Oracle Notes Part-5

CURSOR: A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the *active* set.

Two Types of Cursor :

1)Implicit Cursor

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

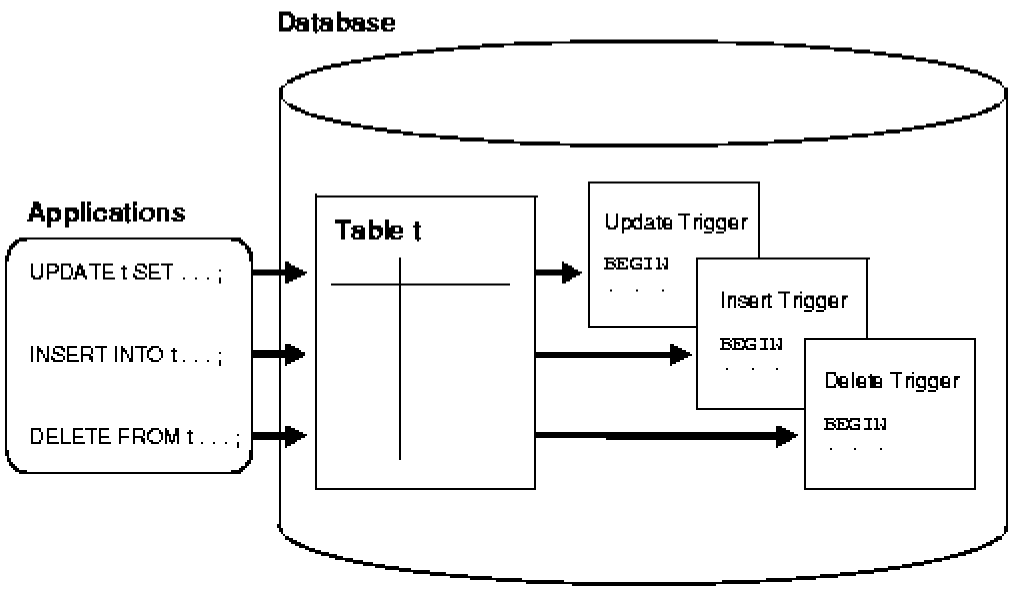
2)Explicit Cursor

They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row. When you fetch a row the current row position moves to next row.

For Example: When you execute INSERT, UPDATE, or DELETE statements the cursor attributes tell us whether any rows are affected and how many have been affected. When a SELECT… INTO statement is executed in a PL/SQL Block, implicit cursor attributes can be used to find out whether any row has been returned by the SELECT statement. PL/SQL returns an error when no data is selected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has the attributes like %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK_ROWCOUNT and %BULK_EXCEPTIONS, designed for use with the FORALL statement.

TRIGGER: **Triggers** are stored programs, which are automatically executed or fired when some events occur.Trigger automatically associated with DML statement, when DML statement execute trigger implicitly execute.You can create trigger using the CREATE TRIGGER statement. If trigger activated, implicitly fire DML statement and if trigger deactivated can't fire.



Triggers could be defined on the table, view, schema, or database with which the event is associated.

Advantages of trigger:

1) Triggers can be used as an alternative method for implementing referential integrity constraints.

2) By using triggers, business rules and transactions are easy to store in database and can be used consistently even if

there are future updates to the database.

3) It controls on which updates are allowed in a database.

4) When a change happens in a database a trigger can adjust the change to the entire database.

5) Triggers are used for calling stored procedures.

Use the CREATE TRIGGER statement to create and enable a database trigger, which is:

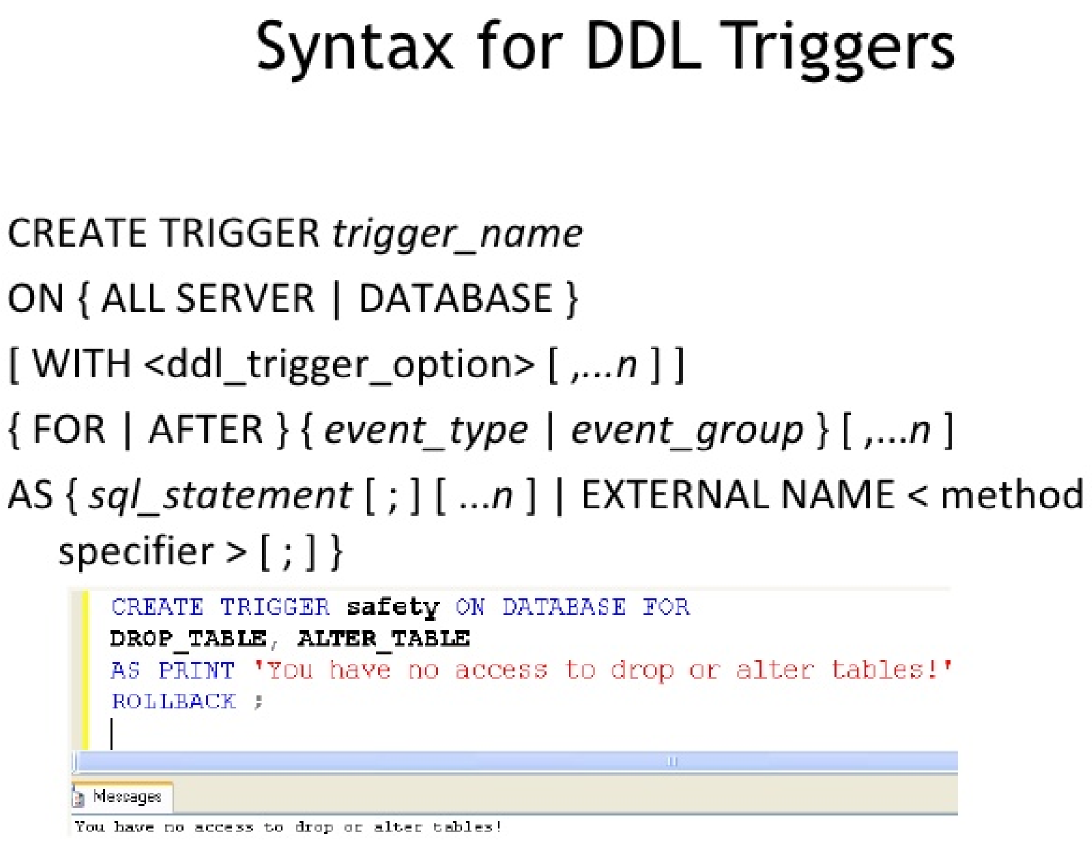
- A stored PL/SQL block associated with a table, a schema, or the database or
- An anonymous PL/SQL block or a call to a procedure implemented in PL/SQL or Java

