Database Concepts

8 Memory Architecture

This chapter discusses the memory architecture of an Oracle Database instance. It contains the following topics:

See Also:

Oracle Database Administrator's Guide (../../server.111/b28310/memory.htm#ADMIN00207) for instructions for configuring and managing memory

Introduction to Oracle Database Memory Structures

Oracle Database uses memory to store information such as the following:

- Program code
- Information about a connected session, even if it is not currently active
- Information needed during program execution (for example, the current state of a query from which rows are being fetched)
- Information that is shared and communicated among Oracle Database processes (for example, locking information)
- Cached data (for example, data blocks and redo log entries) that is also permanently stored on storage devices

Basic Memory Structures

The basic memory structures associated with Oracle Database include:

- Software code areas

  Software code areas are portions of memory used to store code that is being run or can be run. Oracle Database code is stored in a software area that is typically at a different location from users' programs—a more exclusive or protected location.

- System global area (SGA)

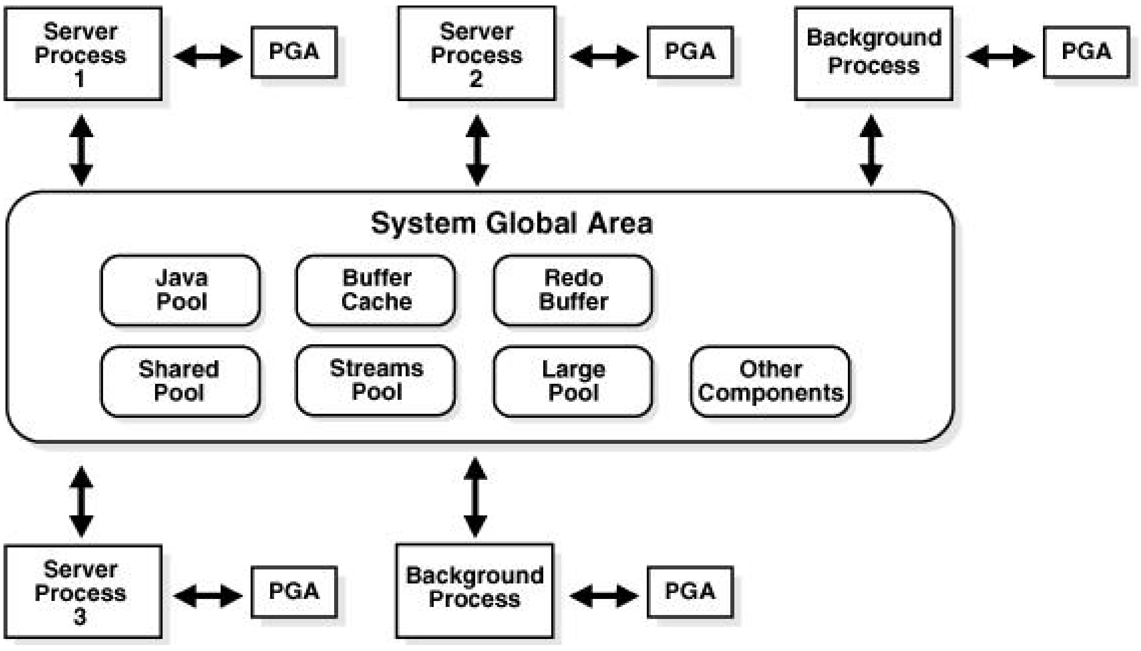
  The SGA is a group of shared memory structures, known as *SGA components*, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.

- Program global area (PGA)

  A PGA is a memory region that contains data and control information for a server process. It is nonshared memory created by Oracle Database when a server process is started. Access to the PGA is exclusive to the server process. There is one PGA for each server process. Background processes also allocate their own PGAs. The total memory used by all individual PGAs is known as the **total instance PGA memory**, and the collection of individual PGAs is referred to as the **total instance PGA**, or just **instance PGA**. You use database initialization parameters to set the size of the instance PGA, not individual PGAs.

Figure 8-1 (#CHDHAHIJ) illustrates the relationships among these memory structures.

