

Architecture of Oracle Database Management System

Oracle is a Relational Database Management System (RDBMS), which uses Relational Data Model to store its database and SQL (commonly abbreviated as Structured Query Language) to process the stored data. The architecture of Oracle system can be best explained in terms of client/server paradigm. Thus, we will explain the architecture of Oracle server using the structure called *instance*.

An oracle *instance* is a complex set of memory structures and operating system processes. It is the Oracle instance, which manages all database activities, such as transaction processing, database recovery, form generation, and so on. The instance structure is loosely styled after UNIX's implementation of multitasking operating system. Discrete processes perform specialized tasks within the RDBMS that work together to accomplish the goals of the *instance*. Each process has a separate memory block that it uses to store local and private variables, address stacks and other runtime information. The processes use a common shared memory area for processing data concurrently. This memory block is called the System Global Area (SGA). Figure 1 illustrates the architecture of an *instance*. Each component of the instance is described below.

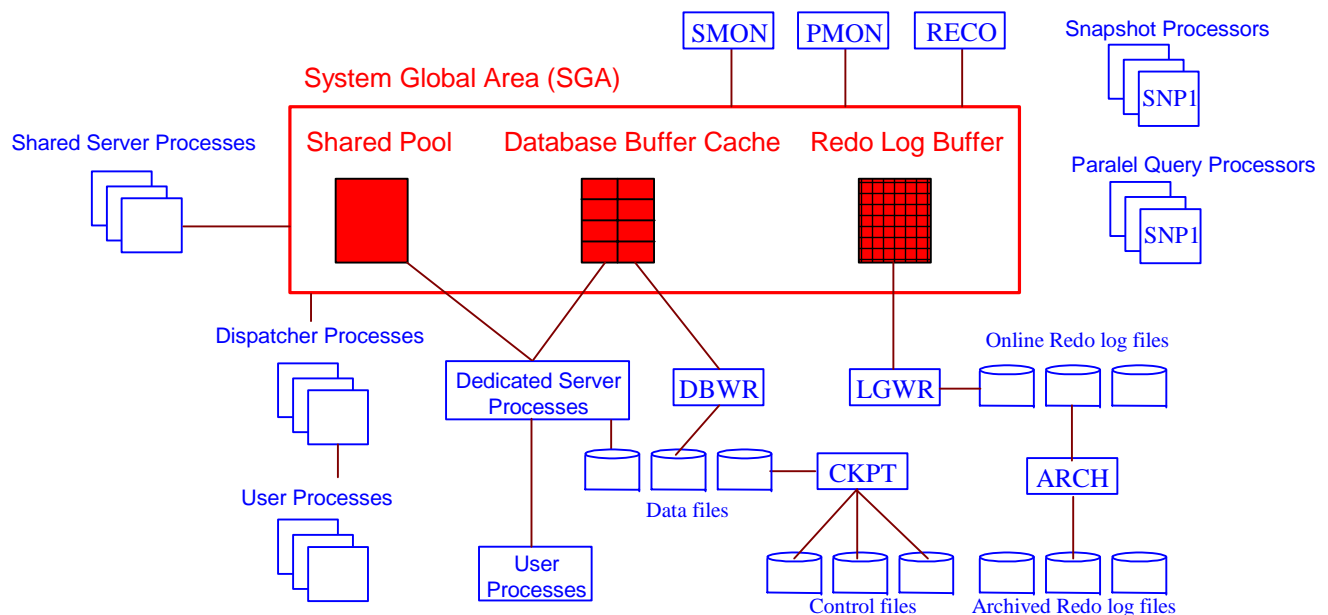


Figure 12. Oracle *instance* architecture

System Global Area (SGA)

The *SGA* is the primary memory component of the *instance*. It provides memory structure necessary for data manipulation, SQL statement parsing, and redo caching. The *SGA* is shared, which means that the multiple processes can access and modify the data

contained in it in a synchronized manner. The **SGA** consists of the following components:

- Shared pool
- Database buffer cache
- Redo log buffer
- Multithread server (MTS) structures

The Shared Pool

Figure 2 illustrates the structure of a shared pool. It contains the library cache, the dictionary cache, and server control structures (such as the database character set).

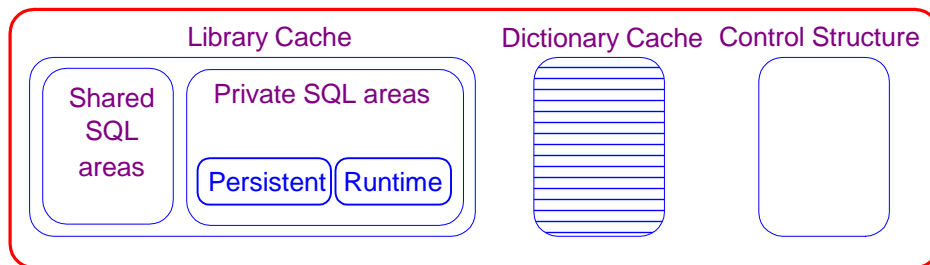


Figure 2: Oracle *Shared pool* architecture

Library cache: Stores the text, parsed format, and execution plan of SQL statements that have been submitted to the RDBMS, as well as the headers of PL/SQL packages and procedures that have been executed. For each SQL statement the server first checks the **library cache** to see if an identical statement has already been submitted and cached. If it has, then the server uses the stored parse tree and execution path for the statement, rather than building these structures from scratch.

The **library cache** has *shared* and *private* SQL areas. The *shared* area contains the parse tree and execution path for SQL statement. The *private* SQL area contains session specific information, such as bind variable, environment and session parameters, runtime stacks and buffers, etc. A *private* SQL area is created for each transaction initiated, and it is deallocated after the cursor corresponding to that *private* area is closed. Using these two structures, the server can reuse the information common across all execution of an SQL statement, while session specific information to the execution can be retrieved from the *private* SQL area.

