ORACLE
UNIVERSITY

Oracle Database 12*c*: Administration Workshop

Student Guide – Volume I

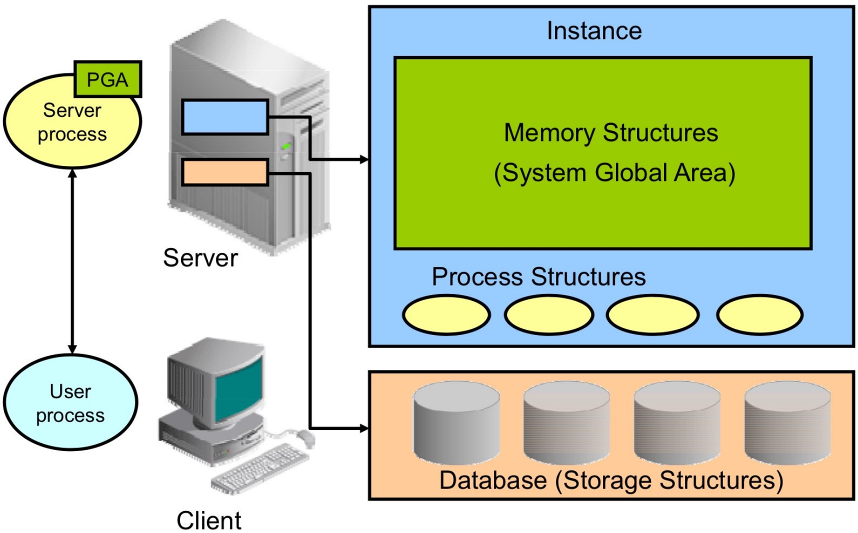
D78846GC20
Edition 2.0 | December 2014 | D89298

Learn more from Oracle University at **oracle.com/education/**

ORACLE

Exploring Oracle Database Architecture

Oracle Database Server Architecture: Overview



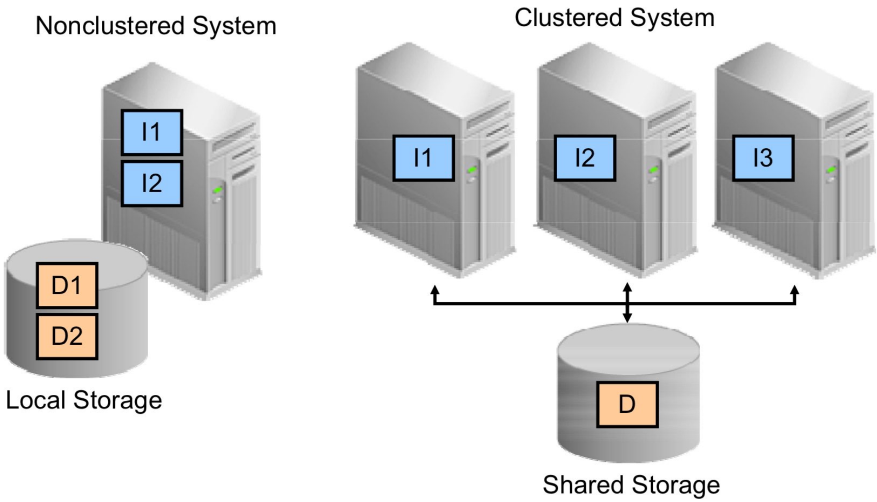
There are three major structures in Oracle Database server architecture: memory structures, process structures, and storage structures. A basic Oracle database system consists of an Oracle database and a database instance.

The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

The instance consists of memory structures and background processes associated with that instance. Every time an instance is started, a shared memory area called the System Global Area (SGA) is allocated and the background processes are started. Processes are jobs that work in the memory of computers. A process is defined as a “thread of control” or a mechanism in an operating system that can run a series of steps. After starting a database instance, the Oracle software associates the instance with a specific database. This is called *mounting the database*. The database is then ready to be opened, which makes it accessible to authorized users.