Figure 8-1 Oracle Database Memory Structures



Description of "Figure 8-1 Oracle Database Memory Structures" (img_text/cncpt151.htm)

See Also:

- "Overview of the System Global Area" (#i10093)
- "Overview of the Program Global Area" (#i14490)
- "About Software Code Areas" (#i21266)
- "Overview of Oracle Database Processes" (process.htm#i21918)

Overview of the System Global Area

The System Global Area (SGA) and the set of database processes constitute an Oracle Database instance. Oracle Database automatically allocates memory for an SGA when you start an instance, and the operating system reclaims the memory when you shut down the instance. Each instance has its own SGA.

The SGA is read/write. All database background processes and all server processes that execute on behalf of users can read information contained within the instance's SGA, and several processes write to the SGA during database operation.

Part of the SGA contains general information about the state of the database and the instance, which the background processes need to access. This is called the **fixed SGA**. No user data is stored here. The SGA also includes information communicated between processes, such as locking information.

If the system uses shared server architecture, then the request and response queues and some contents of the PGA are in the SGA.

As shown in Figure 8-1 (#CHDHAHIJ) , the SGA consists of a number of memory **components**, which are pools of memory used to satisfy a particular class of memory allocation requests.

The most important SGA components are the following:

- Database Buffer Cache (#i10221)
- Redo Log Buffer (#i21738)
- Shared Pool (#i10223)
- Large Pool (#BGBGHJAA)
- Java Pool (#BGBGEDJG)
- Streams Pool (#BGBEFJGA)

See Also:

- "Introduction to an Oracle Instance" (startup.htm#CHDFGJIH) for more information about an Oracle Database instance
- "Overview of the Program Global Area" (#i14490)
- "Dispatcher Request and Response Queues" (process.htm#i19339)

Database Buffer Cache

The database buffer cache is the portion of the SGA that holds copies of data blocks read from datafiles. All users concurrently connected to the instance share access to the database buffer cache.

This section includes the following topics:

- Organization of the Database Buffer Cache (#BABIJIIF)
- The LRU Algorithm and Full Table Scans (#BABCCJCB)

Organization of the Database Buffer Cache

The buffers in the cache are organized in two lists: the write list and the least recently used (LRU) list. The **write list** holds *dirty* buffers, which contain data that has been modified but has not yet been written to disk. The **LRU list** holds free buffers, *pinned* buffers, and dirty buffers that have not yet been moved to the write list. **Free buffers** do not contain any useful data and are available for use. **Pinned buffers** are currently being accessed.

When an Oracle Database process accesses a buffer, the process moves the buffer to the most recently used (MRU) end of the LRU list. As more buffers are continually moved to the MRU end of the LRU list, dirty buffers age toward the LRU end of the LRU list.

The first time an Oracle Database user process requires a particular piece of data, it searches for the data in the database buffer cache. If the process finds the data already in the cache (a **cache hit**), it can read the data directly from memory. If the process cannot find the data in the cache (a **cache miss**), it must copy the data block from a datafile on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than data access through a cache miss.

Before reading a data block into the cache, the process must first find a free buffer. The process searches the LRU list, starting at the least recently used end of the list. The process searches either until it finds a free buffer or until it has searched the threshold limit of buffers.

If the user 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 disk into the buffer and moves the buffer to the MRU end of the LRU list.

If an Oracle Database user process searches the threshold limit of buffers without finding a free buffer, the process stops searching the LRU list and signals the DBW0 background process to write some of the dirty buffers to disk.

See Also:

"Database Writer Process (DBW***n***)" (process.htm#i7259) for more information about DBW*n* processes

The LRU Algorithm and Full Table Scans

When the user process is performing a full table scan, it reads the blocks of the table into buffers and puts them on the LRU end (instead of the MRU end) of the LRU list. This is because a fully scanned table usually is needed only briefly, so the blocks should be moved out quickly to leave more frequently used blocks in the cache.

You can control this default behavior of blocks involved in table scans on a table-by-table basis. To specify that blocks of the table are to be placed at the MRU end of the list during a full table scan, use the **`CACHE`** clause when creating or altering a table or cluster. You can specify this behavior for small lookup tables or large static historical tables to avoid I/O on subsequent accesses of the table.

