TEMA 4
Administración de las estructuras de Memoria

4.1. ARQUITECTURA BÁSICA DE UN SERVIDOR DE BASE DE DATOS.

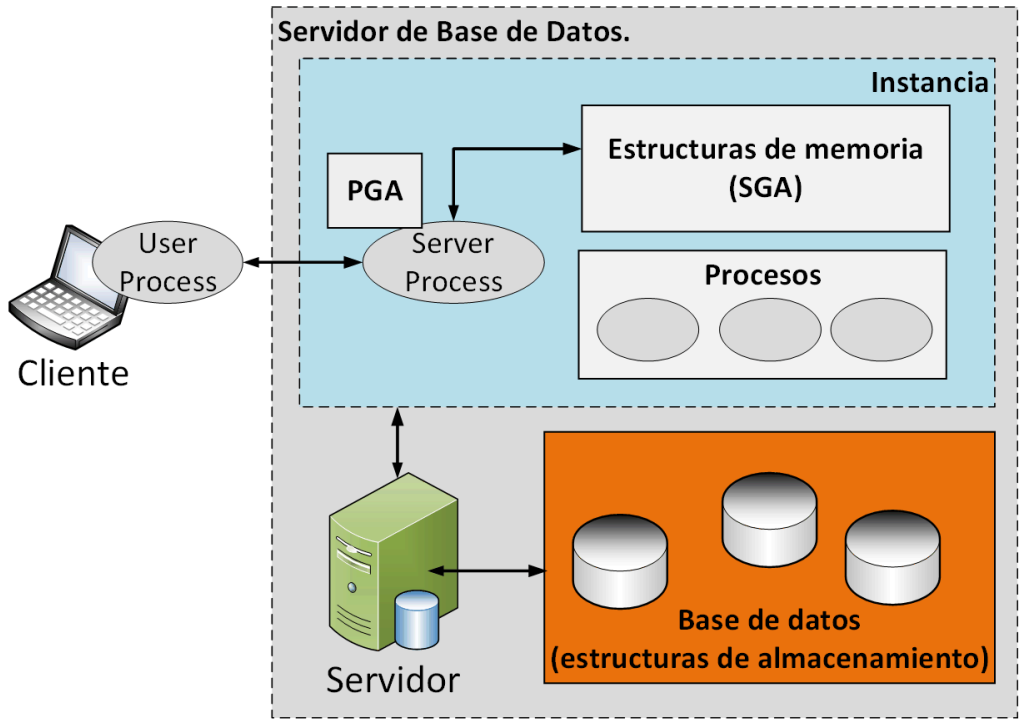
- A nivel general, un servidor de bases de datos administra de forma segura grandes cantidades de datos en ambientes donde múltiples usuarios pueden acceder de forma concurrente.
- El servidor también previene accesos no autorizados.
- Proporciona soluciones para atender fallas a través del concepto de "database recovery".

Instancia y base de datos

Un servidor de base de datos está formado por una base de datos y al menos una ***instancia de base de datos***. Debido a que estos 2 conceptos están fuertemente ligados, suele emplearse el término "***Base de datos Oracle***" para hacer referencia a ambos.

- Base de datos: Conjunto de archivos localizados en disco que contienen datos. Estos archivos pueden existir independiente a la existencia de la instancia.
- Instancia: Conjunto de estructuras de memoria ***(SGA)*** que administran a los archivos que contienen datos. Formada a su vez por un conjunto de sub-áreas de memoria y por un conjunto de ***procesos de background***.

La siguiente figura muestra a nivel básico la arquitectura de un servidor de bases de datos Oracle.



Estructuras principales

- Áreas de memoria
- Procesos de background
- Estructura lógica de los datos
- Estructuras de almacenamiento (estructuras físicas)
    - Estas estructuras son totalmente *independientes*. Si se altera la estructura física, la parte lógica no se ve afectada.

Variantes de arquitecturas

- Oracle RAC:
    - Múltiples instancias
    - Una sola base de datos
- Oracle Multitenant Architecture
    - Una Instancia
    - Múltiples Bases de datos.
- Las variantes anteriores se pueden combinar.

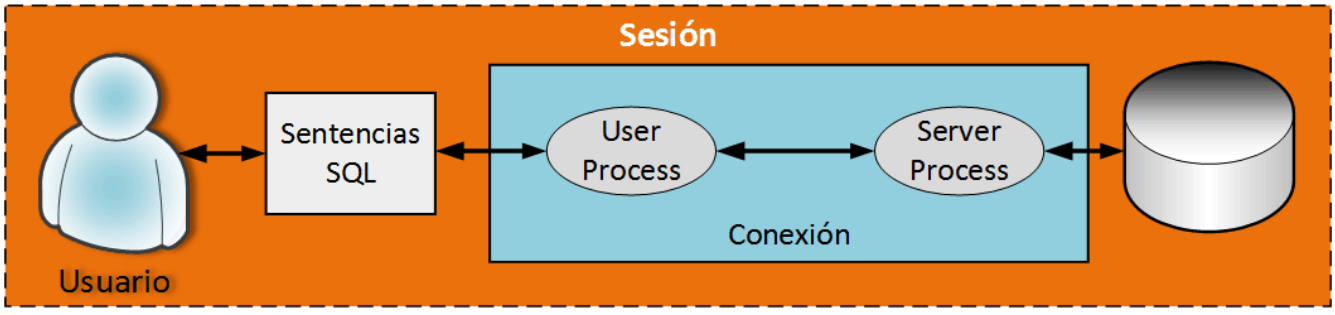
¿Qué sucede cuando se inicia una instancia?

- Se crea un área de memoria compartida llamada SGA (System Global Area)
- Procesos de background son ejecutados. Llamados también "*Hilos de Control*".

Conexión, sesión, user y server process

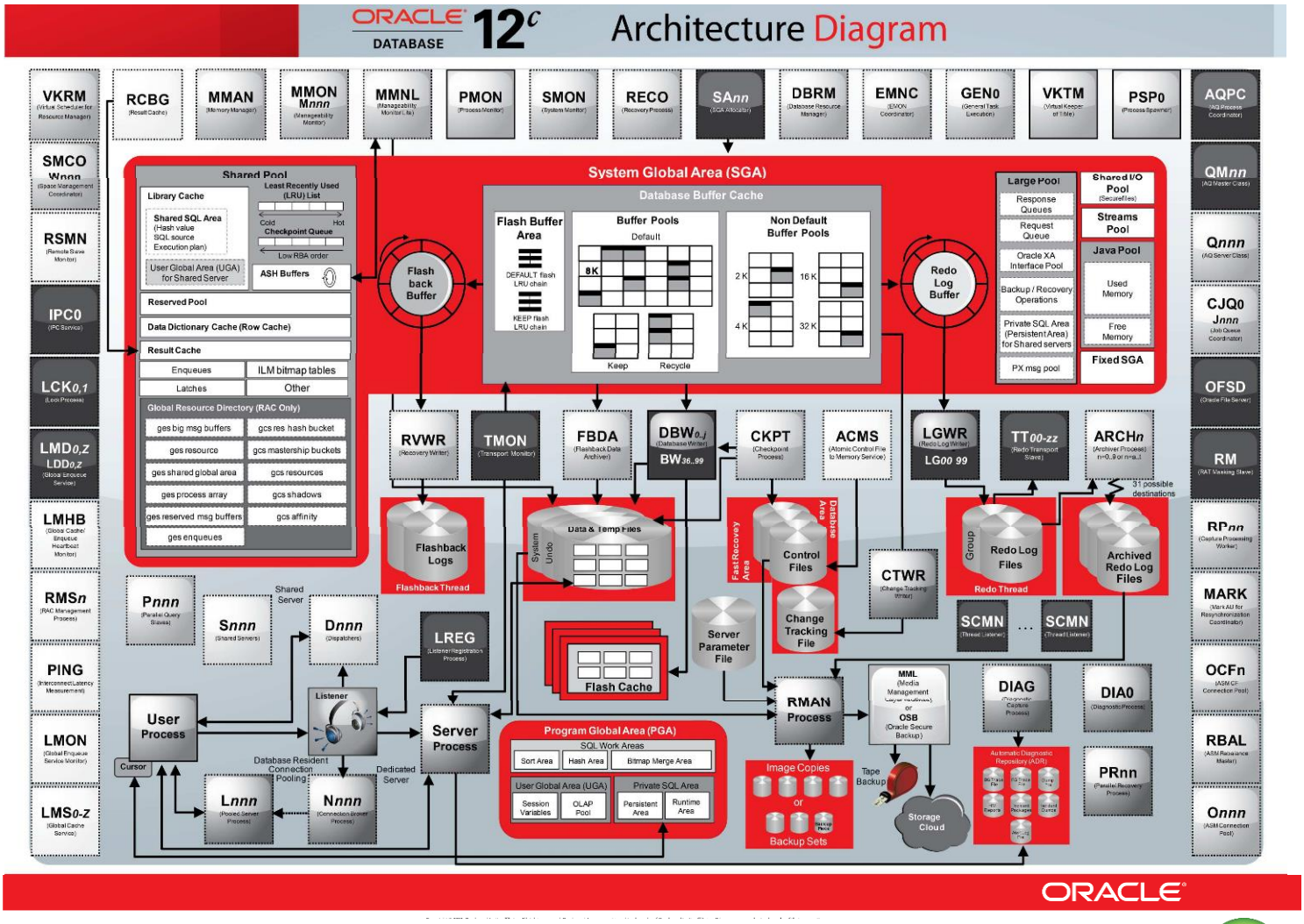
- Por cada conexión realizada hacia la instancia se crea un ***user process.*** Este proceso se ejecuta del lado de la aplicación (donde se encuentra el cliente).
- Cada ***user process*** tiene asociado a un ***server process*** que se encuentra del lado de la instancia.
- A cada ***server process*** se le asocia su propia área de memoria privada (sesión) llamada ***PGA.***