Note: Oracle Automatic Storage Management (ASM) uses the concept of an instance for the memory and process components, but is not associated with a specific database.

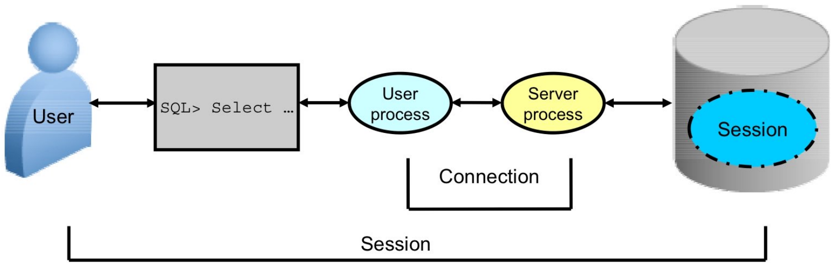
Oracle Database Instance Configurations



Each database instance is associated with one and only one database. If there are multiple databases on the same server, then there is a separate and distinct database instance for each database. A database instance cannot be shared. A Real Applications Cluster (RAC) database usually has multiple instances on separate servers for the same shared database. In this model, the same database is associated with each RAC instance, which meets the requirement that, at most, only one database is associated with an instance.

Connecting to the Database Instance

- Connection: Communication between a user process and an instance
- Session: Specific connection of a user to an instance through a user process



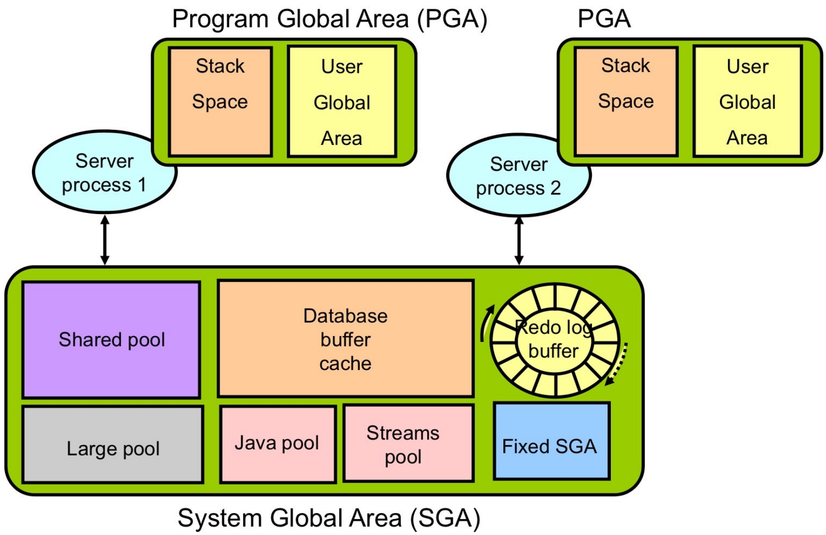
Connections and sessions are closely related to user processes but are very different in meaning.

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

A *session* represents the state of a current user login to the database instance. 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 a user connects until the user disconnects or exits the database application.

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

Oracle Database Memory Structures



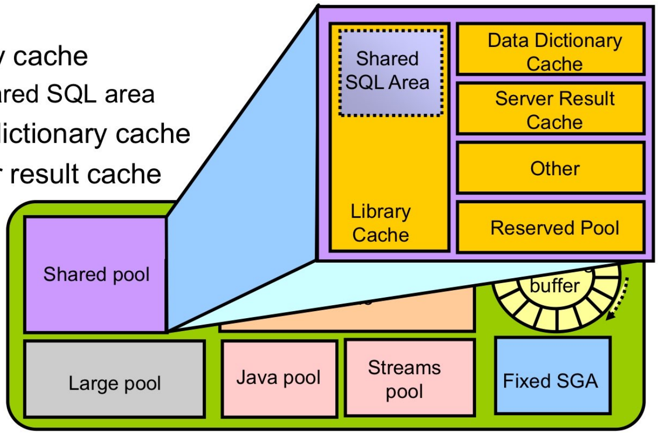
Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.