See Also:

Oracle Database SQL Language Reference (../../server.111/b28286/toc.htm) for information about the **`CACHE`** clause

Redo Log Buffer

The **redo log buffer** is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. **Redo entries** contain the information necessary to reconstruct, or redo, changes made to the database by **`INSERT`**, **`UPDATE`**, **`DELETE`**, **`CREATE`**, **`ALTER`**, or **`DROP`** operations. Redo entries are used for database recovery, if necessary.

Redo entries are copied by Oracle Database processes from the user's memory space to the redo log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The background process LGWR writes the redo log buffer to the active redo log file (or group of files) on disk.

See Also:

- "Log Writer Process (LGWR)" (process.htm#i7261) for more information about how the redo log buffer is written to disk
- *Oracle Database Backup and Recovery User's Guide* (../../backup.111/b28270/toc.htm) for information about redo log files and groups

Shared Pool

The shared pool portion of the SGA contains the library cache, the dictionary cache, the result cache, buffers for parallel execution messages, and control structures.

This section includes the following topics:

- Library Cache (#i11734)
- Dictionary Cache (#BABEJJEG)
- Result Cache (#BGBEEFBE)

Library Cache

The library cache includes the shared SQL areas, private SQL areas (in the case of a shared server configuration), PL/SQL procedures and packages, and control structures such as locks and library cache handles.

Shared SQL areas are accessible to all users, so the library cache is contained in the shared pool within the SGA.

Shared SQL Areas and Private SQL Areas

Oracle Database represents each SQL statement it runs with a shared SQL area and a private SQL area. Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users. However, each user must have a separate copy of the statement's private SQL area.

A shared SQL area contains the parse tree and execution plan for a given SQL statement. Oracle Database saves memory by using one shared SQL area for SQL statements run multiple times, which often happens when many users run the same application.

Oracle Database allocates memory from the shared pool when a new SQL statement is parsed, to store in the shared SQL area. The size of this memory depends on the complexity of the statement. If the entire shared pool has already been allocated, Oracle Database can deallocate items from the pool using a modified LRU (least recently used) algorithm until there is enough free space for the new statement's shared SQL area. If Oracle Database deallocates a shared SQL area, the associated SQL statement must be reparsed and reassigned to another shared SQL area at its next execution.

See Also:

- "Private SQL Area" (#i17716)
- *Oracle Database Performance Tuning Guide (../../server.111/b28274/toc.htm)*

PL/SQL Program Units and the Shared Pool

Oracle Database processes PL/SQL program units (procedures, functions, packages, anonymous blocks, and database triggers) much the same way it processes individual SQL statements. Oracle Database allocates a shared area to hold the parsed, compiled form of a program unit. Oracle Database allocates a private area to hold values specific to the session that runs the program unit, including local, global, and package variables (also known as package instantiation) and buffers for executing SQL. If more than one user runs the same program unit, then a single, shared area is used by all users, while each user maintains a separate copy of his or her private SQL area, holding values specific to his or her session.

Individual SQL statements contained within a PL/SQL program unit are processed as described in the previous sections. Despite their origins within a PL/SQL program unit, these SQL statements use a shared area to hold their parsed representations and a private area for each session that runs the statement.

Allocation and Reuse of Memory in the Shared Pool

In general, any item (shared SQL area or dictionary row) in the shared pool remains until it is flushed according to a modified LRU algorithm. The memory for items that are not being used regularly is freed if space is required for new items that must be allocated some space in the shared pool. A modified LRU algorithm allows shared pool items that are used by many sessions to remain in memory as long as they are useful, even if the process that originally created the item terminates. As a result, the overhead and processing of SQL statements associated with a multiuser Oracle Database system is minimized.

When a SQL statement is submitted to Oracle Database for execution, Oracle Database automatically performs the following memory allocation steps:

1. Oracle Database checks the shared pool to see if a shared SQL area already exists for an identical statement. If so, that shared SQL area is used for the execution of the subsequent new instances of the statement. Alternatively, if there is no shared SQL area for a statement, Oracle Database allocates a new shared SQL area in the shared pool. In either case, the user's private SQL area is associated with the shared SQL area that contains the statement.

   Note:
   A shared SQL area can be flushed from the shared pool, even if the shared SQL area corresponds to an open cursor that has not been used for some time. If the open cursor is subsequently used to run its statement, Oracle Database reparses the statement, and a new shared SQL area is allocated in the shared pool.

