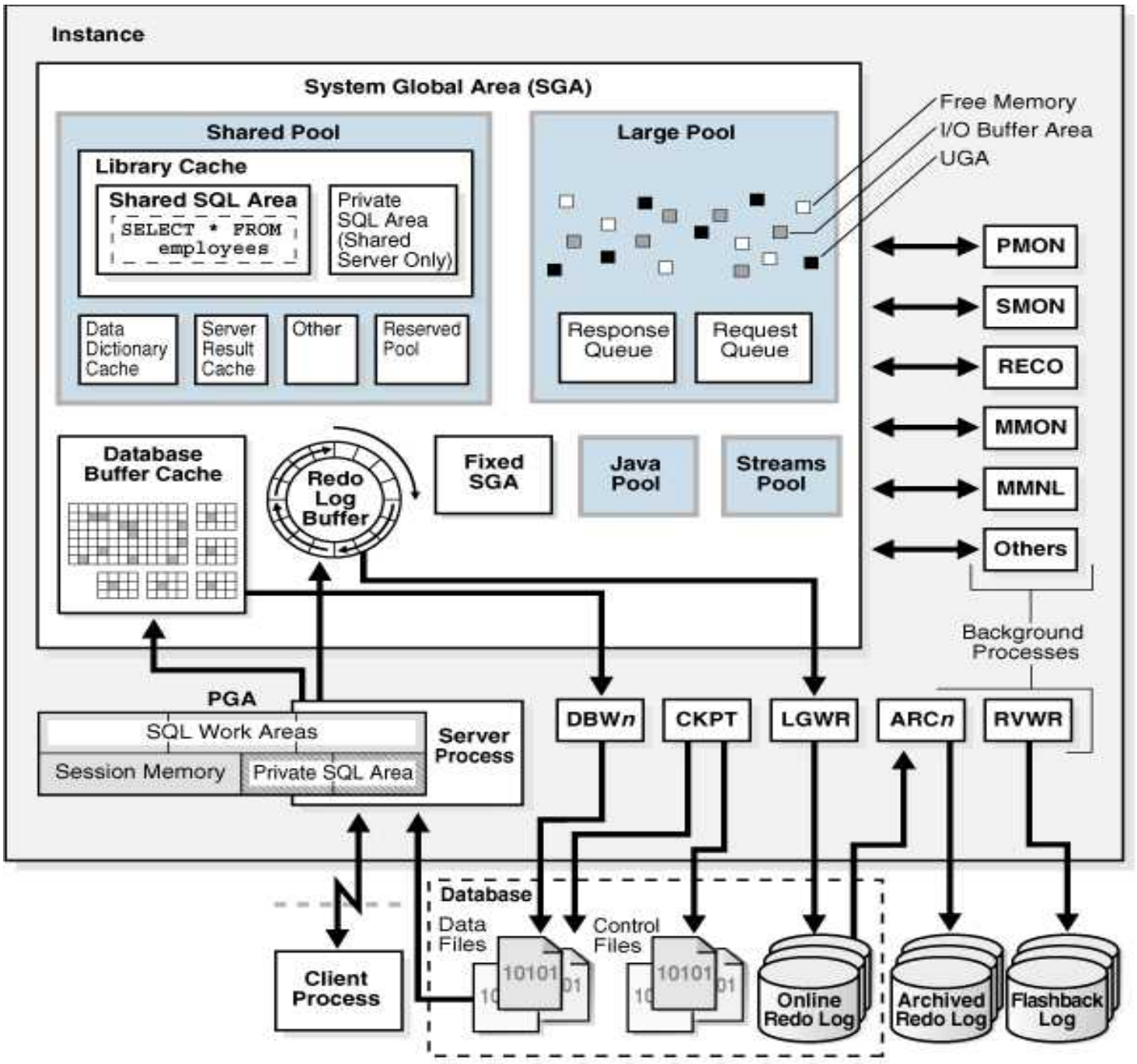
UD2. Arquitectura de Oracle database

Índice

Arquitectura general de Oracle



Oracle crea una serie de procesos servidores que manejan las peticiones de los usuarios que se conectan a la instancia. Una conexión de cliente tiene dos partes: una del lado cliente (un programa como SQLPlus, una aplicación, etc.) y un proceso servidor que capta la petición, la resuelve y devuelve los resultados. Los procesos servidores pueden ser dedicados o compartidos. En los primeros cada proceso cliente tiene un proceso servidor para sí que sirve su petición, es bueno para baja carga y cuando tenemos muchos recursos en el sistema. En modo compartido un repartidor asigna cada petición a una cola de procesos compartidos.