- *Conexión*: Comunicación entre un *user process* y un *server process*.
- *Sesión*: Conexión específica de un usuario en la instancia a través el uso de un *user process.*
    - Representa el estado actual de un usuario conectado a la BD.



4.2. Arquitectura completa de una base de datos Oracle.

La siguiente imagen muestra la arquitectura de una base de datos Oracle 12c similar a la anterior, pero a detalle (real). La imagen fue tomada de este enlace el cual representa una guía interactiva rápida para un DBA. Se recomienda ampliamente revisarla.



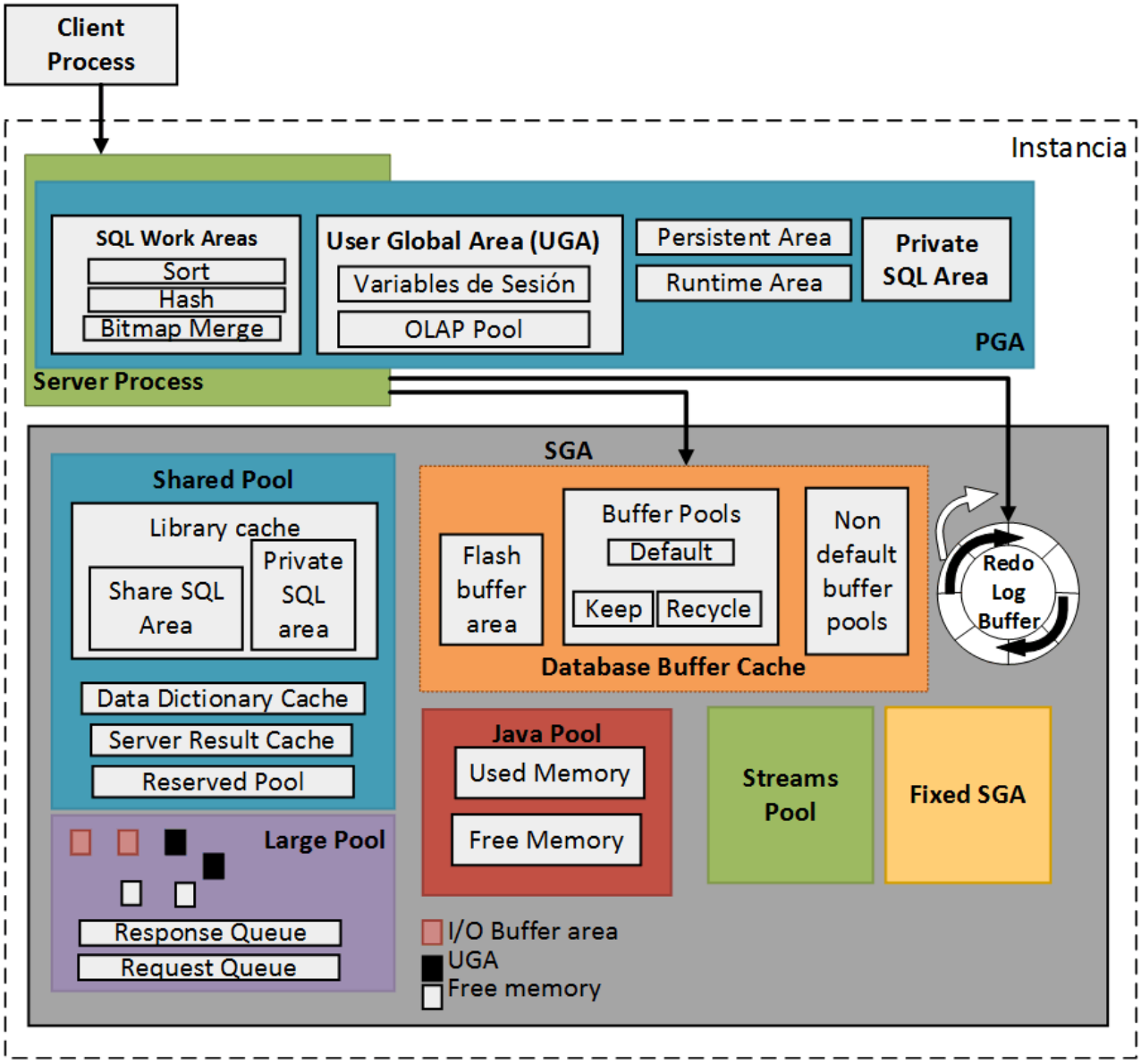
4.3. ARQUITECTURA DE LAS ESTRUCTURAS DE MEMORIA.

Al momento de iniciar una instancia, se crean diversas áreas de memoria, así como diversos procesos de background.

En las áreas de memoria se almacena información de diversos tipos:

- Código de programas
- Información acerca de cada sesión sin importar si está activa o inactiva.
- Información que muestra el status actual de la ejecución de un programa o de una sentencia SQL
- Información de los bloqueos existentes en la base de datos
- Datos en cache: bloques de datos, datos REDO.

Las principales áreas de memoria se especifican en el siguiente diagrama:



- *System Global Area* (SGA).
- *Program Global Area (PGA)*.
- *User Global Area (UGA).*

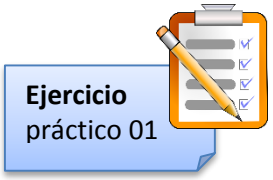
4.3.1. System Global Area (SGA)

- Conjunto de áreas de memoria compartidas que contienen datos e información de control empleada para el correcto funcionamiento de la instancia.
  - Es compartida por todos los procesos de background y por todos los *servers processes.*
  - Principales estructuras que conforman a la SGA:
    - Shared pool
    - Database buffer cache
    - Redo Log Buffer
    - Large Pool
    - Java Pool
    - Streams Pool
    - Fixed SGA