Oracle Database automatically executes a trigger when specified conditions occur.When you create a trigger, the database enables it automatically. You can subsequently disable and enable a trigger with the DISABLE and ENABLE clause of the ALTER TRIGGER or ALTER TABLE statement.

Before a trigger can be created, the user SYS must run a SQL script commonly called DBMSSTDX.SQL. The exact name and location of this script depend on your operating system.

- To create a trigger in your own schema on a table in your own schema or on your own schema (SCHEMA), you must have the CREATE TRIGGERsystem privilege.
- To create a trigger in any schema on a table in any schema, or on another user’s schema (schema.SCHEMA), you must have the CREATE ANYTRIGGER system privilege.
- In addition to the preceding privileges, to create a trigger on DATABASE, you must have the ADMINISTER DATABASE TRIGGER system privilege.

If the trigger issues SQL statements or calls procedures or functions, then the owner of the trigger must have the privileges necessary to perform these operations. These privileges must be granted directly to the owner rather than acquired through roles.



Syntax for DDL Triggers

CREATE TRIGGER *trigger_name*
ON { ALL SERVER | DATABASE }
[WITH <ddl_trigger_option> [,...*n*]]
{ FOR | AFTER } { *event_type* | *event_group* } [,...*n*]
AS { *sql_statement* [;] [...*n*] | EXTERNAL NAME < method specifier > [;] }

```
CREATE TRIGGER safety ON DATABASE FOR
DROP_TABLE, ALTER_TABLE
AS PRINT 'You have no access to drop or alter tables!'
ROLLBACK ;
```

Messages

```
You have no access to drop or alter tables!
```

Data Blocks

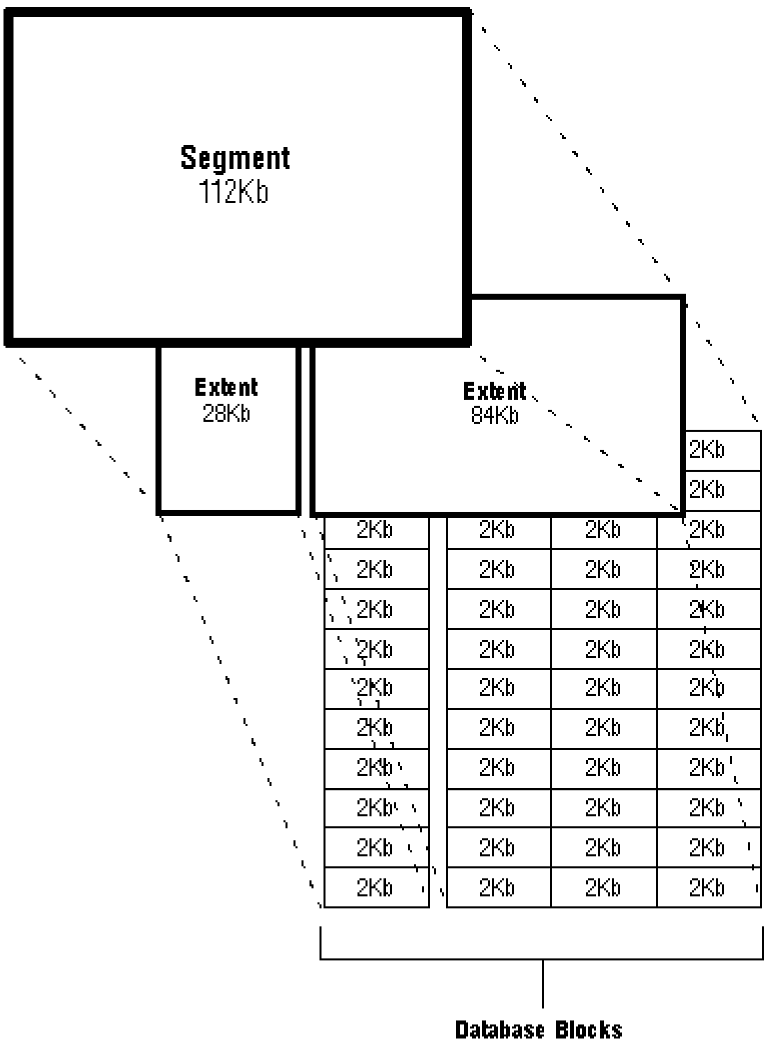
At the finest level of granularity, Oracle stores data in *data blocks* (also called logical blocks, Oracle blocks, or pages). One data block corresponds to a specific number of bytes of physical database space on disk. You set the data block size for every Oracle database when you create the database. This data block size should be a multiple of the operating system's block size within the maximum limit. Oracle data blocks are the smallest units of storage that Oracle can use or allocate.In contrast, all data at the physical, operating system level is stored in bytes. Each operating system has what is called a *block size*. Oracle requests data in multiples of Oracle blocks, not operating system blocks. Therefore, you should set the Oracle block size to a multiple of the operating system block size to avoid unnecessary I/O.