La instancia de Oracle

La **base de datos** vimos que es una colección de archivos en uno o más discos. La base de datos está compuesta por los datafiles, los tempfiles, el controlfile, redo logs, archive logs, archivos de traza...

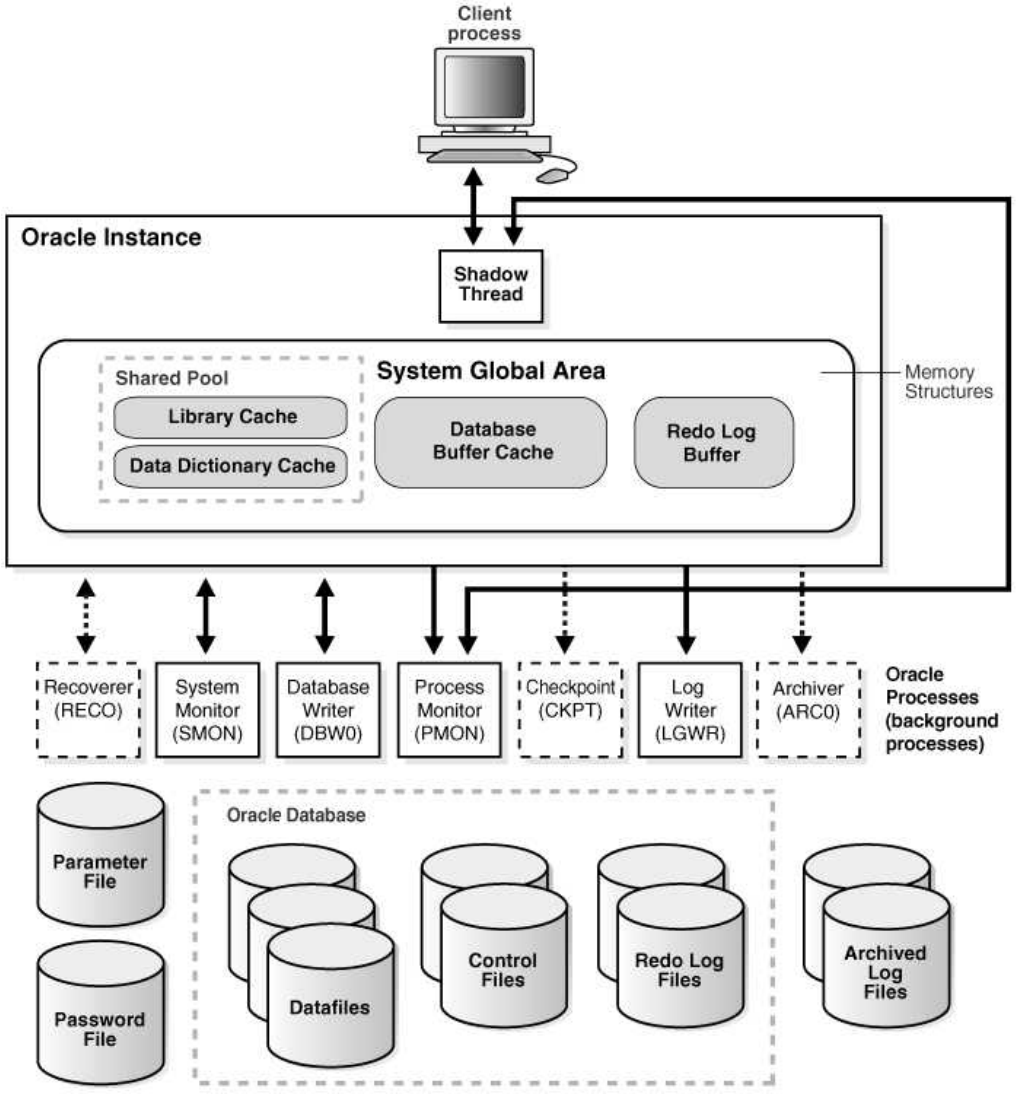
La **instancia** de Oracle que está compuesta por los procesos que corren en el ordenador en modo background y la memoria y es identificada por su ORACLE_SID

Procesos de background que son los procesos o threads que hacen el trabajo de acceder, guardar, monitorizar y recuperar los datos del usuario, los metadatos e información de control que se encuentra en los discos.