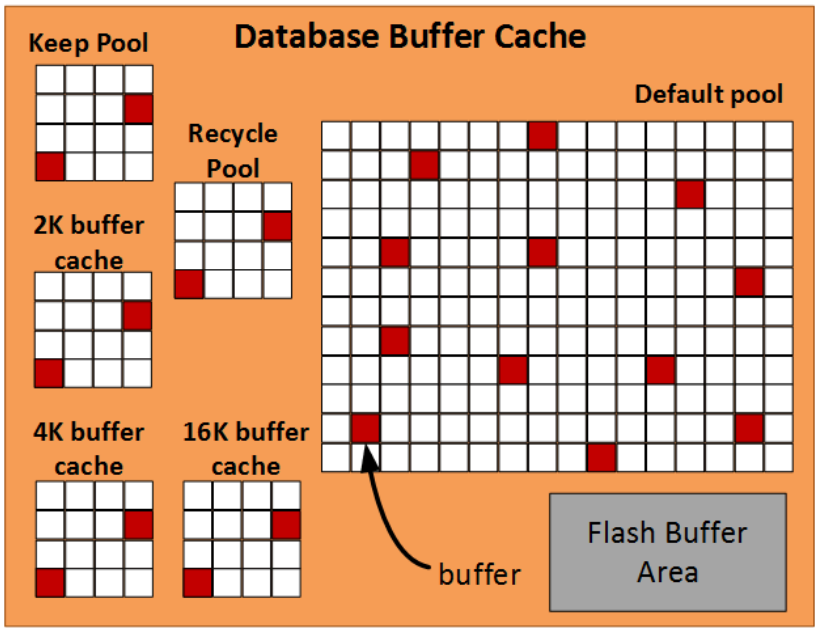
- Todos los *servers processes* leen y escriben datos en la SGA requeridos por sus respectivos usuarios.
- El tamaño de la SGA se puede apreciar al momento de iniciar la instancia:

```
sql> startup
Total System Global Area  368283648 bytes
Fixed Size                  1300440 bytes
Variable Size             343935016 bytes
Database Buffers           16777216 bytes
Redo Buffers                6270976 bytes
Database mounted.
Database opened.
```

Revisar el documento correspondiente al *ejercicio práctico 01*. En este ejercicio se realizan consultas básicas hacia las vistas del diccionario de datos que muestran información y el uso de memoria de las áreas que integran a la SGA.



4.3.1.1. Database buffer cache.



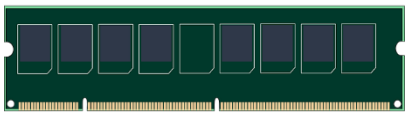
Características generales.

- Llamado también *buffer cache*
- Guarda copias de bloques de datos que son leídos de los data files.
- Buffer: Dirección de memoria en la que se almacena temporalmente un bloque de datos (caché) recientemente empleado por uno o varios usuarios.
- Empleado para optimizar operaciones I/O.
  - Los cambios a los datos se realizan en este caché (actualiza bloques de datos).
  - Almacena metadatos que describen a los datos modificados en el *Redo Log Buffer.*
  - Al aplicar una operación `commit`, el contenido del *Redo Log Buffer* es escrito al archivo llamado *Online Redo Log.*
  - Notar que esta operación no implica escribir en los data files. La sincronización de los *Online Redo Logs* con los *Data files* se realiza de forma *perezosa* a través de un proceso de background llamado *DBW (Database Writer).*

- Empleado para mantener en memoria bloques de datos frecuentemente usados. Los bloques usados con baja frecuencia son escritos en disco para liberar memoria.

Estados de un buffer.

  - *Unused:* El buffer nunca se ha sido empleado, por lo que está totalmente disponible para ser empleado.
  - *Clean:* El buffer ya fue empleado, contiene datos modificados, todos los datos que contiene han sido sincronizados en disco.
  - *Dirty:* El buffer contiene bloques de datos modificados que no han sido sincronizados en disco. Para poder reutilizar dichos bloques, se debe realizar una sincronización con los data files.



Modos de acceso.

Cuando un usuario solicita un dato, la instancia lo recupera del db buffer cache y puede utilizar alguno de los siguientes modos:

- *Current mode:* Obtiene el dato directamente del buffer cache sin importar si este ha sido modificado o no. Útil para transacciones que están actualizando datos del mismo buffer dentro de la misma transacción.
- *Consistent mode:* Se llama consistente debido a que el dato que se desea leer no debe contener cambios realizados por otras transacciones que aún no han hecho `commit` o `rollback`. La BD genera una especie de "clone" del buffer para poder ofrecer lecturas consistentes. Llamado también "*consistent read clone".*

Buffer pools.

El db buffer cache se divide a su vez en varios buffer pools encargados de administrar los bloques de datos leídos.

- *Default pool*: Como su nombre lo indica, representa el lugar por default donde se guardan los bloques de datos leídos de los data files a menos de que se haya especificado una configuración diferente.
- *Keep pool*: Contiene bloques de datos que son accedidos frecuentemente, pero que por falta de espacio han sido desplazados o eliminados del *default pool.*
- *Recycle pool:* Contiene bloques de datos que han sido empleados de forma infrecuente.
- *nK pool:* Donde N es el tamaño del bloque de datos. Empleados para guardar bloques de datos de tablespaces que fueron creados con un tamaño de bloque diferente al default. El tamaño default del bloque de datos es 8K, por lo que pueden existir pools para almacenar bloques de 2K, 4K, 16K.
- *Smart Flash Cache:* En caso de estar habilitada, parte del contenido del *database buffer caché* es escrito en esta área. Esta funcionalidad mejora el desempeño de la BD ya que resulta más eficiente escribir y leer a partir de una memoria Flash que de un disco magnético.
- Los parámetros `db_flash_cache_file` y `db_flash_cache_size` son empleados para configurar el *flash caché.*

Buffer I/O vs Physical I/O

- Operaciones *Buffer I/O* conocidas también como *Logical I/O,* se refiere a operaciones de lectura y escritura que se realizan sobre los buffers leídos del db cache.

- Operaciones *Physical I/O* Ocurren cuando se lee un bloque de datos ya sea del disco magnético o de la memoria flash para ser cargado en memoria.

Reemplazo de buffers en el cache.

Para contar con un desempeño óptimo del db buffer cache, la instancia debe decidir de forma eficiente en qué momento se debe realizar el reemplazo de buffers. Existen 2 algoritmos:

- *LRU Based, block level replacement algorithm*. Emplea el algoritmo Last Recently Used (LRU) que contiene punteros a los buffers sucios y limpios. Los buffers menos usados son candidatos a ser reemplazados. Representa el algoritmo por default.
- *Temperature-based, object-level replacement algorithm.* Este algoritmo se emplea en situaciones donde se requiere cargar al cache grandes cantidades de bloques, por ejemplo, un full table access de una tabla. Si su contenido no cabe en memoria, solo se carga cierto porcentaje de bloques. Por ejemplo, la BD puede decidir cargar el 95% de los bloques y dejar el 5% restante en disco. El 95% corresponde a los bloques "más populares" (hotter tables) o más empleados y el 5% corresponde con los bloques menos populares "(cooler tables).
    - En algunos casos, la tabla puede cargarse directamente a la memoria PGA pasando por alto el db buffer cache: *Direct path read.*
- El parámetro `db_big_table_cache_percent_target` permite establecer el porcentaje del db buffer cache que puede hacer uso de este algoritmo.

Proceso de escritura de buffers.

- Como se mencionó anteriormente el proceso de background DBW es el encargado de escribir buffers sucios a disco (data files).
- Este proceso realiza las escrituras básicamente al ocurrir los siguientes eventos:
    - El server process no puede encontrar buffers limpios para ser reutilizados con datos nuevos.
    - Se ha alcanzado el umbral para decidir realizar escrituras a disco. Es decir, el buffer cache contiene un número alto de buffers sucios.
    - Otros eventos se revisarán en el siguiente capítulo.

Proceso de lectura de buffers:

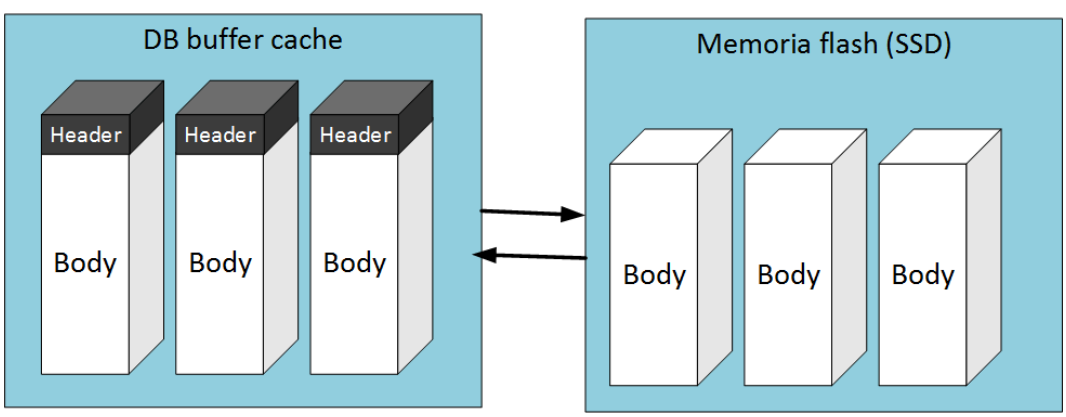
Suponer que un usuario conectado a la BD realiza una consulta a la base de datos. Internamente ocurre la siguiente secuencia de pasos:

1. El server process que representa a la petición del usuario verifica si existe en el *buffer cache* un buffer que contenga la información solicitada.
2. Si se encuentra, se realiza la lectura del buffer *(Cache hit).*
3. Si el buffer no se encuentra (*cache miss*), se realiza la lectura del bloque de datos y se almacena en el cache dejándolo disponible para posibles lecturas futuras.