The *private* SQL area is further divided into *persistent* and *runtime* areas. The *persistent* area contains information that is valid and applicable through multiple executions of the SQL statement. The

runtime area contains data that is used only while the SQL statement is being executed.

Dictionary cache: Stores data dictionary rows that have been used to parse SQL statements. Information such as segment information, security and access privileges, and available free storage space is held in this area.

The Database Buffer Cache

The buffer cache is composed of memory blocks. All data manipulated by Oracle server is first loaded into the buffer cache before being used. All data updates are performed in the buffer blocks

The data movement (swapping and loading) between buffer and disk or other parts of RAM is by least recently Used (LRU) algorithm. The LRU list keeps track of what data blocks are accessed and how often.

Buffer blocks that have been modified are called dirty and are placed on the dirty list. The dirty list keeps track of all data modifications made to the cache data that have not been flushed to disk. When Oracle receives a request to change data, the data change is made to the blocks in the buffer cache and written to the redo log, and then the block is put on the dirty list. Subsequent access to this data reads the new value from the changed data in the buffer cache. Dirty data from the dirty list are written to the disk database under deferred update policy.

The Redo Log Buffer

The redo log buffer is used to store redo information in memory before it is flushed to the redo log files on the disk. It is circular buffer.

The Oracle Background Process

The Oracle server process transactions concurrently. Thus, at any time there may be hundreds of simultaneous users performing a number of different operations. To accomplish these tasks, the server divides the entire workload between a number of programs, each of which operates largely independently of one another and has a specific role to play. These programs are referred to as the *Oracle background processes*. The Oracle background processes are:

- **SMON (System Monitor):** SMON is the process that performs automatic instance recovery. If the last database shutdown was not clean, SMON automatically rolls forward the operations that were complete but could not be installed in the database, and rolls back unfinished transactions. SMON process also manages certain database segments, reclaiming temporary segment space no longer in use, and automatically combining contiguous blocks of free space in the data files.

- **PMON (Process Monitor):** PMON is responsible for cleaning up terminated or failed processes, rolling back uncommitted transactions, releasing the locks held by disconnected processes, and freeing SGA resources held by failed processes. It also monitors the server and dispatcher processes, automatically starting them if they fail.
- **DBWR:** Database Writer process is responsible for writing the dirty blocks from the database buffer cache to the data files on the disk. The process waits until certain criteria are met, then reads the dirty list and writes a set of modified blocks in batch. In most installations, there is one DBWR process to handle all the write activity of the database. However, more than one DBWR process can be started if one is incapable of keeping up with the demands of the database.
- **LGWR:** Log Writer is the process that writes redo log entries from the redo log buffer in the SGA to the online log files. LGWR performs this write when a commit occurs, the inactivity timeout for LGWR is reached, the redo log buffer becomes one-third full, or DBWR completes a flush of the data buffer blocks at a checkpoint. LGWR also handles multiple user commits simultaneously, if one or more users issue a commit before LGWR has completed flushing the buffer on behalf of another user's commit. It is important to note that Oracle does not regard a transaction as being complete until LGWR has flushed the redo information from the redo buffer to the online redo logs. It is LGWR's successful writing of the redo log entries into the online redo logs, and not the changing of data in the data files, which returns a success code to the server process.
- **DISPATCHER Process (Dnnn):** The dispatcher process passes user requests to the SGA request queue and returns the server responses back to the correct user process.
- **ARCH:** The archiver process is responsible for copying full online redo logs to the archived redo log files. While the archiver is copying the redo log, no other processes can write to the log. This is important to keep in mind, because of the circular nature of the redo logs. If the database needs to switch redo logs but the archiver is still copying the next log in the sequence, all database activity halts until archiver finishes.
- **CKPT:** The Checkpoint Process, is an optional background process that performs the checkpoint tasks that LGWR would normally perform—namely updating the data file and control file headers with the current version information. This process reduces the amount of work on LGWR when there are frequent checkpoints occurring, frequent log switches, or many data files in the database.
- **RECO:** Recovery Process is responsible for recovering failed transactions. In distributed database systems. It is automatically started when the database is configured for distributed transactions. The RECO process operates with little or no DBA intervention when an in-doubt transaction occurs in a distributed system. The RECO process attempts to connect to the remote database and resolves the in-doubt transaction when a database connection is successful.

- **SNPn**: The Snapshot Process, handles the automatic refreshing of database snapshots and runs the database procedures scheduled through the database system's job package.
- **LCKn**: The lock process is responsible for managing and coordinating the locks held by the individual instances. Each instance in the parallel server installation has 1-10 lock processes assigned, and each instance must have the same number. This process has no purpose in a non-parallel server environment.
- **Pnnn**: Parallel query processes are named **Pnnn**. These processes are involved in parallel index creations, table creations, and queries.

USER AND SERVER PROCESSES (Snnn)

Applications and utilities access the **RDBMS** through a user process. The user process connects to a server process, which can be dedicated to one user process or shared among many. The server process parses and executes **SQL** statements that are submitted to it and returns the result sets back to the user process. It is also the process that reads data blocks from the data files into the database buffer cache.

Each process is allocated a section of memory referred to as the *Process Global Area (PGA)*. The contents of the **PGA** differ depending on what type of connection is made to the database. When a user process connects to the database via a dedicated server process, user session data, stack space, and cursor state information is stored in the **PGA**. The user session data consists of security and resource usage information; the stack space contains local variables specific to the user session; and the cursor state area contains runtime information for the cursor, including rows returned and cursor return codes. If, however, the user process connects through a shared server process, the session and cursor state information is stored within the **SGA**.