La **shared memory** es el área de memoria usada de modo compartido por los procesos de background.

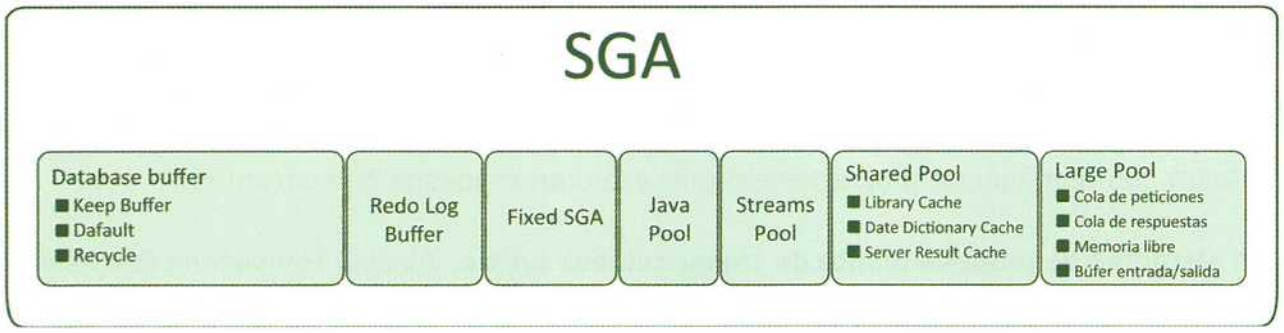
Pocesos Servidor que trabajan para las aplicaciones y usuarios conectados trabajando con la memoria y el almacenamiento temporal PGA. Ejecutan las sentencias SQL y devuelven los resultados al usuario.

Oracle Net. Es una capa de software que habilita a las aplicaciones para comunicarse sobre la red. El Listener escucha las peticiones de conexión y el tnsnames describe las cadenas de conexión



La memoria compartida SGA

La memoria SGA está compartida por un conjunto de procesos asociados a una instancia Oracle y se asigna cuando esta es iniciada. Cuando se para una instancia, esta memoria es liberada. Todos los usuarios conectados a la instancia comparten esta sección de memoria. Los datos cargados en la SGA permiten ser retutilizados para servirlos a varios usuarios sin tener que acceder todo el tiempo a disco. Es por tanto importante indicar en el archivo de inicialización una correcta cantidad dememoria. La SGA está formada por varios componentes, cada uno de ellos sirve a una función determinada y se puede establecer el tamaño de cada uno o bien establecer un tamaño para la SGA global y dejar que Oracle la distribuya como mejor le parezca.



Además de la memoria global compartida, SGA, existe la memoria privada de cada proceso que se denomina Program Global Area o PGA. Este espacio contiene las variables e información de la sesión y un espacio de trabajo SQL, SQL Work Area.

Se dispone de un conjunto de parámetros para configurar de forma manual la gestión de la memoria compartida. Un parámetro interesante es aquel que limita el tamaño máximo que Oracle puede asignar a una instancia y se denomina SGA_MAX__SIZE. Para conocer el valor actual de un parámetro se usa el comando **SHOW PARAMETERS**. El valor de un argumento se puede dar en bytes, de modo que no es necesario usar ninguna letra, o en unidades de medidas con las letras K, M y G. El siguiente ejemplo define el tamaño maximo en 2 GB.

SGA_MAX_SIZE=2G

Tambien se podria haber dimensionado indicando el número de bytes sin ninguna letra:

SGA_MAX_SIZE= 2147483648

Se puede usar el parámetro SGA_TARGET para utilizar la gestión automática. Oracle atendiendo al valor de este parámetro, configura el resto de los valores si estos no están definidos. Este valor nunca puede superar el valor configurado con el parámetro SGA_MAX_SIZE. Si SGA_TARGET=0, la configuración automática está desactivada.