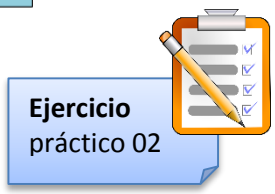
¿Qué sucede si un buffer limpio permanece en el cache por un largo periodo de tiempo sin uso?

El buffer sin uso puede reutilizarse. Si posteriormente se vuelve a requerir leer su contenido, el bloque de datos debe copiarse nuevamente al db buffer cache. Se emplea la siguiente estrategia para mejorar el desempeño la cual involucra a la memoria flash mencionada anteriormente:

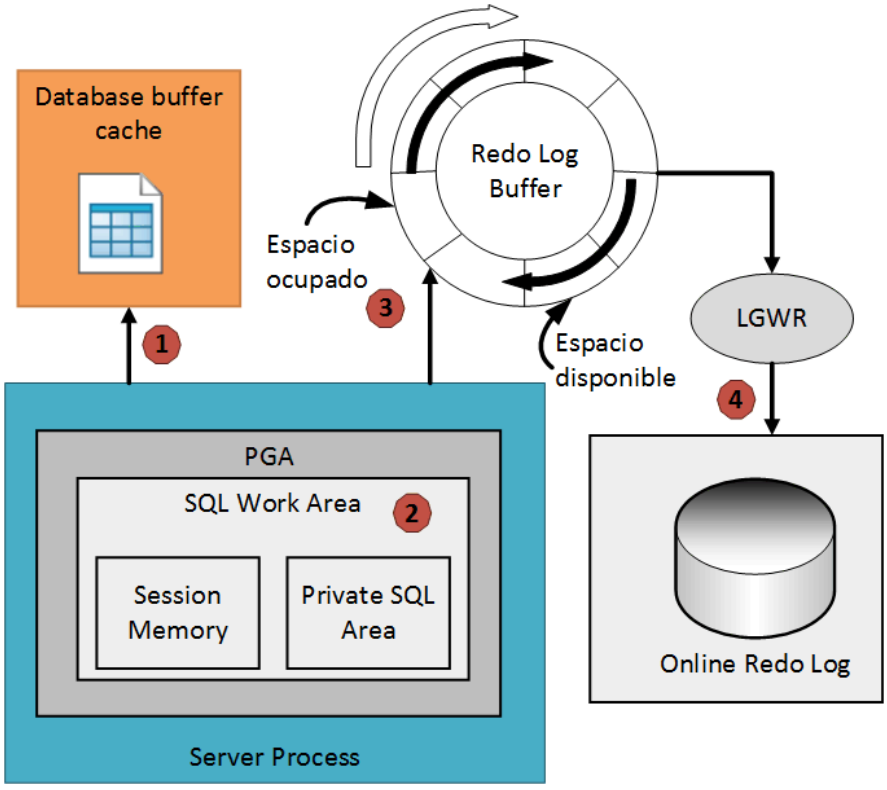
- *Uso de memoria flash deshabilitada*: Debido al algoritmo LRU, el buffer se reutiliza ya que no está siendo utilizado frecuentemente. Si posteriormente se requiere leer su contenido, el buffer se deberá leer nuevamente de disco.
- *Uso de memoria flash habilitada*: El DBW escribe el "cuerpo (body)" del bloque de datos de un buffer limpio a la memoria flash. Al ser respaldado en esta memoria, el buffer se puede reutilizar. En el db buffer cache se conserva el header del bloque. Si posteriormente el bloque se necesita, el bloque se lee de la memoria flash en lugar de leerlo de disco.



Revisar el documento correspondiente al ejercicio práctico 2. Se incluyen algunas configuraciones y consultas relacionadas con el DB buffer caché y la simulación de una memoria flash.



4.3.1.2. Redo Log Buffer



- Buffer circular ubicado en la SGA que almacena *REDO records* que describen los cambios realizados en la base de datos.

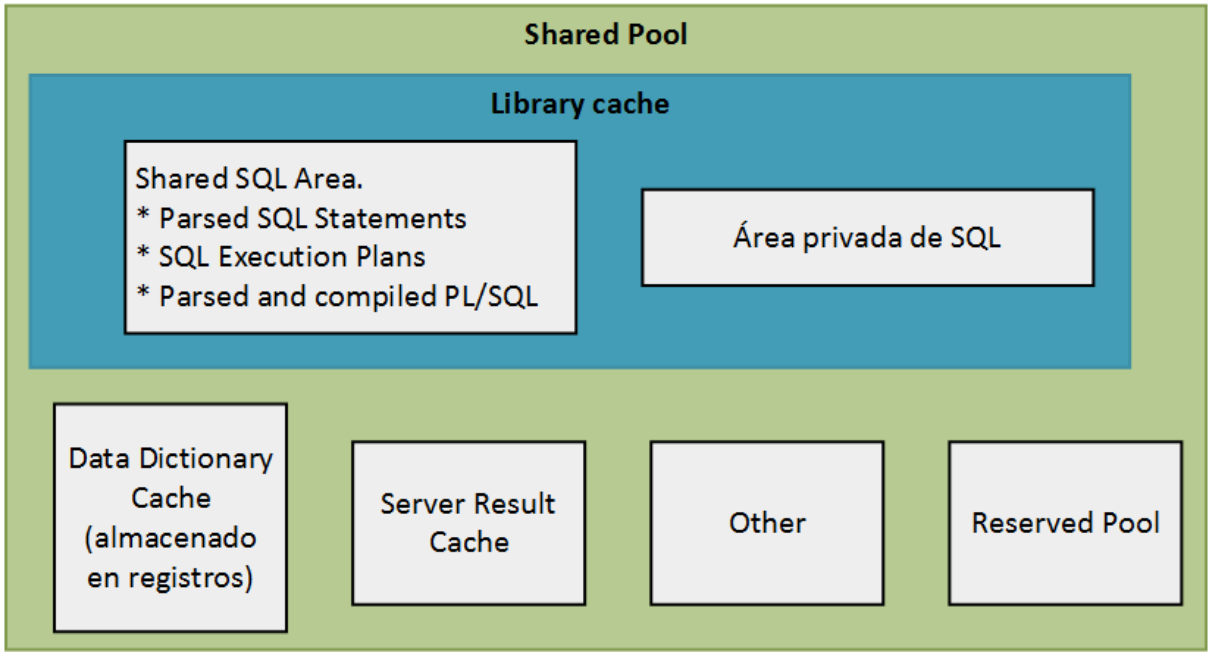
- *REDO Record.* Estructura de datos que contiene los datos necesarios para poder reconstruir los cambios realizados por sentencias DML o DDL. Estos datos son empleados entre otras cosas, para realizar la recuperación de datos (Database recovery).

A nivel general la forma en la que opera un Redo Log Buffer se describe en los siguientes pasos:

A. Los cambios que los usuarios realizan a los datos se graban o se guardan en el *buffer cache.*
B. Los *servers processes* toman los *REDO Records* que se encuentran en el espacio de memoria de los usuarios para ser guardados en el *Redo Log Buffer* dentro de la SGA (memoria).
C. Estos datos se almacenan en algún espacio disponible dentro del *Redo Log Buffer*. Se dice que es circular debido a que los espacios disponibles se reutilizan. En un punto del círculo se escriben y en otro se liberan. Al ser liberado, el espacio previamente ocupado se puede reutilizar.
D. El proceso de background llamado Log Writer Process (LGWR) es el encargado de liberar estos espacios ocupados. Para ello, toma el contenido del *Redo Log buffer* y lo escribe el al grupo activo de *Online Redo Logs* (archivos en disco). En capítulos posteriores se revisará a detalle las características y estructura de los *Online Redo Logs*.

- El parámetro `log_buffer` indica la cantidad de memoria empleada para almacenar datos REDO en el Online Redo Log.