Two basic memory structures are associated with an instance:

- **System Global Area (SGA):** Group of shared memory structures, known as SGA components, which 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 Areas (PGA):** Memory regions that contain data and control information for a server or background process. A PGA is nonshared memory created by Oracle Database when a server or background process is started. Access to the PGA is exclusive to the server process. Each server process and background process has its own PGA.

Shared Pool

- Is a portion of the SGA
- Contains:
  - Library cache
    - Shared SQL area
  - Data dictionary cache
  - Server result cache



System Global Area (SGA)

The shared pool portion of the SGA contains the library cache, the data dictionary cache, the server result cache containing the SQL query result cache and the PL/SQL function result cache, buffers for parallel execution messages, and control structures.

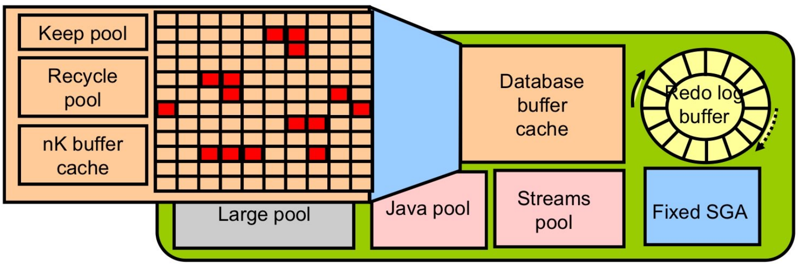
The *data dictionary* is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

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 (buffers hold entire blocks of data). The other area in memory that holds dictionary data is the *library cache*. All Oracle Database user processes share these two caches for access to data dictionary information.

Oracle Database represents each SQL statement that it runs with a shared SQL area (as well as a private SQL area kept in the PGA). Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users.

Database Buffer Cache

- Is part of the SGA
- Holds copies of data blocks that are read from data files
- Is shared by all concurrent users



System Global Area (SGA)

The database buffer cache is the portion of the SGA that holds block images read from the data files or constructed dynamically to satisfy the read consistency model. All users who are concurrently connected to the instance share access to the database buffer cache.

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 data file on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than accessing data through a cache miss.

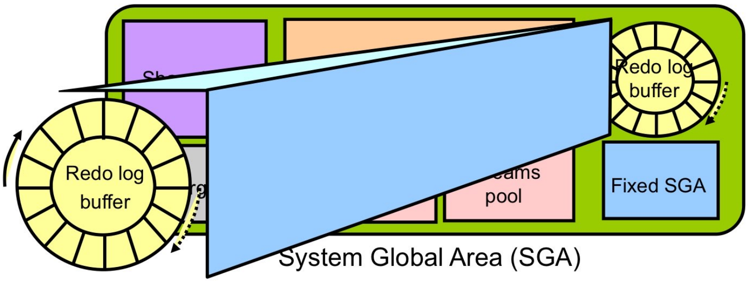
The buffers in the cache are managed by a complex algorithm that uses a combination of least recently used (LRU) lists and touch count. The LRU helps to ensure that the most recently used blocks tend to stay in memory to minimize disk access.

The keep buffer pool and the recycle buffer pool are used for specialized buffer pool tuning. The keep buffer pool is designed to retain buffers in memory longer than the LRU would normally retain them. The recycle buffer pool is designed to flush buffers from memory faster than the LRU normally would.

Additional buffer caches can be configured to hold blocks of a size that is different from the default block size.