La memoria compartida SGA se divide en diferentes zonas, con el fin de usarlas con propósitos diferentes. La zona principal es la zona de buffer de datos que contiene los datos en memoria, también llamada Database Buffer Caché.

El redo log buffer es también una estructura fundamental. Contendrá todas las transacciones hechas commit que se bajarán a disco en determinados momentos. Mientras están en esas estructuras de memoria.

La shared pool es también una estructura importante que veremos más adelante. En la página siguiente se muestran las partes de la SGA.

Estructura	Traducción	Uso	Parámetros
Database Buffer Cache	Cache búfer para datos	Memoria para la caché de datos	DB_CACHE_SIZE DB_nK_CACHE_SIZE DB_KEEP_CACHE_SIZE DB_RECYCLE_CACHE_SIZE
Java Pool	Sección Java	Memoria para la máquina virtual Java	JAVA_POOL_SIZE
Large Pool	Sección grande	Es opcional en configuraciones especiales	LARGE_POOL_SIZE
Shared Pool	Sección compartida	Espacio para la *library cache*, *dictionary cache* y *result cache*	SHARED_POOL_SIZE RESULT_CACHE_MAX_SIZE RESULT_CACHE_MODE
Streams Pool	Sección interprocesos	Espacio para la comunicación de los procesos	STREAMS_POOL_SIZE
Redo Log Buffer	Búfer para los registros de actualización	Almacena los datos sobre las modificaciones realizadas sobre la base de datos	LOG_BUFFER

La zona Databuffer Cache

El espacio DataBuffer Cache contiene los datos usados más recientemente. Los datos leidos desde los ficheros de datos son volcados a esta área de memoria. Todos los usuarios o aplicaciones comparten los mismos datos de la buffer caché. El servidor de Oracle usa un algoritmo LRU para actualizar los datos: los datos usados más antiguos son eliminados para introducir nuevos datos en esta caché. Cuando se ejecuta un comando DML, los cambios se hacen en memoria y la escritura en memoria secundaria es diferida. Cuando estos datos cambian se registra en el búfer de redo log.

Cuando se procesa una consulta, el proceso servidor busca los bloques necesarios en esta zona. Si el bloque no se encuentra aquí , el proceso servidor lee el bloque de los datafiles y sitúa una copia en el buffer cache (también llamado buffer de datos). Solicitudes posteriores del mismo bloque siguen el mismo camino para encontrar el bloque en la memoria por lo que no se necesitan tantas lecturas físicas. El tamaño de esta ára será el número de buffers (DB_BLOCK_BUFFERS) multiplicado por el tamaño del bloque (DB_BLOCK_SIZE), normalmente 2KB.

Cuanto mayor es el tamaño de esta zona de memoria, menos operaciones de entrada y salida a memoria secundaria habrá, pero más espacio de memoria principal se requerirá.

Se usa el parámetro **DB_CACHE_SIZE** para indicar el tamaño de la caché para los datos. Se usa DB_2K_CACHE_SIZE para la caché de los bloques de 2 kB, DB_4K_CACHE_SIZE para los de 4 kB, y asi sucesivamente.

La siguiente declaración asigna un caché DataBuffer de 2048 megabytes:

```
DB_CACHE_SIZE = 2048M
```

Generalmente el tamaño de este espacio suele ser hasta 2/3 del tamaño de la SGA. Estos valores dependen del tamaño de la base de datos y del número de conexiones que esta tenga que gestionar en paralelo.

Esta zona de memoria se divide en tres caches denominadas Default, Keep y Recycle. La zona Default contiene los datos necesarios para las consultas que ya se han ejecutado.

La vista a emplear para obtener información de la zona DataBuffer Cache de la SGA es V$DB_CACHE_ADVICE. Las siguientes consultas muestran información de interés:

```
SELECT NAME, BLOCK_SIZE, ADVICE_STATUS, SIZE_FOR_ESTIMATE,
SIZE_FACTOR, BUFFERS_FOR_ESTIMATE FROM V$DB_CACHE_ADVICE;
```

```
SELECT NAME, ESTD_PHYSICAL__READ_FACTOR, ESTD_PHYSICAL_READS,
ESTD_PCT_OF_DB_TIME_FOR_READS FROM V$DB_CACHE_ADVICE ;
```

Cuando el porcentaje de información que se lee desde esta caché es alto, el rendimiento también es alto, ya que los accesos a esta caché son mucho más rápidos que en memoria secundaria.