4.3.1.3. Shared Pool



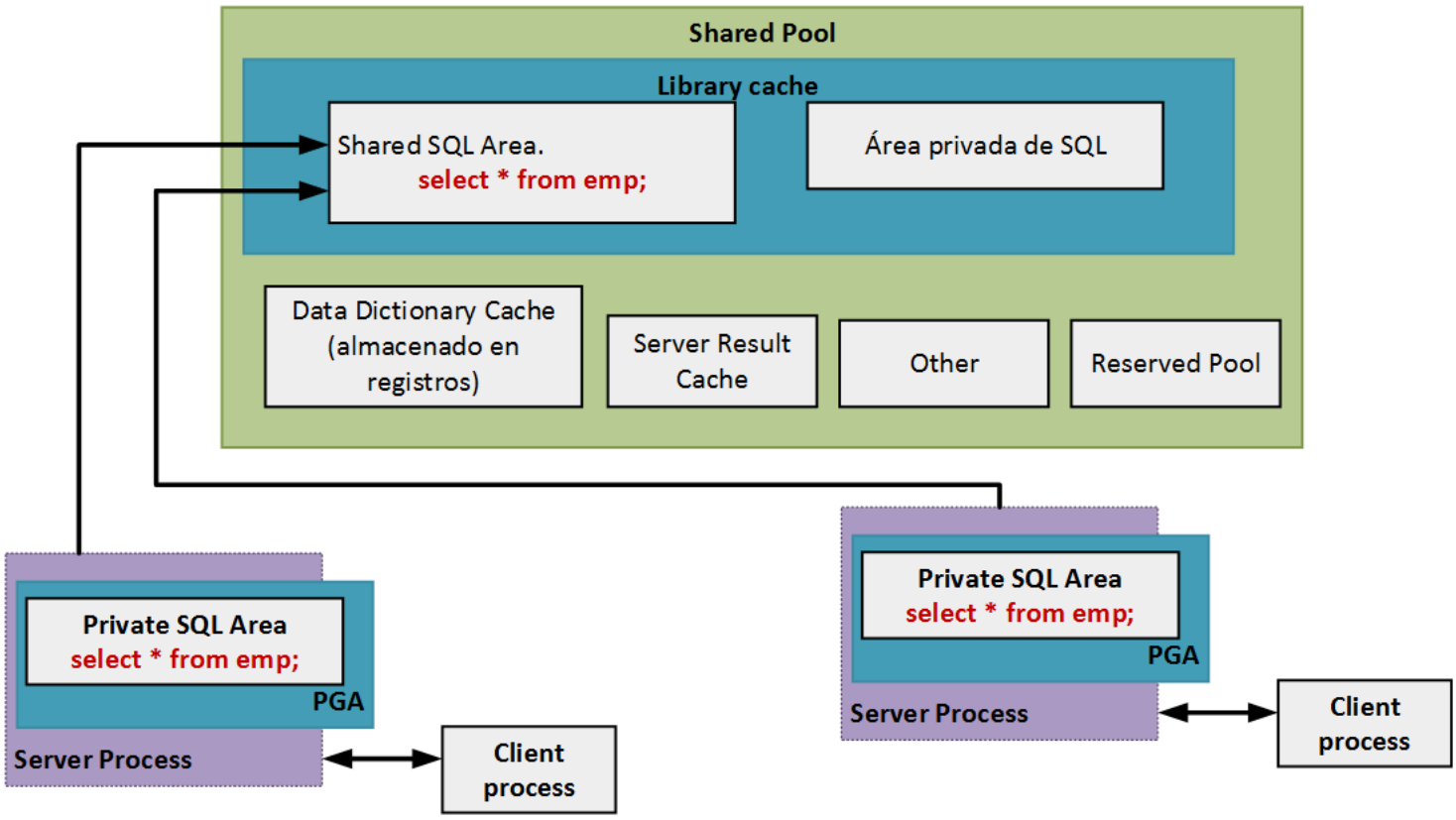
- En general esta área de memoria se emplea como caché de datos requeridos por diversos programas.
- Entre otras cosas almacena datos de código PL/SQL compilado, parámetros del sistema, información del diccionario de datos.
- El Shared Pool se emplea en prácticamente cualquier operación que se realiza en la base de datos.
- Shared Pool está integrado por los siguientes componentes:
  - Library Cache
  - Data Dictionary Cache
  - Server Result Cache
  - Reserved Pool

Library Cache:

- Se encarga de almacenar código ejecutable tanto SQL como PL/SQL
- Al momento de ejecutar una sentencia SQL, la BD intenta reutilizar código previamente ejecutado.
- Si este código es encontrado en este caché y si es compatible, se reutilizará. Acción conocida como *library cache hit (Soft parse).*
- Si el código no se encuentra, se crea una nueva entrada al caché. *Library cache miss (Hard parse).*

Áreas SQL privadas y compartidas.

- La BD hace uso de un área compartida SQL para procesar por primera vez una sentencia SQL. Esta área puede ser empleada por todos los usuarios y contiene:
  - La sentencia SQL parseada.
  - El plan de ejecución.
- Por otro lado, cada sesión de usuario define su área SQL privada ubicada en su PGA.
- Si varias sesiones ejecutan la misma sentencia SQL desde sus respectivas áreas privadas, ambas sesiones apuntarán a la misma área SQL compartida ubicada en la SGA.
- Este proceso se ilustra en la siguiente figura:



Data Dictionary Cache

- Como parte de la operación normal de la base de datos, por ejemplo, hace uso del DD mientras se realiza el parseo de sentencias SQL.
- Debido a la alta frecuencia con la que se accede al DD, se tiene un área de memoria designada para mejorar el desempeño llamada *Data Dictionary Cache.*

- A este buffer también se le conoce como *Row Cache* debido a que los datos del DD se guardan en forma de renglones en lugar de ser almacenados como bloques de datos (buffers).

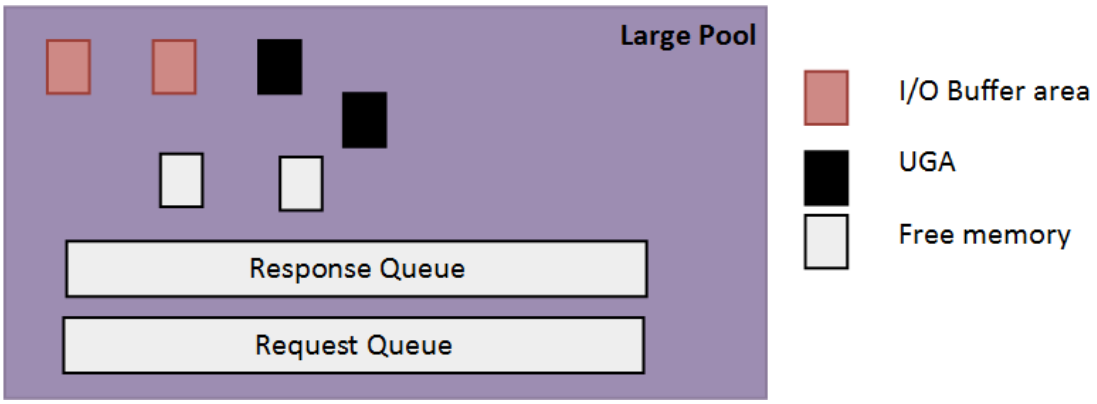
Server Result Cache

- Encargado de guardar en caché los resultados de sentencias SQL.
- Si una sesión ejecuta la misma sentencia `select` repetidas veces, los resultados se pueden obtener del caché en lugar de ejecutar la sentencia y realizar todo el proceso de recuperación de datos.
- La BD invalida de forma automática el caché de resultados si alguna transacción modifica o altera los datos o metadatos de los objetos que fueron empleados para construir el caché de resultados.
- Existen distintos tipos de *Server Result Cache*:
  - SQL query result cache
  - PL/SQL function result cache.

Reserved pool

- Área de memoria ubicada en el Shared pool empleada para asignar cantidades grandes y contiguas de memoria.
- La Base de datos otorga memoria del Shared pool en forma de fragmentos. Esta fragmentación permite cargar a memoria objetos grandes (más de 5Kb) cuyos fragmentos no necesariamente están contiguos. Este tipo de asignación permite cuidar las áreas de memoria contigua.