Extents

The next level of logical database space is called an *extent*. An extent is a specific number of contiguous data blocks that is allocated for storing a specific type of information.

Segments

The level of logical database storage above an extent is called a *segment*. A segment is a set of extents that have been allocated for a specific type of data structure, and that all are stored in the same tablespace. For example, each table's data is stored in its own *data segment*, while each index's data is stored in its own *index segment*.Oracle allocates space for segments in extents. Therefore, when the existing extents of a segment are full, Oracle allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk. The segments also can span files, but the individual extents cannot.



There are four types of segments used in Oracle databases:

– data segments
– index segments
– rollback segments
– temporary segments

Data Segments:
There is a single data segment to hold all the data of every non clustered table in an oracle database. This data segment is created when you create an object with the CREATE TABLE/SNAPSHOT/SNAPSHOT LOG command. Also, a data segment is created for a cluster when a CREATE CLUSTER command is issued.
The storage parameters control the way that its data segment’s extents are allocated. These affect the efficiency of data retrieval and storage for the data segment associated with the object.

Index Segments:
Every index in an Oracle database has a single index segment to hold all of its data. Oracle creates the index segment for the index when you issue the CREATE INDEX command. Setting the storage parameters directly affects the efficiency of data retrieval and storage.

Rollback Segments
Rollbacks are required when the transactions that affect the database need to be undone. Rollbacks are also needed during the time of system failures. The way the roll-backed data is saved in rollback segment, the data can also be redone which is held in redo segment.

A rollback segment is a portion of the database that records the actions of transactions if the transaction should be rolled back. Each database contains one or more rollback segments. Rollback segments are used to provide read consistency, to rollback transactions, and to recover the database.

Types of rollbacks:
– statement level rollback
– rollback to a savepoint
– rollback of a transaction due to user request
– rollback of a transaction due to abnormal process termination
– rollback of all outstanding transactions when an instance terminates abnormally
– rollback of incomplete transactions during recovery.

Temporary Segments:
The SELECT statements need a temporary storage. When queries are fired, oracle needs area to do sorting and other operation due to which temporary storages are useful.

The commands that may use temporary storage when used with SELECT are:

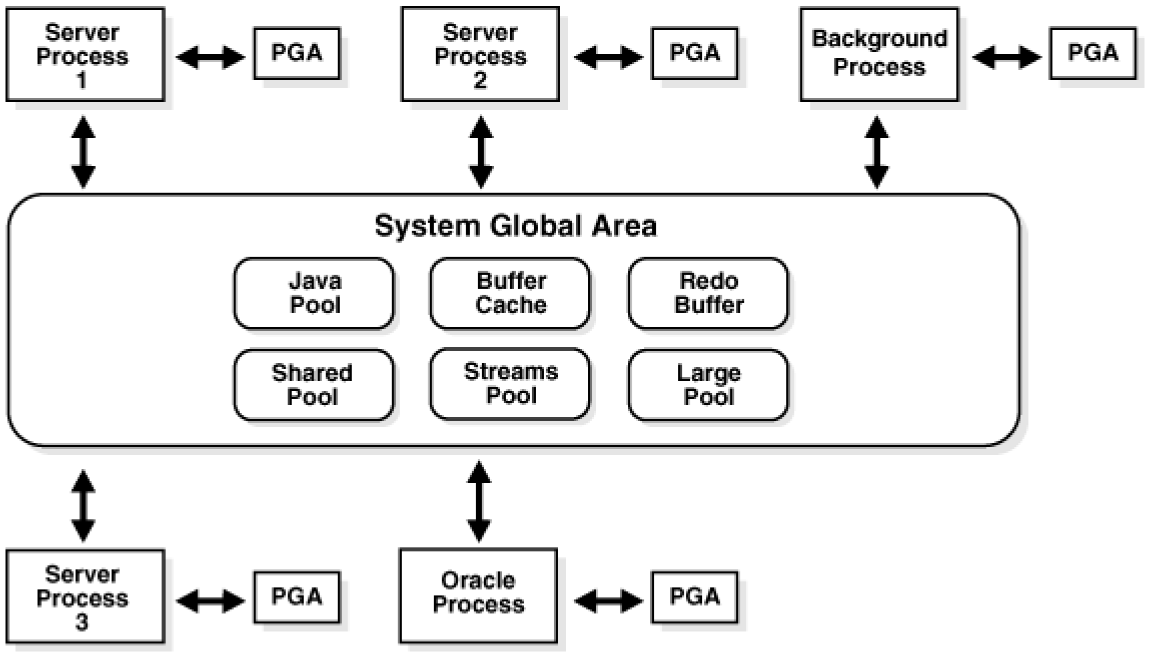
GROUP BY, UNION, DISTINCT, etc.