El espacio denominado Keep Buffer se usa para registrar los datos que son muy usados. El parámetro que determina su tamaño es DB_KEEP_CACHE_SIZE. Para obligar que

una tabla se almacene en esta caché se usa la opción BUFFER_POOL KEEP en la cláusula STORAGE. Por ejemplo, la siguiente tabla se crea en el tablespace USERS y los datos que contiene se mantienen en el área Keep Buffer:

```
CREATE TABLE Producto(
        Producto_ID NUMBER(5) PRIMARY KEY,
        Nombre VARCHAR2(30) NOT NULL)
TABLESPACE USERS STORAGE(BUFFER_POOL KEEP);
```

Si se prevé que los datos no serán muy usados, se puede incluir en la zona Recycle Buffer. El tañamo de esta zona se parametriza con DB_RECYCLE_CACHE_SIZE. Con la siguiente tabla se usa el búfer Recycle:

```
CREATE TABLE Contacto(
        Contacto_id number(3) PRIMARY KEY,
        NombreCompleto VARCHAR2(36) NOT NULL,
        Telefono NUMBER(9) NOT NULL)
TABLESPACE USERS STORAGE(BUFFER_POOL RECYCLE);
```

Zonas Shared Pool y Redo Log Buffer

Shared Pool

Guarda información que va a ser compartida por varios usuarios, en concreto: sentencias SQL (para volver a usarlas), información del catálogo sobre por ejemplo el usuario conectado, permisos etc, procedimientos almacenados... El tamaño suele ser no más del 50% de la SGA. El área de memoria Shared Pool está compuesta por tres zonas, la caché de consultas, denominada Library Cache, la caché del diccionario de datos, denominada Dictionary Caché y la caché de los resultados de operaciones SQL y PL/SQL, denominada Result Caché. Se usa el parámetro **SHARED_POOL_SIZE** para dimensionar este espacio. El siguiente ejemplo establece la sección compartida con un tamaño de 128 megabytes:

```
SHARED_POOL_S|ZE = 128M
```

En la zona **Library Cache** se almacenan los comandos SQL ejecutados por si se vuelven a usar. No se almacenan ni los datos, que para ello se usa la zona DataBuffer ni sultado, que para ello se usa el área Result Cache. Se almacena el texto de la sentencia, el árbol de análisis (la versión compilada de la sentencia) y el plan de ejecución que define los pasos a seguir para la ejecución de la sentencia como se determina en el planificador. Ya que la library cache almacena esta información, si una consulta es ejecutada antes de que el plan de ejecución sea sobreescito por otras sentencias más recientes, el proceso servidor no necesita analizar la sentencia por lo que ayuda a mejorar el rendimiento de las aplicaciones.

En la zona data **dictionary cache** se almacena información de los datos del diccionario usados más recientemente como definiciones de tablas y columnas, nombres de usuarios, claves y privilegios. Durante la fase de parse, el servidor busca la información en el diccionario para resolver los nombres de los objetos especificados en la sentencia y validar los privilegios de acceso. Si es necesario el proceso servidor inicia la carga de esta información a partir de las tablas del diccionario (esquema de SYS). Cuando una base de datos es creada se ejecuta un procedimiento llamado catalog.sql que crea una serie de vistas para poder manejar este diccionario (por eso al diccionario de datos también se le llama el “catálogo”.

El parámetro SHARED_POOL_RESERVED_SIZE determina el tamaño máximo de los comandos SQL y la vista que debe utilizarse para obtener información de este espacio es V$SHARED_POOL_RESERVED. La siguiente consulta muestra el espacio usado y libre actual de la memoria Shared Pool:

```
COL Usado HEA “Espacio usado";
COL Libre HEA "Espacio libre";
SELECT USED_SPACE Usado, FREE_SPACE Libre FROM
V$SHARED_POOL_RESERVED;
```

Buffer de Redo Log

En el espacio Redo Log Bufer se almacenan las actualizaciones antes de que se escriban en los ficheros de actualización, los archivos físicos de redo log que se guardan en disco. Se usa el parámetro LOG_BUFFER para indicar su tamaño. El valor que este parámetro tiene por defecto suele ser el suficiente para gestionar las entradas redo logs.