2. Oracle Database allocates a private SQL area on behalf of the session. The location of the private SQL area depends on the type of connection established for the session.

Oracle Database also flushes a shared SQL area from the shared pool in these circumstances:

- When the `ANALYZE` statement is used to update or delete the statistics of a table, cluster, or index, all shared SQL areas that contain statements referencing the analyzed schema object are flushed from the shared pool. The next time a flushed statement is run, the statement is parsed in a new shared SQL area to reflect the new statistics for the schema object.

- If a schema object is referenced in a SQL statement and that object is later modified in any way, the shared SQL area is **invalidated** (marked invalid), and the statement must be reparsed the next time it is run.

- If you change a database's global database name, all information is flushed from the shared pool.

- The administrator can manually flush all information in the shared pool to assess the performance (with respect to the shared pool, not the data buffer cache) that can be expected after instance startup without shutting down the current instance. The statement `ALTER SYSTEM FLUSH SHARED_POOL` is used to do this.

See Also:

- "Shared SQL Areas and Private SQL Areas" (#i9081) for more information about the location of the private SQL area

- Chapter 6, "Schema Object Dependencies" (dependencies.htm#CHDFADFI) for more information about the invalidation of SQL statements and dependency issues

- *Oracle Database SQL Language Reference (../../server.111/b28286/statements_2.htm#SQLRF009)* for information about using **`ALTER SYSTEM FLUSH SHARED_POOL`**

- *Oracle Database Reference* (../../server.111/b28320/dynviews_part.htm#REFRN003) for information about **`V$SQL`** and **`V$SQLAREA`** dynamic views

Dictionary Cache

The data dictionary is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

The data dictionary is accessed so often by Oracle Database that two special locations in memory are designated to hold dictionary data. One area is called the **data dictionary cache**, also known as the **row cache** because it holds data as rows instead of buffers (which hold entire blocks of data). The other area in memory to hold dictionary data is the library cache. All Oracle Database user processes share these two caches for access to data dictionary information.

See Also:

Chapter 7, "The Data Dictionary" (datadict.htm#g6891)

Result Cache

The result cache is composed of the SQL query result cache and PL/SQL function result cache, which share the same infrastructure.

The **`DBMS_RESULT_CACHE`** package provides administration subprograms, which, for example, flush all cached results and turn result-caching on or off systemwide. The dynamic performance views **`V$RESULT_CACHE_*`** allow the developer and DBA to determine, for example, the cache-hit success for a certain SQL query or PL/SQL function.

Similar to the result cache, the client result cache also caches results, except that the caching is done on the client side.

See Also:

- *Oracle Database Administrator's Guide* (../../server.111/b28310/memory004.htm#ADMIN11228) for information about sizing the result cache
- *Oracle Database PL/SQL Packages and Types Reference* (../../appdev.111/b28419/d_result_cache.htm#ARPLS202) for information about the **`DBMS_RESULT_CACHE`** package
- *Oracle Database Reference* (../../server.111/b28320/dynviews_part.htm#REFRN003) for information about dynamic performance ($V) views
- *Oracle Call Interface Programmer's Guide* (http://www.oracle.com/pls/topic/lookup?ctx=db111&id=LNOCI10103) for more information about the client result cache

SQL Query Result Cache

Results of queries and query fragments can be cached in memory in the SQL query result cache. The database can then use cached results to answer future executions of these queries and query fragments. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached. Users can annotate a query or query fragment with a *result cache hint* to indicate that results are to be stored in the SQL query result cache.

You can set the **`RESULT_CACHE_MODE`** initialization parameter to control whether the SQL query result cache is used for all queries (when possible), or only for queries that are annotated.

The database automatically invalidates a cached result whenever a transaction modifies the data or metadata of any of the database objects used to construct that cached result.