4.3.1.4. Large Pool

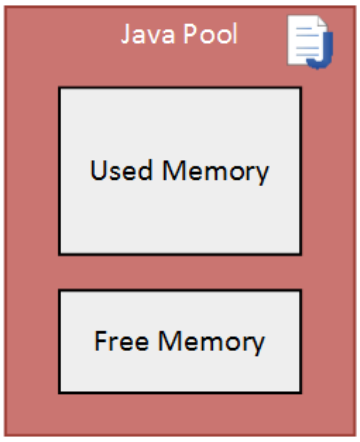


- Área de memoria opcional que es empleada para situaciones donde la memoria del *Shared Pool* no es suficiente o adecuada.
- Large pool puede ser empleado para ofrecer mayores cantidades de memoria en procesos como:
  - Oracle XA (funcionalidad empleada para administrar transacciones en bases de datos distribuidas).
  - Buffers empleados por RMAN
  - Buffers para mensajes en ejecución en paralelo.
  - Buffers de mensajes empleados en la ejecución de sentencias SQL en paralelo.

4.3.1.5. Java pool

Esta estructura de memoria proporciona un área de memoria para el código Java y datos específicos dentro de la Máquina Virtual de Java (JVM).

- Para las conexiones de un servidor dedicado, el Java Pool incluye la parte compartida de cada clase Java, incluyendo:
  - Métodos
  - Memoria de sólo lectura, etc.
- El parámetro `java_pool_size` determina el tamaño del pool.



4.3.1.6. Streams Pool

- Empleado exclusivamente para implementar Oracle Streams.
  - Oracle Streams es una funcionalidad que permite compartir información entre una o diferentes bases de datos a través de flujos de datos.
- El parámetro `streams_pool_size` muestra el tamaño del pool.

4.3.1.7. Fixed SGA

- Es un área de gestión interna. Su tamaño es determinado por la base de datos y no puede alterarse de forma manual y puede variar de versión a versión del manejador.

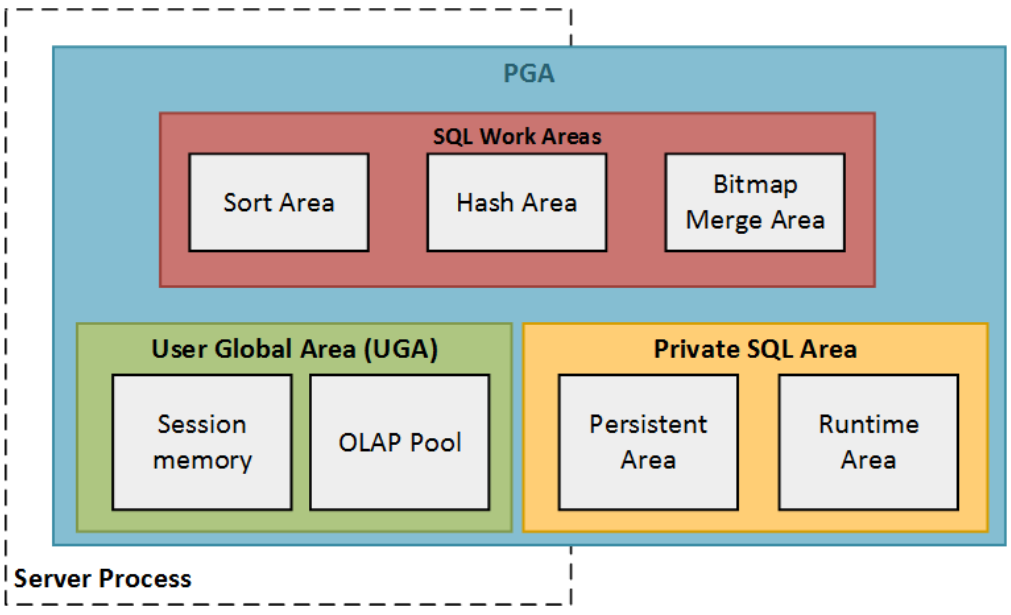
Puede contener, por ejemplo:

- Información general sobre el estado de la base de datos y la instancia a los cuales los procesos de background necesitan acceder.
- Información que es comunicada entre los procesos, como información de los bloqueos (locks).

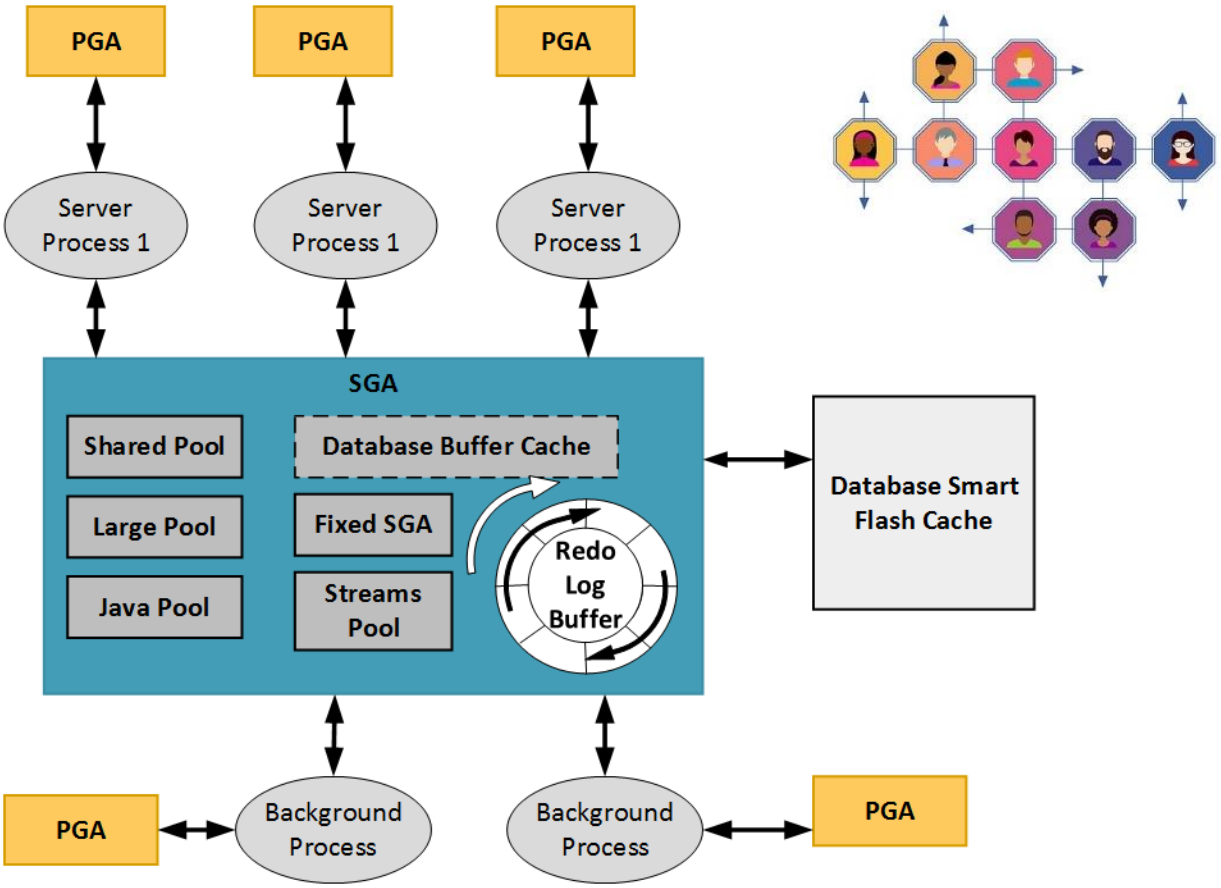
4.3.1.8. In-Memory Area

- Este pool opcional de la SGA fue implementado desde la versión de la base de datos 12.1.0.2.
- Almacena copias de tablas y particiones en un *formato columnar* especial que fue optimizado para escaneo rápido. A esta área de almacenamiento se le conoce como ***In-memory column store***.
- No reemplaza al buffer caché, pero actúa como un complemento.
- Ambas áreas de memoria pueden almacenar la misma información pero en diferentes formatos.

4.3.2. Program Global Area (PGA)



- Región de memoria que contiene datos e información de control de un solo *server process.*
  - No es compartida
  - Se crea al momento de crear un *server process.*
  - El total de la memoria PGA representa a la suma de todas las PGAs creadas por cada server process.
  - Notar que los procesos de background también pueden contar con su propia PGA.



