**Module title: Database administration**

**REGNO: 22RP08068**

**NAMES: NIYOMUFASHA Mediatrice**

**DATE: 27/1/2024**

**Individual assignment:**

**Q1.Discribe oracle memory structures and background processes.**

**Q2. Describe oracle logical and physical storage structure.**

**Q1.Discribe oracle memory structures and background processes**

**Basic Memory Structures**

Oracle Database includes several memory areas, each of which contains multiple subcomponents. The SGA is a group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. All server and background processes share the SGA.

**Oracle uses memory to store various information**:

* program code being executed
* 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 processes (for example, locking information)
* cached data that is also permanently stored on peripheral memory (for example, data blocks and redo log entries)

**The basic memory structures associated with Oracle include:**

1. Software Code Areas
2. System Global Area (SGA):
3. the database buffer cache
4. the redo log buffer
5. the shared pool
6. Program Global Areas (PGA):
7. the stack areas
8. the data areas

**Sort Areas**

**System Global Area (SGA)**

**A system global area (SGA**) is a group of shared memory structures that contain data and control information for one Oracle database instance. If multiple users are concurrently connected to the same instance, the data in the instance's SGA is "shared" among the users. Consequently, the SGA is sometimes referred to as the "shared global area".

As described in "Overview of an Oracle Instance" on page 5-2, an SGA and Oracle processes constitute an Oracle instance. Oracle 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 users connected to a multiple-process database instance may read information contained within the instance's SGA, and several processes write to the SGA during execution of Oracle.

**The SGA contains the following data structures:**

1. the database buffer cache
2. the redo log buffer
3. the shared pool
4. the data dictionary cache
5. other miscellaneous information

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 multithreaded server architecture the request and response queues, and some contents of the program global areas, are in the SGA. (See "Program Global Areas (PGA)"

The Database Buffer Cache

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

The database buffer cache and the shared SQL cache are logically segmented into multiple sets. This organization into multiple sets reduces contention on multiprocessor systems.

**Organization of the Database Buffer Cache**

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

When an Oracle 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" towards the LRU end of the LRU list.

The first time an Oracle 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 dirty 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 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. For more information about the DBW0 process (or multiple DBWn 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 may want to specify this behavior for small lookup tables or large static historical tables to avoid I/O on subsequent accesses of the table.

**Size of the Database Buffer Cache**

The initialization parameter DB\_BLOCK\_BUFFERS specifies the number of buffers in the database buffer cache. Each buffer in the cache is the size of one Oracle data block (which is specified by the initialization parameter DB\_BLOCK\_SIZE); therefore, each database buffer in the cache can hold a single data block read from a datafile.

The cache has a limited size, so not all the data on disk can fit in the cache. When the cache is full, subsequent cache misses cause Oracle to write dirty data already in the cache to disk to make room for the new data. (If a buffer is not dirty, it does not need to be written to disk before a new block can be read into the buffer.) Subsequent access to any data that was written to disk results in additional cache misses.

The size of the cache affects the likelihood that a request for data will result in a cache hit. If the cache is large, it is more likely to contain the data that is requested. Increasing the size of a cache increases the percentage of data requests that result in cache hits.

**Multiple Buffer Pools**

You can configure the database buffer cache with separate buffer pools that either keep data in the buffer cache or make the buffers available for new data immediately after using the data blocks. Particular schema objects (tables, clusters, indexes, and partitions) can then be assigned to the appropriate buffer pool to control the way their data blocks age out of the cache.

The KEEP buffer pool retains the schema object's data blocks in memory.

The RECYCLE buffer pool eliminates data blocks from memory as soon as they are no longer needed.

The DEFAULT buffer pool contains data blocks from schema objects that are not assigned to any buffer pool, as well as schema objects that are explicitly assigned to the DEFAULT pool.

The initialization parameters that configure the KEEP and RECYCLE buffer pools are BUFFER\_POOL\_KEEP and BUFFER\_POOL\_RECYCLE.

**The 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 server 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 online redo log file (or group of files) on disk.

The initialization parameter LOG\_BUFFER determines the size (in bytes) of the redo log buffer. In general, larger values reduce log file I/O, particularly if transactions are long or numerous. The default setting is four times the maximum data block size for the host operating system.

**The Shared Pool**

The shared pool portion of the SGA contains three major areas: library cache, dictionary cache, and control structures.

The total size of the shared pool is determined by the initialization parameter SHARED\_POOL\_SIZE. The default value of this parameter is 3,500,000 bytes. Increasing the value of this parameter increases the amount of memory reserved for the shared pool, and therefore increases the space reserved for shared SQL areas.

**Q2. Describe oracle logical and physical storage structure**

**Physical structures** are those that can be seen and operated on from the operating system, such as the physical files that store data on disk.

**Logical structures** are created and recognized by the Oracle database server and are not known to the operating system.

In essence, the logical design is about what data the database will store and how it is related, while the physical design is about how this data will be stored and accessed in the actual computing environment.

**Logical Database structures**

Logical structures include tablespaces, schema objects, data blocks, extents and segments.

**Tablespaces**

Database is logically divided into one or more tablespaces. Each tablespace creates one or more datafiles to physically store data.

**Schema objects**

Schema objects are the structure that represents database's data. Schema objects include structures such as tables, views, sequences, stored procedures, indexes, synonyms, clusters and database links.

**Data Blocks**

Data block represents specific number of bytes of physical database space on disk.

**Extents**

An extent represents continuous data blocks that are used to store specific data information.

**Segments**

A segment is a set of extents allocated for a certain logical structure.

**Physical database structure**

The physical database structure comprises of datafiles, redo log files and control files

**Datafiles**

Datafiles contain database's data. The data of logical data structures such as tables and indexes is stored in datafiles of the database. One or more datafiles form a logical unit of database storage called a tablespace.

**Redo log files**

The purpose of these files is to record all changes made to data. These files protect database against failures.

**Control files**

Control files contain entries such as database name, name and location of datafiles and redo log files and time stamp of database creation.