> **See Also:**
>
> *Oracle Database Performance Tuning Guide* (../../server.111/b28274/memory.htm#PFGRF10121) for information about the **`RESULT_CACHE_MODE`** initialization parameter

PL/SQL Function Result Cache

A PL/SQL function is sometimes used to return the result of a computation whose inputs are one or several parameterized queries issued by the function. In some cases, these queries access data (for example, the catalog of wares in a shopping application) that changes very infrequently compared to the frequency of calling the function. You can include syntax in the source text of a PL/SQL function to request that its results be cached and, to ensure correctness, that the cache be purged when any of a list of tables experiences DML. The look-up key for the cache is the combination of actual arguments with which the function is invoked. When a particular invocation of the result-cached function is a cache hit, then the function body is not executed; instead, the cached value is returned immediately.

> **See Also:**
>
> *Oracle Database PL/SQL Language Reference (../../appdev.111/b28370/subprograms.htm#LNPLS00817)* for more information about the PL/SQL function result cache

Large Pool

The database administrator can configure an optional memory area called the **large pool** to provide large memory allocations for:

- Session memory for the shared server and the Oracle XA interface (used where transactions interact with more than one database)
- I/O server processes
- Oracle Database backup and restore operations

By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead caused by shrinking the shared SQL cache.

In addition, the memory for Oracle Database backup and restore operations, for I/O server processes, and for parallel buffers is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool.

The large pool does not have an LRU list. It is different from reserved space in the shared pool, which uses the same LRU list as other memory allocated from the shared pool.

> **See Also:**
>
> - "Shared Server Architecture" (process.htm#i19036) for information about allocating session memory from the large pool for the shared server
> - *Oracle Database Advanced Application Developer's Guide (../../appdev.111/b28424/toc.htm)* for information about Oracle XA
> - *Oracle Database Performance Tuning Guide* (../../server.111/b28274/toc.htm) for more information about the large pool, reserve space in the shared pool, and I/O server processes
> - "Overview of Parallel Execution" (bus_intl.htm#i32362) for information about allocating memory for parallel execution

Java Pool

Java pool memory is used in server memory for all session-specific Java code and data within the JVM. Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

The Java Pool Advisor statistics provide information about library cache memory used for Java and predict how changes in the size of the Java pool can affect the parse rate. The Java Pool Advisor is internally turned on when **`statistics_level`** is set to **`TYPICAL`** or higher. These statistics reset when the advisor is turned off.

> **See Also:**
>
> *Oracle Database Java Developer's Guide* (../../java.111/b31225/toc.htm)

Streams Pool

The streams pool is used exclusively by Oracle Streams. The Streams pool stores buffered queue messages, and it provides memory for Oracle Streams capture processes and apply processes.

Unless you specifically configure it, the size of the Streams pool starts at zero. The pool size grows dynamically as needed when Oracle Streams is used.

> **See Also:**
>
> *Oracle Streams Concepts and Administration* (../../server.111/b28321/toc.htm)

Overview of the Program Global Area

Oracle Database allocates a program global area (PGA) for each server process. The PGA is used to process SQL statements and to hold logon and other session information. For the purposes of memory management, the collection of all PGAs is known as the **instance PGA**. Using an initialization parameter, you set the size of the instance PGA, and the database distributes memory to individual PGAs as needed.

Note:

Background processes also allocate their own PGAs. This discussion focuses on server process PGAs only.

This section contains the following topics:

- Content of the PGA (#BGBGEAGD)
- PGA Memory Use in Dedicated and Shared Server Modes (#BGBBGGHE)

See Also:

"Connections and Sessions" (process.htm#i18532) for information about sessions

Content of the PGA

The content of the PGA memory varies, depending on whether or not the instance is running the shared server option. Generally speaking, the PGA memory is divided into the following areas:

- Session Memory (#BGBJAIDF)
- Private SQL Area (#i17716)

Session Memory

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

See Also:

- "Connections and Sessions" (process.htm#i18532) for more information about sessions
- *Oracle Database Net Services Administrator's Guide (../../network.111/b28316/toc.htm)*

Private SQL Area

The private SQL area contains data such as bind variable values, query execution state information, and query execution work areas. 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 location of a private SQL area depends on the type of connection established for a session. If a session is connected through a dedicated server, private SQL areas are located in the server process's PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.