Oracle Notes Part-4

Oracle Memory Architecture:

Oracle memory architecture is divided in following memory structure:-

1. System Global Area (SGA):- This is a large, shared memory segment that virtually all Oracle processes will access at one point or another.
2. Process Global Area (PGA): This is memory that is private to a single process or thread; it is not accessible from other processes/threads.
3. User Global Area (UGA): This is memory associated with your session. It is located either in the SGA or the PGA, depending whether you are connected to the database using a shared server (it will be in the SGA), or a dedicated server (it will be in the PGA).



1)SGA:

There are five memory structures that make up the System Global Area (SGA). The SGA will store many internal data structures that all processes need access to, cache data from disk, cache redo data before writing to disk, hold parsed SQL plans and so on.SGA is used to store database information that is shared by database processes. It contains data and control information for the Oracle Server and is allocated in the virtual memory if the computer where Oracle resides.

SGA consists of several memory structures:-

1.Redo Buffer: The redo buffer is where data that needs to be written to the online redo logs will be cached temporarily, before it is written to disk. Since a memory-to-memory transfer is much faster than a memory-to-disk transfer, use of the redo log buffer can speed up database operation. The data will not reside in the redo buffer for very long. In fact, LGWR initiates a flush of this area in one of the following scenarios:

• Every three seconds
• Whenever someone commits

• When LGWR is asked to switch log files
• When the redo buffer gets one-third full or contains 1MB of cached redo log data

Use the parameter LOG_BUFFER parameter to adjust but be-careful increasing it too large as it will reduce your I/O but commits will take longer.

2.Buffer Cache: The block buffer cache is where Oracle stores database blocks before writing them to disk and after reading them in from disk. There are three places to store cached blocks from individual segments in the SGA:
• **Default pool (hot cache)**: The location where all segment blocks are normally cached.
• **Keep pool (warm cache)**: An alternate buffer pool where by convention you assign segments that are accessed fairly frequently, but still get aged out of the default buffer pool due to other segments needing space.
• **Recycle pool (do not care to cache)**: An alternate buffer pool where by convention you assign large segments that you access very randomly, and which would therefore cause excessive buffer flushing of many blocks from many segments. There’s no benefit to caching such segments because by the time you wanted the block again, it would have been aged out of the cache. You would separate these segments out from the segments in the default and keep pools so they would not cause those blocks to age out of the cache.

The standard block size is determined by the DB_CACHE_SIZE, if tablespaces are created with a different block sizes then you must also create an entry to match that block size.

DB_2K_CACHE_SIZE (used with tablespace block size of 2k)
DB_4K_CACHE_SIZE (used with tablespace block size of 4k)
DB_8K_CACHE_SIZE (used with tablespace block size of 8k)
DB_16K_CACHE_SIZE (used with tablespace block size of 16k)
DB_32K_CACHE_SIZE (used with tablespace block size of 32k)

3.Shared Pool: The shared pool is where Oracle caches many bits of “program” data. When we parse a query, the parsed representation is cached there. Before we go through the job of parsing an entire query, Oracle searches the shared pool to see if the work has already been done. PL/SQL code that you run is cached in the shared pool, so the next time you run it, Oracle doesn’t have to read it in from disk again. PL/SQL code is not only cached here, it is shared here as well. If you have 1,000 sessions all executing the same code, only one copy of the code is loaded and shared among all sessions. Oracle stores the system parameters in the shared pool. The **data dictionary cache** (cached information about database objects) is stored here.**Dictionary cache** is a collection of database tables and views containing information about the database, its structures, privileges and users. When statements are issued oracle will check permissions, access, etc and will obtain this information from its dictionary cache, if the information is not in the cache then it has to be read in from the disk and placed in to the cache. The more information held in the cache the less oracle has to access the slow disks.The parameter SHARED_POOL_SIZE is used to determine the size of the shared pool, there is no way to adjust the caches independently, you can only adjust the shared pool size.The shared pool uses a LRU (least recently used) list to maintain what is held in the buffer, see buffer cache for more details on the LRU.

4.Large Pool: The large pool is not so named because it is a “large” structure (although it may very well be large in size). It is so named because it is used for allocations of large pieces of memory that are bigger than the shared pool is designed to handle. Large memory allocations tend to get a chunk of memory, use it, and then be done with it. There was no need to cache this memory as in buffer cache and Shared Pool, hence a new pool was allocated. So basically Shared pool is more like **Keep Pool** whereas Large Pool is similar to the **Recycle Pool**. Large pool is used specifically by:

• Shared server connections, to allocate the UGA region in the SGA.
• Parallel execution of statements, to allow for the allocation of interprocess message buffers, which are used to coordinate the parallel query servers.
• Backup for RMAN disk I/O buffers in some cases.