El tamaño correcto varía según el número de transacciones de la base de datos. Si hubiera un nivel muy alto de actualizaciones, es decir, carga alta, se recomienda un valor alto para esta sección. Sin embargo, si lo que más se hace es consultar la base de datos, no es necesario un tamaño grande.

Otras zonas de la SGA

El espacio Java se dimensiona con el argumento JAVA_POOL_SIZE. Para servidores compartidos se usa el parámetro LARGE_POOL_SIZE.

Para la configuración manual de la memoria PGA se utilizan los parámetros PGA_AGGREGATE_TARGET y PGA_AGGREGATE_LIMIT. El primero indica el tamaño de la memoria para una configuración de servidor compartido. La segunda indica el valor que no puede superar el espacio asignado al servidor compartido. Si vale cero, entonces no hay ningún limite. El siguiente ejemplo asigna un tamaño de Program Global Area de 64 megabytes.

```
PGA_AGGREGATE_TARGET = 67108864
```

Para obtener información de las diferentes zonas de la memoria SGA se puede usar la consulta `SELECT * FROM V$SGAINFO;`. Para evaluar la memoria usada se puede emplear otra consulta, `SELECT * FROM V$SGASTAT;`

PRÁCTICA

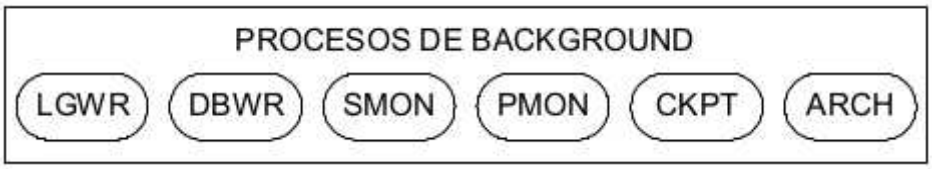
CONFIGURACIÓN DE LA SGA

Indica los parámetros y valores que configurar atendiendo a los siguientes requisitos:
a) El tamaño máximo de la SGA es de 2 gigabytes.
b) El tamaño de la caché de datos es de 512 megabytes.
c) El tamaño del pool compartido es de 128 megabytes.
d) La gestión automática de memoria compartida está activada. El tamaño de la SGA es de 1500 megabytes.

Procesos de Background

Cuando se inicia una instancia, una serie de procesos, más de cincuenta, entran en ejecución y perduran mientras la instancia está arrancada. Estos procesos se encargan de la gestión completa de la instancia y se detienen cuando la instancia está parada. También hay algunos procesos que se paran, aunque la instancia esté arrancada, y otros en ejecución, aunque la instancia esté parada. Estos procesos se denominan **batch.**

El nombre de un proceso batch tiene cuatro caracteres y los principales son ARCn, CKPT, DBWn, LGWR, PMON y SMON, siendo n un número o una letra variable, dependiendo de las copias que se levanten. En la Tabla siguiente se describe la función que tienen estos procesos.



Database Writer (DBW): Escribe los bloques de datos del Buffer cache al disco. Esto se produce cuando ol número de buffers sucios alcanza un nivel, cuando un proceso busca buffers libres y no los encuentra o cuando llega un ckeckpoint.

Log Writer (LGWR): Escribe los datos del redo log buffer al disco (a los archivos de redo log) que son escritos de forma secuencial. Se produce cuando el buffer está a lleno en su tercera parte, cada interrupción (normalmente cada 3 segundos), antes de que DBWR haga su trabajo) o cuando se confirma unha transacción (Se hace COMMIT).

Checkpoint (CKPT): En momentos específicos, todos os buffers de la SGA en la base de datos son escritos por los DBWs.

El Checkpoint es el responsable de ordenar a éstos ol comienzo de la escritura. Así mismo escribe en el archivo de control (control file) la información del último checkpoint realizado.

System Monitor (SMON): Realiza la recuperación de los datos ante una caída de la instancia para que la información sea estable. También se encarga de compactar el espacio contiguo de los tablespaces.