This section includes the following topics:

- Cursors and SQL Areas (#BABEECCB)
- Private SQL Area Components (#BABGEDCH)

- SQL Work Areas (#i20763)

> **See Also:**
>
> "Shared SQL Areas and Private SQL Areas" (#i9081)

Cursors and SQL Areas

The application developer of an Oracle Database precompiler program or OCI program can explicitly open **cursors**, or handles to specific private SQL areas, and use them as a named resource throughout the execution of the program. Recursive cursors that Oracle Database issues implicitly for some SQL statements also use shared SQL areas.

The management of private SQL areas is the responsibility of the user process. The allocation and deallocation of private SQL areas depends largely on which application tool you are using, although the number of private SQL areas that a user process can allocate is always limited by the initialization parameter **`OPEN_CURSORS`**. The default value of this parameter is 50.

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

> **See Also:**
>
> "Cursors" (sqlplsql.htm#i1209)

Private SQL Area Components

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

- **The persistent area**—This area contains bind variable values. It is freed only when the cursor is closed.
- **The runtime area**—Oracle Database creates this area as the first step of an execute request. It contains the following structures:
  - Query execution state information

    For example, for a full table scan, this area contains information on the progress of the scan
  - SQL work areas

    These areas are allocated as needed for memory-intensive operations like sorting or hash-joins. More detail is provided later in this section.

  For DML, the run-time area is freed when the statement finishes running. For queries, it is freed after all rows are fetched or the query is canceled.

SQL Work Areas

SQL work areas are allocated to support memory-intensive operators such as the following:

- Sort-based operators (order by, group-by, rollup, window function)
- Hash-join
- Bitmap merge
- Bitmap create

For example, a sort operator uses a work area (sometimes called the sort area) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (also called the hash area) to build a hash table from its left input. If the amount of data to be processed by these two operators does not fit into a work area, the input data is divided into smaller pieces. This enables some data pieces to be processed in memory while the rest are spilled to temporary disk storage to be processed later. Although bitmap operators do not spill to disk when their associated work area is too small, their complexity is inversely proportional to the size of their work area. Thus, these operators run faster with larger work area.

The size of a work area can be controlled and tuned. The database automatically tunes work area sizes when automatic PGA memory management is enabled. See "Overview of Memory Management Methods" (#BGBDFIGD) for more information.

Generally, bigger work areas can significantly improve the performance of a particular operator at the cost of higher memory consumption. Optimally, the size of a work area is big enough to accommodate the input data and auxiliary memory structures allocated by its associated SQL operator. If not, response time increases, because part of the input data must be spilled to temporary disk storage. In the extreme case, if the size of a work area is far too small compared to the input data size, multiple passes over the data pieces must be performed. This can dramatically increase the response time of the operator.

PGA Memory Use in Dedicated and Shared Server Modes

PGA memory allocation depends, in some specifics, on whether the system uses dedicated or shared server architecture. Table 8-1 (#g26073) shows the differences.

Table 8-1 Differences in Memory Allocation Between Dedicated and Shared Servers

Memory Area	**Dedicated Server**	**Shared Server**
Nature of session memory	Private	Shared
Location of the persistent area	PGA	SGA
Location of part of the run-time area for `SELECT` statements	PGA	PGA
Location of the run-time area for DML/DDL statements	PGA	PGA

Overview of Memory Management Methods

Memory management involves maintaining optimal sizes for the Oracle database instance memory structures as demands on the database change. The memory that must be managed is the system global area (SGA) memory and the instance program global area (instance PGA) memory. The instance PGA memory is the collection of memory allocations for all individual PGAs.

Oracle Database supports various memory management methods, which are chosen by initialization parameter settings. Oracle recommends that you enable the *automatic memory management* method.

Automatic Memory Management – For Both the SGA and Instance PGA

Beginning with Oracle Database 11*g*, Oracle Database can manage the SGA memory and instance PGA memory completely automatically. You designate only the total memory size to be used by the instance, and Oracle Database dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands. This capability is referred to as *automatic memory management*. With this memory management method, the database also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs.

Automatic Shared Memory Management – For the SGA

If you want to exercise more direct control over the size of the SGA, you can disable automatic memory management and enable *automatic shared memory management*. With automatic shared memory management, you set target and maximum sizes for the SGA. The database then tunes the total size of the SGA to your designated target, and dynamically tunes the sizes of all SGA components.