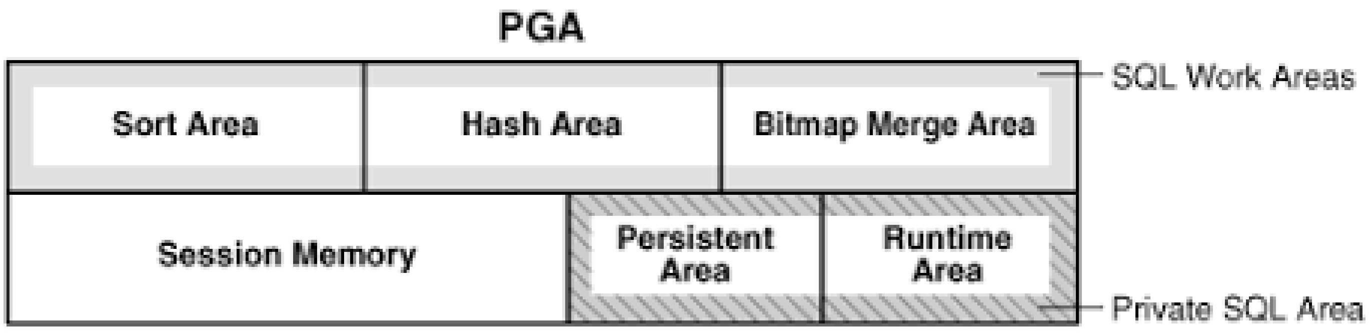
5.Java Pool: The Java pool is used in different ways, depending on the mode in which the Oracle server is running. In dedicated server mode the total memory required for the Java pool is quite modest and can be determined based on the number of Java classes you'll be using. In shared server connection the java pool includes shared part of each java class and Some of the UGA used for per-session state of each session, which is allocated from the JAVA_POOL within the SGA.

6.Streams Pool: The Streams pool (or up to 10 percent of the shared pool if no Streams pool is configured) is used to buffer queue messages used by the Streams process as it moves or copies data from one database to another.

The SGA comprises a number of memory components, which are pools of memory used to satisfy a particular class of memory allocation requests. Examples of memory components include the shared pool (used to allocate memory for SQL and PL/SQL execution), the java pool (used for java objects and other java execution memory), and the buffer cache (used for caching disk blocks). All SGA components allocate and deallocate space in units of granules. Oracle Database tracks SGA memory use in internal numbers of granules for each SGA component.Granule size is determined by total SGA size. On most platforms, the size of a granule is 4 MB if the total SGA size is less than 1 GB, and granule size is 16MB for larger SGAs. Some platform dependencies arise. For example, on 32-bit Windows, the granule size is 8 M for SGAs larger than 1 GB.Oracle Database can set limits on how much virtual memory the database uses for the SGA. It can start instances with minimal memory and allow the instance to use more memory by expanding the memory allocated for SGA components, up to a maximum determined by the SGA_MAX_SIZEinitialization parameter. If the value for SGA_MAX_SIZE in the initialization parameter file or server parameter file (SPFILE) is less than the sum the memory allocated for all components, either explicitly in the parameter file or by default, at the time the instance is initialized, then the database ignores the setting for SGA_MAX_SIZE.

2)PGA:

PGA is the memory reserved for each user process connecting to an Oracle Database and is allocated when a process is created and deallocated when a process is terminated.



Contents of PGA:-

- *Private SQL Area*: Contains data such as bind information and run-time memory structures. It contains *Persistent Area* which contains bind information and is freed only when the cursor is closed and *Run time Area* which is created as the first step of an execute request. This area is freed only when the statement has been executed. The number of Private SQL areas that can be allocated to a user process depends on the OPEN_CURSORS initialization parameter.
- *Session Memory:* Consists of memory allocated to hold a session's variable and other info related to the session.
- *SQL Work Areas*: Used for memory intensive operations such as: Sort, Hash-join, Bitmap merge, Bitmap Create.

Automatic PGA Memory Management

Before Auto-Memory management DBA had to allocate memory to:-

- SORT_AREA_SIZE: The total amount of RAM that will be used to sort information before swapping out to disk.
- SORT_AREA_RETAINED_SIZE: The amount of memory that will be used to hold sorted data after the sort is complete.
- HASH_AREA_SIZE: The amount of memory your server process can use to store hash tables in memory. These structures are used during a hash join, typically when joining a large set with another set. The smaller of the two sets would be hashed into memory and anything that didn’t fit in the hash area region of memory would be stored in the temporary tablespace by the join key.

