMIT is issued

Correct answer: D. A system change number (SCN) is assigned when the transaction is committed. The SCN is a unique number acting as an internal timestamp, used for recovery and read-consistent queries.

19. Which of the following is not part of the database buffer pool?

A. KEEP

B. RECYCLE

C. LIBRARY

D. DEFAULT

Correct answer: C. There is no database buffer cache named LIBRARY. The DBA can configure multiple buffer pools by using the appropriate initialization parameters for performance improvements. The KEEP buffer pool retains the data blocks in memory; they are not aged out. The RECYCLE buffer pool removes the buffers from memory as soon as they are not needed. The DEFAULT buffer pool contains the blocks that are not assigned to the other pools.

20. Memory granules are not allocated at instance startup for which of the following SGA components?

A. Database buffer cache

B. Shared pool

C. Redo log buffers

D. Large pool

E. None of the above

Correct answer: E. All of these SGA components are allocated in granule units. A minimum of three granules are allocated for the SGA at instance startup: one for the fixed portion of the SGA, one for the database buffer cache, and one for the shared pool.