Redo Log Buffer

- Is a circular buffer in the SGA
- Holds information about changes made to the database
- Contains redo entries that have the information to redo changes made by operations such as DML and DDL



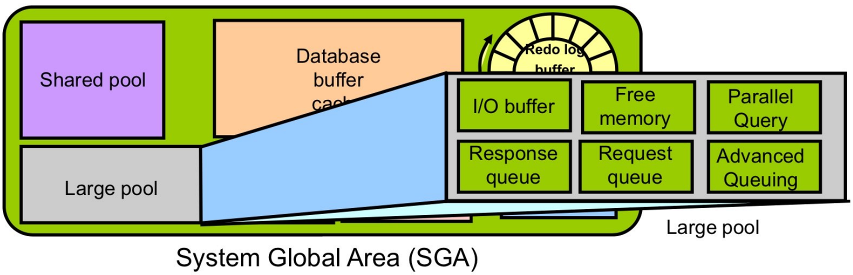
The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct (or redo) changes that are made to the database by DML, DDL, or internal operations. Redo entries are used for database recovery if necessary.

As the server process makes changes to the buffer cache, redo entries are generated and written to the redo log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The log writer background process writes the redo log buffer to the active redo log file (or group of files) on disk.

Large Pool

Provides large memory allocations for:

- Session memory for the shared server and the Oracle XA interface
- I/O server processes
- Oracle Database backup and restore operations



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 multiple databases)
- I/O server processes
- Oracle Database backup and restore operations
- Parallel Query operations
- Advanced Queuing memory table storage

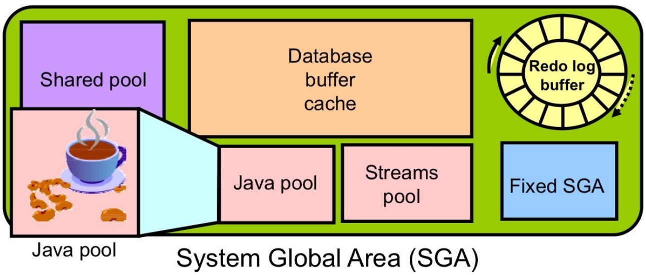
By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead that is caused by shrinking the shared SQL cache.

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 is not managed by a least recently used (LRU) list.

Java Pool

Java pool memory is used to store all session-specific Java code and data in the JVM.

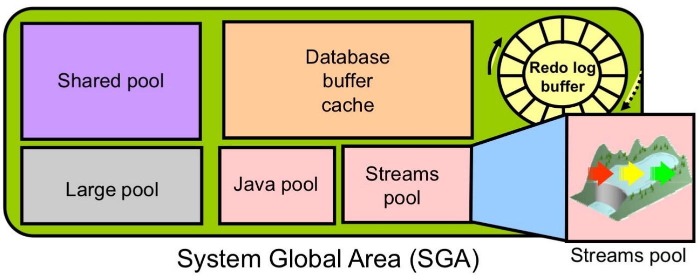


Java pool memory is used to store all session-specific Java code and data in the Java Virtual Machine (JVM). Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

Streams Pool

Streams pool memory is used exclusively by Oracle Streams to:

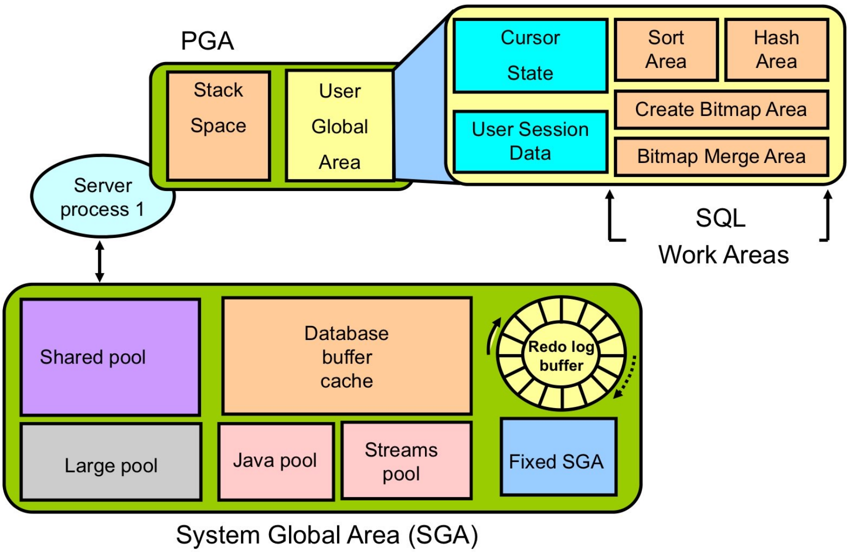
- Store buffered queue messages
- Provide memory for Oracle Streams processes



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.