4.3.2.1. Private SQL Area.

- Contiene información de sentencias SQL parseadas de cada sesión.
- Los server processes emplean esta área para almacenar variables, datos asociados con cursores.

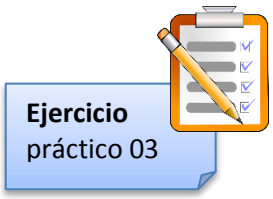
4.3.2.2. SQL work areas.

- Empleadas para realizar operaciones de memoria de forma intensiva.
  - Ordenamiento de datos.
  - Construcción de Hash tables para ejecutar joins con la técnica de *Hash Join*
  - Procesamiento de índices tipo Bitmap.

4.3.2.3. User Global Area (UGA)

- Memoria asociada con las sesiones de los usuarios.
  - Información de login
  - En general guarda el estado de cada sesión de usuario.

Revisar el documento correspondiente al ejercicio práctico 03 - Revisión de las áreas de memoria Redo Log buffer, shared pool, large pool, Java pool, In-memory column store, estructura de la PGA.



4.4. VISTAS DEL DICCIONARIO DE DATOS ASOCIADAS CON LAS ÁREAS DE MEMORIA.

4.4.1. V$SGA

Muestra un resumen de las áreas de memoria de la SGA:

```
select name,value, value/(1024*1024) as value_mb from v$sga;
```

NAME	VALUE	VALUE_MB
Fixed Size	8659768	8.25859832763671875
Variable Size	465567744	444
Database Buffers	54525952	52
Redo Buffers	8114176	7.73828125

Para comprobar el total de memoria de la SGA:

```
select sum(value)/(1024*1024) as total from v$sga;
```

TOTAL
511.99687957763671875

```
sys@jrcbd2> show parameter memory_target;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
memory_target                        big integer 512M
```

4.4.2. V$SGA_DYNAMIC_COMPONENTS

Muestra información de los componentes dinámicos de la SGA.

```
select component,current_size, current_size/(1024*1024) as current_size_mb,
min_size,max_size,oper_count,last_oper_type,
to_char(last_oper_time,'yyyy-mm-dd hh24:mi:ss') as last_oper_time
from v$sga_dynamic_components
order by current_size desc;
```

	COMPONENT	CURRENT_SIZE	CURRENT_SIZE_MB	MIN_SIZE	MAX_SIZE	OPER_COUNT	LAST_OPER_TYPE	LAST_OPER_TIME
1	shared pool	230686720	220	230686720	234881024	4	SHRINK	2019-09-08 22:45:27
2	DEFAULT buffer cache	46137344	44	41943040	46137344	4	GROW	2019-09-08 22:45:27
3	large pool	20971520	20	20971520	20971520	0	STATIC	(null)
4	Shared IO Pool	8388608	8	8388608	8388608	0	STATIC	(null)
5	java pool	4194304	4	4194304	4194304	0	STATIC	(null)
6	ASM Buffer Cache	0	0	0	0	0	STATIC	(null)
7	DEFAULT 8K buffer cache	0	0	0	0	0	STATIC	(null)
8	DEFAULT 16K buffer cache	0	0	0	0	0	STATIC	(null)
9	DEFAULT 32K buffer cache	0	0	0	0	0	STATIC	(null)
10	Data Transfer Cache	0	0	0	0	0	STATIC	(null)
11	In-Memory Area	0	0	0	0	0	STATIC	(null)
12	In Memory RW Extension Area	0	0	0	0	0	STATIC	(null)
13	In Memory RO Extension Area	0	0	0	0	0	STATIC	(null)
14	DEFAULT 2K buffer cache	0	0	0	0	0	STATIC	(null)
15	RECYCLE buffer cache	0	0	0	0	0	STATIC	(null)
16	KEEP buffer cache	0	0	0	0	0	STATIC	(null)
17	memoptimize buffer cache	0	0	0	0	0	STATIC	(null)
18	streams pool	0	0	0	0	0	STATIC	(null)
19	DEFAULT 4K buffer cache	0	0	0	0	0	STATIC	(null)

- `oper_count` indica el número de operaciones realizadas desde el inicio de la instancia.
- `last_oper_type` indica el último tipo de operación realizada.

4.4.3. V$SGASTAT

Muestra información detallada de la SGA.

```
select * from V$SGASTAT order by pool;
```

Solo se muestra un extracto de la salida.

	POOL	NAME	BYTES	CON_ID
1	java pool	free memory	4194304	0
2	large pool	free memory	5242880	0
3	large pool	PX msg pool	15728640	0
4	shared pool	free memory	11456376	0
5	shared pool	archive_lag_target	9632	0
6	shared pool	Pre-Warm Initialized Seti	1152	0
7	shared pool	simulator hash buckets	98304	0
8	shared pool	enqueue_hash	50064	0
9	shared pool	kmgsb circular statistics	162560	0
10	shared pool	object temp hash table la	188416	0
11	shared pool	primem_kfmdsg	8200	0
12	shared pool	ASM KFFD SO	2992	0
13	shared pool	KCB cacherm	106576	0
14	shared pool	AQ skiplist state object	32256	0
15	shared pool	dpslut_kfdsg	4096	0
16	shared pool	PRE_11g UNDOSEG DROP BITV	520	0

4.4.4. V$SGAINFO

Muestra de forma adicional a la información de los componentes de la SGA, la memoria libre disponible.

```
select name, bytes, bytes/(1024*1024) bytes_mb
from V$SGAINFO
order by bytes desc;
```

	NAME	BYTES	BYTES_MB
1	Maximum SGA Size	536867640	511.99687957763671875
2	Shared Pool Size	230686720	220
3	Free SGA Memory Available	209715200	200
4	Startup overhead in Shared Pool	195226024	186.18204498291015625
5	Buffer Cache Size	54525952	52
6	Large Pool Size	20971520	20
7	Fixed SGA Size	8659768	8.25859832763671875
8	Shared IO Pool Size	8388608	8
9	Redo Buffers	8114176	7.73828125
10	Java Pool Size	4194304	4
11	Granule Size	4194304	4
12	Streams Pool Size	0	0
13	In-Memory Area Size	0	0
14	Data Transfer Cache Size	0	0

4.4.5. V$PGASTAT

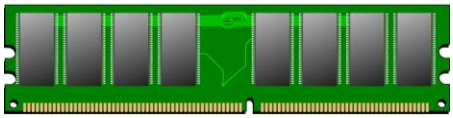
- Muestra el contenido de la PGA.

```
select name, value,
unit, value/(1024*1024) value_mb
from v$pgastat
order by value desc;
```

	NAME	VALUE	UNIT	VALUE_MB
1	maximum PGA allocated	285776896	bytes	272.5380859375
2	aggregate PGA target parameter	209715200	bytes	200
3	total PGA allocated	157816832	bytes	150.505859375
4	PGA memory freed back to OS	134676480	bytes	128.4375
5	total PGA inuse	112465920	bytes	107.255859375
6	aggregate PGA auto target	87690240	bytes	83.6279296875
7	bytes processed	81446912	bytes	77.673828125
8	global memory bound	41943040	bytes	40
9	total freeable PGA memory	24182784	bytes	23.0625
10	maximum PGA used for auto workareas	4918272	bytes	4.6904296875
11	cache hit percentage	100	percent	0.000095367431640625
12	recompute count (total)	88	(null)	0.00008392333984375
13	max processes count	81	(null)	0.00007724761962890625
14	process count	65	(null)	0.00006198883056640625
15	over allocation count	0	(null)	0
16	MGA allocated (under PGA)	0	bytes	0
17	extra bytes read/written	0	bytes	0
18	total PGA used for manual workareas	0	bytes	0
19	maximum MGA allocated	0	bytes	0
20	total PGA used for auto workareas	0	bytes	0
21	maximum PGA used for manual workareas	0	bytes	0