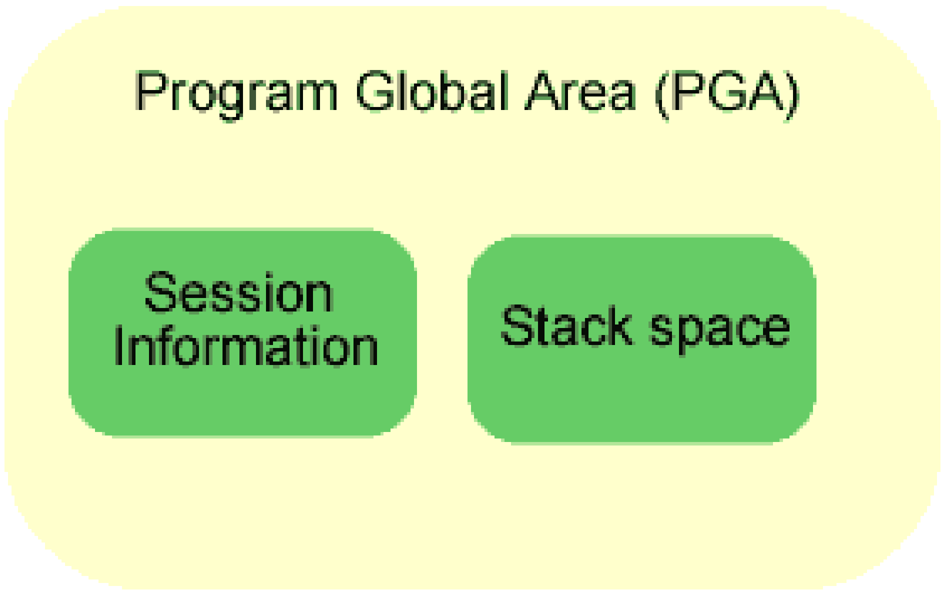
To enable PGA Auto-Mem Management enable the parameter WORKAREA_SIZE_POLICY and allocate total memory to be used for this purpose to PGA_AGGREGATE_TARGET.

NOTE:- *From 11gR1 You can set MEMORY_TARGET and auto-mem management for both SGA and PGA is taken care.*

I came across several DBAs enquiring about how the PGA Memory is allocated and from their I cam to know about several misconceptions people are having so writing a short note on the same.

The PGA_AGGREGATE_TARGET is a goal of an upper limit. It is not a value that is preallocated when the database is started up. You can observe this by setting the PGA_AGGREGATE_TARGET to a value much higher than the amount of physical memory you have available on your server. You will not see any large allocation of memory as a result. A serial (nonparallel query) session will use a small percentage of the PGA_AGGREGATE_TARGET, typically about 5 percent or less. Hence its not that all of the memory allocated to PGA is granted at the time DB is started and gradually increases with number of user processes. The algorithm that I am aware of, allocates 5%

of PGA to the user process until there is crunch on the PGA and then modifies the allocation based on the usage requirement of the user process.



Staring with Oracle 9i there is a new to manage the above settings that is to let oracle manage the PGA area automatically by setting the parameter following parameters Oracle will automatically adjust the PGA area basic on users demand.

- **workarea_size_policy** – you can set this option to manual or auto (default)
- **pga_aggregate_target** – controls how much to allocate the PGA in total

Oracle will try and keep the PGA under the target value, but if you exceed this value Oracle will perform multi-pass operations (disk operations).

Memory Area	**Dedicated Server**	**Shared Server**
Nature of Session Memory	Private	Shared
Location of Persistent Area	PGA	SGA

Location of the part of the runtime area for SELECT statements	PGA	PGA
Location for the run time area for DDL/DML statements	PGA	PGA

3)UGA:

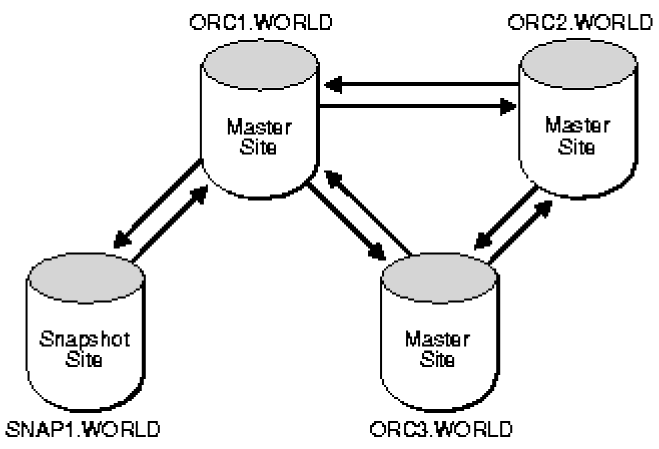
The UGA (User Global Area) is your state information, this area of memory will be accessed by your current session, depending on the connection type (shared server) the UGA can be located in the SGA which is accessible by any one of the shared server processes, because a dedicated connection does not use shared servers the memory will be located in the PGA

- Shared server – UGA will be part of the SGA
- Dedicated server – UGA will be the PGA

Oracle Notes Part-3

A **Snapshot** is a recent copy of a table from db or in some cases, a subset of rows/cols of a table. They are used to dynamically replicate the data between distributed databases.

Snapshot connected to a Single Master Site:



Snapshots can also contain a WHERE clause so that snapshot sites can contain customized data sets. Such snapshots can be helpful for regional offices or sales forces that do not require the complete corporate data set.When a snapshot is refreshed, Oracle must examine all of the changes to the master table to see if any apply to the snapshot. Therefore, if any changes where made to the master table since the last refresh, a snapshot refresh will take some time, even if the refresh does not apply any changes to the snapshot. If, however, no changes at all were made to the master table since the last refresh of a snapshot, the snapshot refresh should be very quick.
Snapshot and materialized view are almost same same but with one difference.
You can say that materialized view =snapshot + query rewrite functionality query rewrite functionality:In materialized view you can enable or disable query rewrite option. which means database server will rewrite the query so as to give high performance. Query rewrite is based on some rewritten standards(by oracle itself).So the database server will follow these standards and rewrite the query written in the materialized view ,but this functionality is not there in snapshots.
Simple snapshots are the only type that can use the FAST REFRESH method. A snapshot is considered *simple* if the defining query meets the following criteria:

- It does not contain any DISTINCT or aggregation functions.
- It does not contain a GROUP BY or CONNECT BY clause.