Manual Shared Memory Management – For the SGA

If you want complete control of individual SGA component sizes, you can disable both automatic memory management and automatic shared memory management. This effectively enables *manual shared memory management*. In this mode, you set the sizes of several individual SGA components, thereby determining the overall SGA size. You then manually tune these individual SGA components on an ongoing basis.

Automatic PGA Memory Management – For the Instance PGA

When you disable automatic memory management and enable automatic shared memory management or manual shared memory management, you also implicitly enable *automatic PGA memory management*. With automatic PGA memory management, you set a target size for the instance PGA. The database then tunes the size of the instance PGA to your target, and dynamically tunes the sizes of individual PGAs. Because automatic PGA memory management is the default method for the instance PGA, if you do not explicitly set a target size, the database automatically computes and configures a reasonable default.

Manual PGA Memory Management – For the Instance PGA

Previous releases of Oracle Database required the DBA to manually specify the maximum work area size for each type of SQL operator (such as sort or hash-join). This proved to be very difficult, because the workload is always changing. Although the current release of Oracle Database supports this manual PGA memory management method, Oracle strongly recommends that you leave automatic PGA memory management enabled.

Summary of Memory Management Methods

Table 8-2 (#CHDFFHGH) summarizes the various memory management methods. If you do not enable automatic memory management, you must separately configure one memory management method for the SGA and one for the PGA.

Note:

When automatic memory management is not enabled, the default method for the instance PGA is automatic PGA memory management.

Table 8-2 Oracle Database Memory Management Modes

Memory Management Mode	For	You Set	Oracle Database Automatically Tunes
Automatic memory management	SGA and PGA	• Total memory target size for the Oracle instance • (Optional) Maximum memory size for the Oracle instance	• Total SGA size • SGA component sizes • Instance PGA size • Individual PGA sizes
Automatic shared memory management (Automatic memory management disabled)	SGA	• SGA target size • (Optional) SGA maximum size	SGA component sizes
Manual shared memory management (Automatic memory management and automatic shared memory management disabled)	SGA	• Shared pool size • Buffer cache size • Java pool size • Large pool size	-
Automatic PGA memory management	PGA	Instance PGA target size	Individual PGA sizes
Manual PGA memory management (not recommended)	PGA	Maximum work area size for each type of SQL operator	-

See Also:

Oracle Database Administrator's Guide (../../server.111/b28310/memory005.htm#ADMIN11236) because automatic memory management is not available on all platforms

Memory Management Options and Defaults for Database Installation

If you create your database with Database Configuration Assistant (DBCA) and choose the basic installation option, automatic memory management is enabled by default. If you choose advanced installation, Database Configuration Assistant (DBCA) enables you to select from the following three memory management configurations:

- Automatic memory management
- Automatic shared memory management + automatic PGA memory management
- Manual shared memory management + automatic PGA memory management

If you create the database with a `CREATE DATABASE` SQL statement and do not choose the memory management mode by specifying the required memory initialization parameters, manual shared memory management and automatic PGA memory management are configured by default.

See also:

Oracle Database Administrator's Guide (../../server.111/b28310/memory.htm#ADMIN00207) for more information about memory management and about memory management initialization parameters.

About Software Code Areas

Software code areas are portions of memory used to store code that is being run or can be run. Oracle Database code is stored in a software area that is typically at a different location from users' programs—a more exclusive or protected location.

Software areas are usually static in size, changing only when software is updated or reinstalled. The required size of these areas varies by operating system.

Software areas are read only and can be installed shared or nonshared. When possible, Oracle Database code is shared so that all users can access it without having multiple copies in memory. This results in a saving of real main memory and improves overall performance.

User programs can be shared or nonshared. Some Oracle tools and utilities (such as Oracle Forms and SQL*Plus) can be installed shared, but some cannot. Multiple instances of Oracle Database can use the same Oracle Database code area with different databases if running on the same computer.

Note:

The option of installing software shared is not available for all operating systems (for example, on PCs operating Windows).
See your Oracle Database operating system-specific documentation for more information.

< (https://docs.oracle.com/cd/B28359_01/server.111/b28318/datadict.htm) > (https://docs.oracle.com/cd/B28359_01/server.111/b28318/process.htm)