4.5. ADMINISTRACIÓN DE LAS ÁREAS DE MEMORIA.

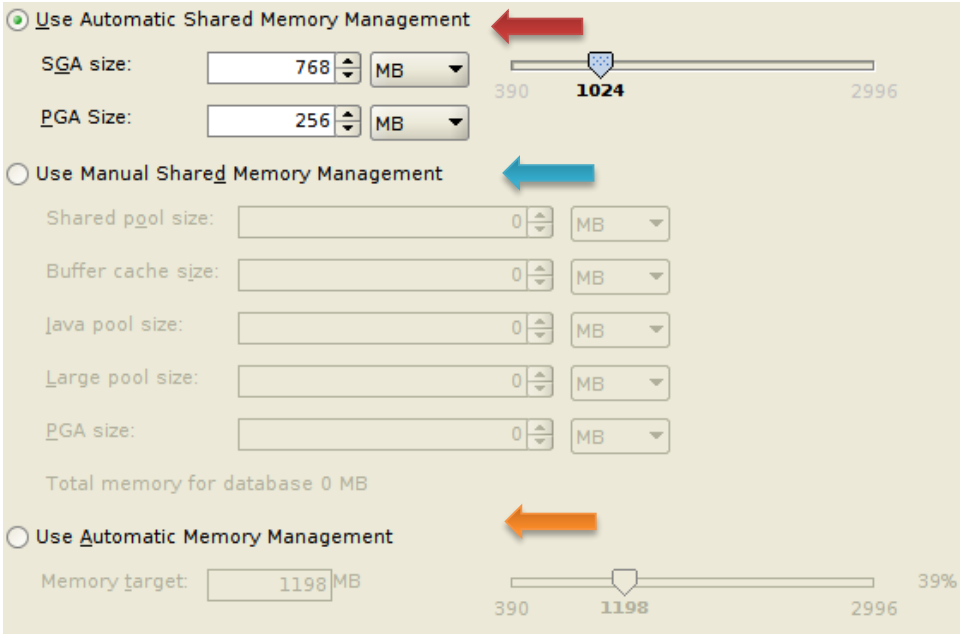
- Tanto la PGA como la SGA son áreas de memoria que deben ser administradas.
- Existen varios métodos para realizar su administración:
  - Organización automática
  - Organización manual.



4.5.1. Organización automática de la memoria.

- La administración es realizada por la propia instancia.
- Para realizar lo anterior, únicamente se debe inicializar el parámetro `memory_target` y opcionalmente el parámetro `memory_max_target`.
  - `memory_max_target` actúa como límite máximo para evitar que el valor de `memory_target` se establezca a un valor mayor por accidente.
  - `memory_target` es dinámico mientras que `memory_max_target` es estático.
- El valor del parámetro `memory_target` permanece relativamente fijo.
- La instancia automáticamente distribuye la memoria en las diferentes áreas de la PGA y SGA.
- Si la demanda de memoria cambia, la instancia automáticamente cambia los valores de asignación de memoria.
- En caso de que la administración automática no esté habilitada, se deberá configurar por separado la cantidad de memoria para la SGA y PGA.
- Si la cantidad total de memoria física a asignar es de más de 4GB, se recomienda realizar la configuración a través de la opción "*Automatic Shared Memory Management".*

La siguiente imagen muestra las opciones de administración empleada por DBCA.



4.5.2. Habilitar la administración automática.

- Se realiza a través de la instrucción `create database` una vez que el parámetro `memory_target` haya sido incluido en el archivo de parámetros.
- A través de DBCA seleccionando la primera o tercer opción.

- En caso de no haberse habilitado, es posible hacerlo posteriormente (requiere reinicio) empleando el siguiente procedimiento:

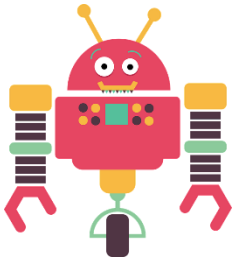
1. Conectarse a la instancia con el privilegio `sysdba`, por ejemplo, con el usuario sys.
2. Calcular el valor mínimo del parámetro `memory_target` de la siguiente forma:

   a. Determinar el tamaño `de sga_target` and `pga_aggregate_target` in megabytes

```
show parameter sga_target
show parameter pa_aggregate_target
```

   b. Determinar el tamaño máximo en MB de la PGA desde que la instancia se inició:

```
select value/1048576
from v$pgastat
where name='maximum PGA allocated';
```



   c. Determinar el valor de la memoria con base a la siguiente fórmula:

```
memory_target = sga_target +
  max(pga_aggregate_target, maximum pga allocated)
```

   d. Con el valor calculado, realizar el cambio de memoria con la siguiente instrucción:

```
alter system set memory_max_target = nM scope = spfile;
```

n corresponde al valor de memoria calculado en el inciso “c”.

   e. Reiniciar la instancia
   f. Ejecutar las siguientes sentencias:

```
alter system set memory_target = nM;
alter system set sga_target = 0;
alter system set pga_aggregate_target = 0;
```

Finalmente, la siguiente sentencia se emplea para monitorear la administración automática:

```
select *
from v$memory_target_advice
order by memory_size;
```

	MEMORY_SIZE	MEMORY_SIZE_FACTOR	ESTD_DB_TIME	ESTD_DB_TIME_FACTOR	VERSION	CON_ID
1	640	1.25	47	0.922	0	0
2	768	1.5	47	0.922	0	0
3	896	1.75	47	0.922	0	0
4	1024	2	47	0.922	0	0

- Observar el nombre de la vista, contiene la palabra *advice.*
- La idea de esta vista es mostrar diferentes escenarios al variar el valor del parámetro `memory_target`.
- La columna `memory_size` muestra diferentes valores que podrían asignarse a este parámetro.

- La columna `memory_size_factor` especifica el factor que se le aplica al valor actual de la memoria. En la imagen anterior, el valor de `memory_target` es 512 MB. Por lo tanto, un factor de 2, genera el valor 1024.
- La columna `estd_db_time` indica el tiempo estimado que tardaría la instancia en procesar la carga de trabajo actual con la cantidad de memoria. Este valor en particular resulta importante ya que a menor cantidad de memoria, los tiempos pueden incrementarse. En el ejemplo los tiempos no varían debido a que la instancia se acaba de iniciar y aun no existe carga de trabajo.

4.6. ADMINISTRACIÓN MANUAL DE LA MEMORIA.

Existen 2 formas para administrar la memoria de forma manual, tanto para la SGA como para la PGA:

- Para la SGA:
  - Automatic Shared memory management
  - Manual Shared memory management
- Para la PGA
  - Automatic PGA memory management
  - Manual PGA memory management.

Notar el uso de la palabra "Automatic" la cual pudiera ser confusa ya que el tema trata de administración automática. Esto se explica a continuación.

4.6.1. Automatic shared memory management

- En esta configuración se especifica de forma manual el total de la memoria que será asignada únicamente a la SGA empleando el parámetro `sga_target` y de forma opcional `sga_max_target`.
  - Con este valor, la instancia realizará la administración automática de la memoria, pero únicamente de la SGA.
- Notar la diferencia con la administración automática vista en la sección anterior en la que se especifica el valor total de memoria a través del parámetro `memory_target`. Este valor comprende también a la PGA.
- La siguiente tabla muestra los componentes de a SGA que son administrados de forma automática y sus respectivos parámetros.
  - Para que la administración sea automática, sus valores deben ser cero.

Componente de la SGA	Parámetro.
Shared pool	`shared_pool_size`
Large pool	`large_pool_size`
Java pool	`java_pool_size`
buffer_cache	`db_cache_size`
streams_pool	`streams_pool_size`
fixed SGA y algunas otras áreas internas	`N/A`

- La siguiente tabla muestra más componentes de la SGA cuya cantidad de memoria no se administra de forma automática:

Componente de la SGA	Parámetro.
Log buffer	log_buffer
Keep buffer cache	db_keep_cache_size
Recycle buffer cache	db_recycle_cache_size
Buffer caches para bloques de datos de tamaño no estándar.	db_nk_cache_size

4.6.2. Manual Shared Memory Management

- En este modo, la cantidad de memoria para los siguientes componentes de la SGA deben ser especificados de forma manual:
  - buffer cache
  - shared pool
  - large pool
  - Java pool

- Adicionalmente, los siguientes parámetros deben ser inicializados a cero.
  - memory_target
  - sga_target

Para regresar a la administración compartida automática a partir del modo manual compartido se realizan las siguientes acciones:

- Ejecutar la siguiente sentencia para determinar el valor de sga_target

```
select (
  (select sum(value) from v$sga) -
   (select current_size from v$sga_dynamic_free_memory)
  ) "sga_target"
from dual;
```

- Establecer los valores de los parámetros:

```
alter system set sga_target=value [scope={spfile|memory|both}]
```

Donde value corresponde al valor obtenido de la consulta.

Revisar el documento correspondiente al ejercicio práctico 04. – Administración manual y automática de las áreas de memoria.