Program Global Area (PGA)



The Program Global Area (PGA) is a private memory region containing data and control information for a server process. Each server process has a distinct PGA. Access to it is exclusive to that server process and it is read only by Oracle code acting on behalf of it. It is not available for developer’s code.

Every PGA contains stack space. In a dedicated server environment, each user connecting to the database instance has a separate server process. For this type of connection, the PGA contains a subdivision of memory known as the user global area (UGA). The UGA is composed of the following:

- Cursor area for storing runtime information on cursors
- User session data storage area for control information about a session
- SQL working areas for processing SQL statements consisting of:
  - A sort area for functions that order data such as `ORDER BY` and `GROUP BY`
  - A hash area for performing hash joins of tables
  - A create bitmap area used in bitmap index creation common to data warehouses
  - A bitmap merge area used for resolving bitmap index plan execution

In a shared server environment, multiple client users share the server process. In this model, the UGA is moved into the SGA (shared pool or large pool if configured) leaving the PGA with only stack space.

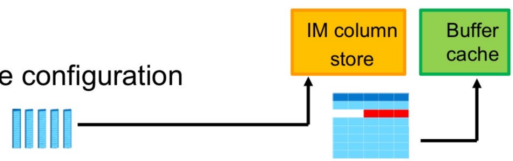
12.1.0.2

In-Memory Column Store: Introduction

- Instant query response:
  - Faster queries on very large tables on any columns (100x)
  - Use of scans, joins, and aggregates
  - Without indexes
  - Best suited for analytics: few columns, many rows
- Faster DML: Removal of most analytics indexes (3 to 4x)
- Full application transparency

ORACLE E-BUSINESS SUITE · ORACLE FUSION APPLICATIONS · ORACLE SIEBEL · ORACLE JD EDWARDS · ORACLE PEOPLESOFT

- Easy setup:
  - In-memory column store configuration
  - Segment attributes



The In-Memory Column Store feature enables objects (tables, partitions, and other types) to be stored in memory in a new format known as the *columnar format*. This format enables scans, joins, and aggregates to perform much faster than the traditional on-disk format, thus providing fast reporting and DML performance for both OLTP and DW environments. This is particularly useful for analytic applications that operate on few columns returning many rows rather than for OLTP that operates on few rows returning many columns. The DBA must define the segments that are to be populated into the in-memory column store (IM column store), such as hot tables, partitions, and more precisely the more frequently accessed columns.

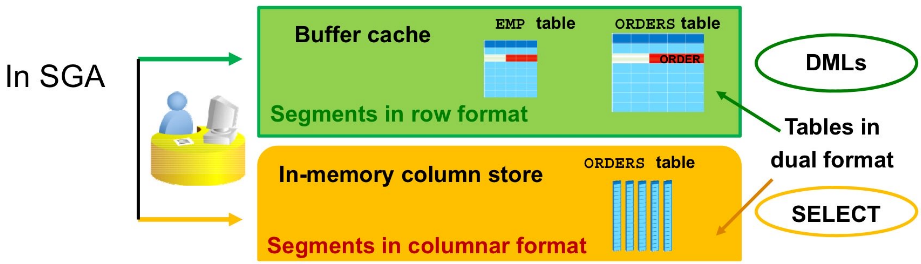
The in-memory columnar format does not replace the on-disk or buffer cache format. It is a consistent copy of a table or of some columns of a table converted to the new columnar format that is independent of the disk format and only available in memory. Because of this independence, applications are able to transparently use this option without any changes. For the data to be converted into the new columnar format, a new pool is requested in the SGA. The pool is the IM column store.