- It does not perform set operations (UNION, UNION ALL, INTERSECT, etc.).
- It does not perform joins other than those used for subquery subsetting.
- Essentially, a simple snapshot is one that selects from a single table and that may or may not use a WHERE clause.

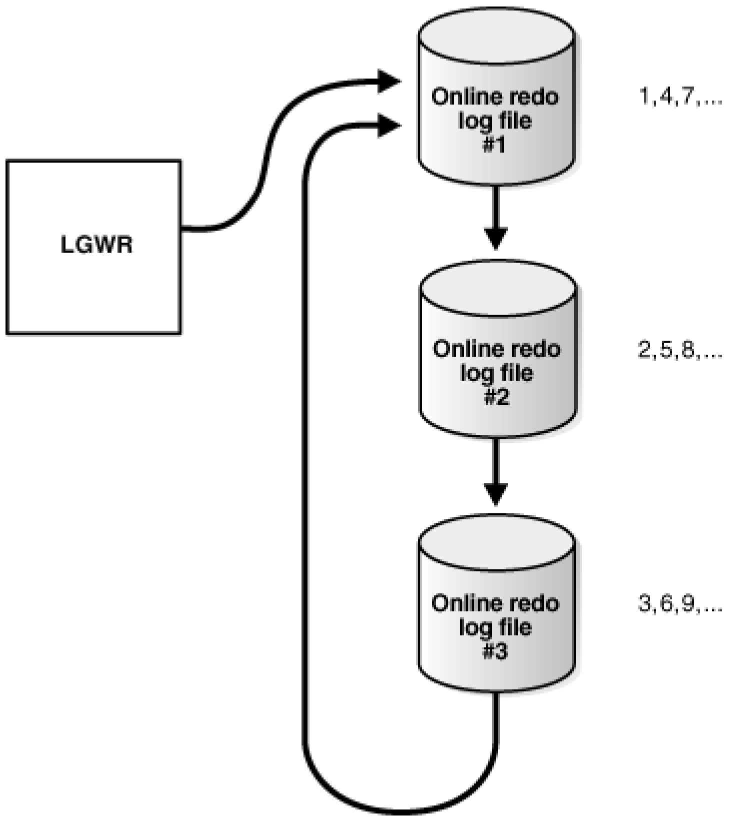
Oracle8 extends the universe of simple snapshots with a feature known as subquery subsetting, described in the later section entitled "Subquery Subsetting."
Not surprisingly, any snapshot that is not a simple snapshot is a *complex* snapshot. Complex snapshots can only use COMPLETE refreshes, which are not always practical. For tables of more than about 100,000 rows, COMPLETE refreshes can be quite unwieldy.You can often avoid this situation by creating simple snapshots of individual tables at the master site and performing the offending query against the local snapshots.

Oracle Notes Part-2

Redo: In the Oracle RDBMS environment, **redo logs** comprise files in a proprietary format which log a history of all changes made to the database. Each redo log file consists of redo records. A redo record, also called a redo entry, holds a group of change vectors, each of which describes or represents a change made to a single block in the database.
For example, if a user UPDATEs a salary-value in a table containing employee-related data, the DBMS generates a redo record containing change-vectors that describe changes to the data segment block for the table. And if the user then COMMIT the update, Oracle generates another redo record and assigns the change a "system change number" (SCN).
LGWR writes to redo log files in a circular fashion. When the current redo log file fills, LGWR begins writing to the next available redo log file. When the last available redo log file is filled, LGWR returns to the first redo log file and writes

to it, starting the cycle again. The numbers next to each line indicate the sequence in which LGWR writes to each redo log file.

Reuse of Redo Log Files by LGWR:



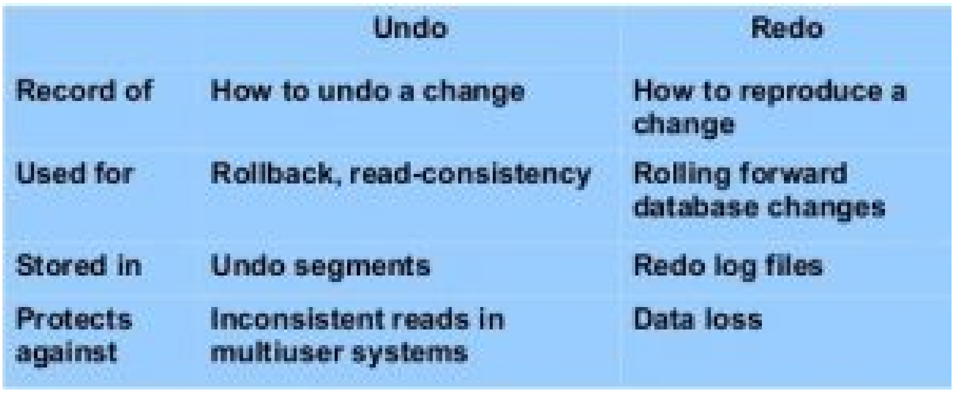
Oracle Database uses only one redo log files at a time to store redo records written from the redo log buffer. The redo log file that LGWR is actively writing to is called the current redo log file.Redo log files that are required for instance recovery are called active redo log files. Redo log files that are no longer required for instance recovery are called inactive redo log files.