If sufficient space is allocated for the IM column store, a query that accesses objects that are candidates to be populated into the IM column store performs much faster. The improved performance allows ad hoc analytic queries to be executed directly on the real-time transaction data without impacting the existing workload.

12.1.0.2

In-Memory Column Store: Overview

- A new pool in the SGA: In-Memory column store
  - Segments populated into the IM column store are converted into a columnar format.
  - In-Memory segments are transactionally consistent with the buffer cache.
- Only one segment on disk and in row format



The in-memory columnar format does not replace the on-disk or buffer cache format. This means that when a segment such as a table or a partition is populated into the IM column store, the on-disk format segment is automatically converted into a columnar format and optionally compressed. The columnar format is a pure in-memory format. There is no columnar format storage on disk. It never causes additional writes to disk and therefore does not require any logging or undo space.

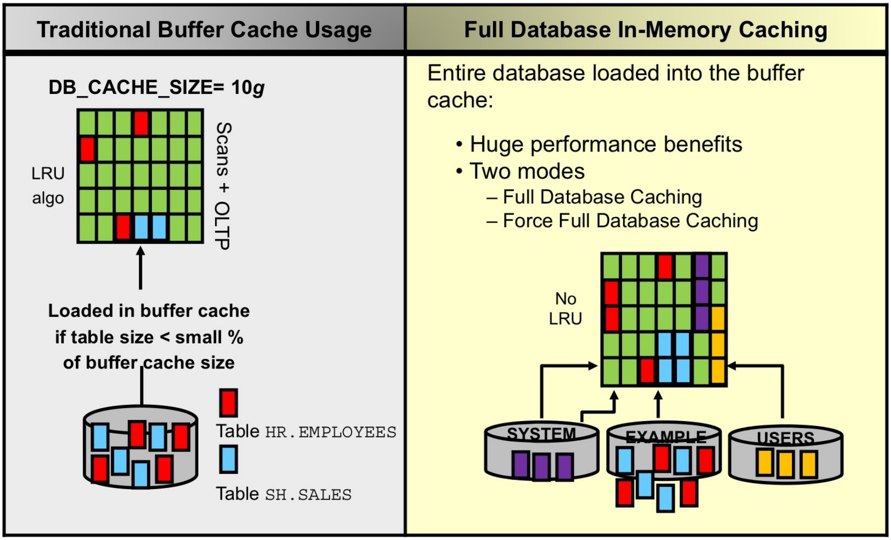
All data is stored on disk in the traditional row format.

Moreover, the columnar format of a segment is a transaction-consistent copy of the segment either on disk or in the buffer cache. Transaction consistency between the two pools is maintained.

If sufficient space is allocated to the IM column store in SGA, a query that accesses objects that are populated into the IM column store performs much faster. The improved performance allows more ad hoc analytic queries to be executed directly on real-time transaction data without impacting the existing workload. A lack of IM column store space does not prevent statements from executing against tables that could have been populated into the IM column store.

12.1.0.2

Full Database In-Memory Caching



The current algorithm for table scans loads a table into the buffer cache only when the table size is less than a small percent of the buffer cache size. For very large tables, the database uses a direct path read, which loads blocks directly into the PGA and bypasses the SGA, to avoid flooding the buffer cache. The DBA must explicitly declare small lookup tables, which are accessed frequently, as `CACHE` to load data into memory and avoid bypassing the SGA. This clause indicates that the blocks retrieved for these tables are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.

The Full Database In-memory Caching feature enables an entire database to be cached in memory when the database size (sum of all data files, `SYSTEM` tablespace, LOB CACHE files minus `SYSAUX`, `TEMP`) is smaller than the buffer cache size. Caching and running a database from memory leads to huge performance benefits. Two modes can be used:

- **Full Database Caching:** Implicit default and automatic mode in which an internal calculation determines if the database can be fully cached for an instance. `NOCACHE` LOBs are not cached in Full Database Caching but in Force Full Database Caching mode even `NOCACHE` LOBs are cached.