A log switch is the point at which the database stops writing to one redo log file and begins writing to another. Normally, a log switch occurs when the current redo log file is completely filled and writing must continue to the next redo log file. However, you can configure log switches to occur at regular intervals, regardless of whether the current redo log file is completely filled. You can also force log switches manually.

Oracle Database assigns each redo log file a new log sequence number every time a log switch occurs and LGWR begins writing to it. When the database archives redo log files, the archived log retains its log sequence number. A redo log file that is

cycled back for use is given the next available log sequence number.
UNDO: Oracle Database creates and manages information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo.
Undo records are used to:

- Roll back transactions when a ROLLBACK statement is issued
- Recover the database
- Provide read consistency
- Analyze data as of an earlier point in time by using Oracle Flashback Query
- Recover from logical corruptions using Oracle Flashback features.



	Undo	Redo
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, read-consistency	Rolling forward database changes
Stored in	Undo segments	Redo log files
Protects against	Inconsistent reads in multiuser systems	Data loss

---

Oracle Notes Part-1

An **Oracle database** is a collection of data treated as a unit. The purpose of a **database** is to store and retrieve related information. A **database** server is the key to solving the problems of information management.

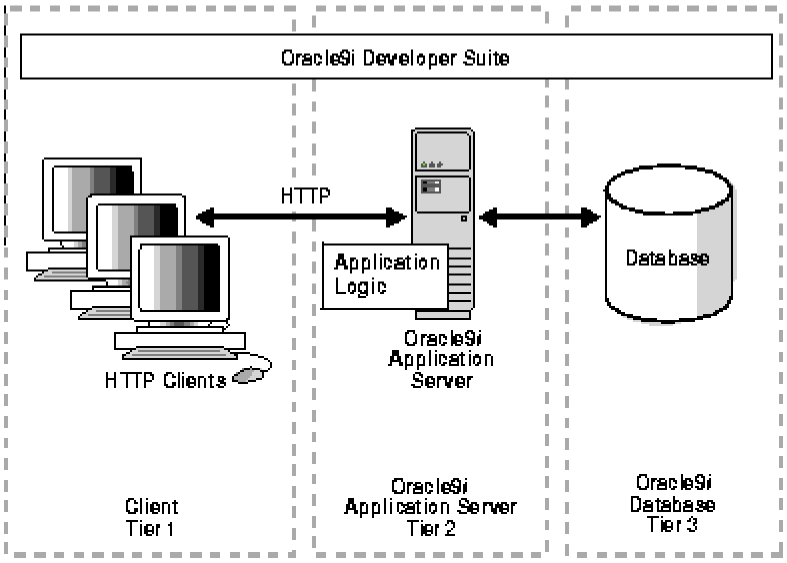
- Oracle 9i is an Object/Relational Database Management

System specifically designed for **e-commerce.**

- Oracle 9i, a version of Oracle database. The letter “i” refers to the **internet.**
- It can scale ten thousands of concurrent users.
- It includes Oracle 9i Application server and Oracle 9i Database that provide a comprehensive high-performance infrastructure for Internet Applications.
- It supports client-server and web based applications.
- The maximum Database holding capacity of Oracle 9i is upto **512** peta bytes(PB).[1 Peta Byte = 1000 Tera Byte]
- It offers Data warehousing features and also many management features.

We can set primary key on table up to 16 columns of table in oracle 9i as well as in Oracle 10g.
The maximum number of data files in Oracle 9i and Oracle 10g Database is 65,536.

Oracle 9i Architecture:



Oracle Storage Structures:

An essential task of a relational database is data storage. This section briefly describes the physical and logical storage structures used by Oracle Database.

Physical Storage Structures

The physical database structures are the files that store the data. When you execute the SQL command CREATE DATABASE, the following files are created:

- Data files
  Every Oracle database has one or more physical data files, which contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the data files.
- Control files
  Every Oracle database has a control file. A control file contains metadata specifying the physical structure of the database, including the database name and the names and locations of the database files.
- Online redo log files
  Every Oracle Database has an online redo log, which is a set of two or more online redo log files. An online redo log is made up of redo entries (also called redo records), which record all changes made to data.

Logical Storage Structures

This section discusses logical storage structures. The following logical storage structures enable Oracle Database to have fine-grained control of disk space use:

- Data blocks
  At the finest level of granularity, Oracle Database data is stored in data blocks. One data block corresponds to a specific number of bytes on disk.
- Extents
  An extent is a specific number of logically contiguous data blocks, obtained in a single allocation, used to store a specific type of information.
- Segments
  A segment is a set of extents allocated for a user object (for example, a table or index), undo data, or

temporary data.

- Tablespaces
  A database is divided into logical storage units called tablespaces. A tablespace is the logical container for a segment. Each tablespace contains at least one data file.