Process Monitor (PMON): Realiza la recuperación cuando un proceso de usuario falla. Es por lo tanto el responsable de borrar la cache y liberar los recursos que el proceso que falló tuviese asignados.

Archiver (ARC): Cuando la base de datos está configurada en modo archive log uno o varios procesos copian los archivos de redo log a un archivo de backup, este archivado tiene lugar cuando los archivos de redo log están llenos o son cambiados.

Se puede usar la vista V$BGPROCESS para conocer los procesos en segundo plano o background que la instancia ejecuta. Para todos los procesos iniciados por la instancia se usa la vista V$PROCESS.

La columna que almacena el nombre del proceso se llama PNAME, por lo que la siguiente consulta muestra información de los procesos cuyo nombre es LGWR, PMON o DBW:

```
SELECT * FROM V$PROCESS WHERE PNAME IN( ' LGWR' , 'PMON' , 'DBW');
```

Otra columna de interés es PGA_USED_MEM, que nuestra la cantidad de memoria PGA actual que el proceso está usando. La siguiente consulta muestra este valor para el proceso con nombre DBW:

```
SELECT PGA_USED_MEM FROM V$PROCESS WHERE PNAME='DBW';
```

Las columnas más interesantes de la vista VSPROCESS son las siguientes:

ADDR: dirección del proceso.
PID: process identification. Contiene el identificador del proceso.
SOSD: contiene el identificador del hilo o del proceso del sistema operativo.
EXECUTION_TYPE: THREAD, tipo de ejecución del sistema operativo.
TRACE_FILE: ubicación y fichero de la traza del proceso. Tiene extensión .trc.
PGA_USED_MEM: cantidad de memoria privada que está usando el proceso.
PGA_ALLOC_MEM: cantidad de memoria privada asignada al proceso.
PGA_FREEABLE_MEM: cantidad de memoria privada que se le añade al proceso.
PGA_MAX_MEM: cantidad máxima de memoria privada asignable al proceso.
CPU_USED: cantidad de CPU que el proceso está usando.

Cuando un proceso tiene un tamaño de PGA asignado demasiado pequeño requiere operaciones de lectura y escritura en el espacio de tablas temporal para llevar a cabo las acciones que el usuario ejecuta, lo que ralentiza el proceso. El tamaño total asignado a los procesos debe ser entre un 20% y un 50% de la memoria (el resto hasta el 100% para la SGA) dependiendo de si hay una tasa muy alta de consultas o no.

Para evaluar el comportamiento de la zona PGA se puede usar la vista dinámica V$PGA_TARGET_ADVICE. Cuando se requiere aumentar el tamaño de PGA asignado a un proceso, Oracle realiza un redimensionamiento de este, y esto se puede realizar más de una vez. Estos valores se pueden analizar a través de la vista V$SQL_WORKAREA_HISTOGRAM y sus columnas ONEPASS_EXECUTIONS y MULTIPASSES_EXECUTIONS. Con valores altos de estos, el rendimiento se ve degradado considerablemente. La siguiente consulta muestra esta información:

```
COL LimInf HEA "Limite inferior (KB) "
COL Unavez HEA "Redimensionamiento de una vez";
COL Masveces HEA "Redimensionamiento de más de una vez";
SELECT LOW_OPTIMAL_SIZE/1024 LimInF, ONEPASS_EXECUTIONS Unavez,
MULTIPASSES_EXECUTIONS Masveces FROM V$SQL_WORKAREA_HISTOGRAM;
```

PRÁCTICA

OBTENER INFORMACIÓN DE LOS PROCESOS

Usando la vista V$PROCESS confecciona las consultas siguientes:
a) Mostrar la ubicación y el fichero de traza de los procesos LGWR y DBW.
b) Con relación a la memoria privada de los procesos, mostrar la cantidad que se está usando, la asignada y la que se puede asignar de los procesos ARC, CKPT y PMON.
c) Mostrar la cantidad de CPU usada del proceso DBW.
d) Mostrar la cantidad de memoria de tipo PGA usada en total.

La vista V$SGAINFO permite obtener más información sobre la memoria compartida. La siguiente consulta muestra información muy interesante sobre la memoria SGA:

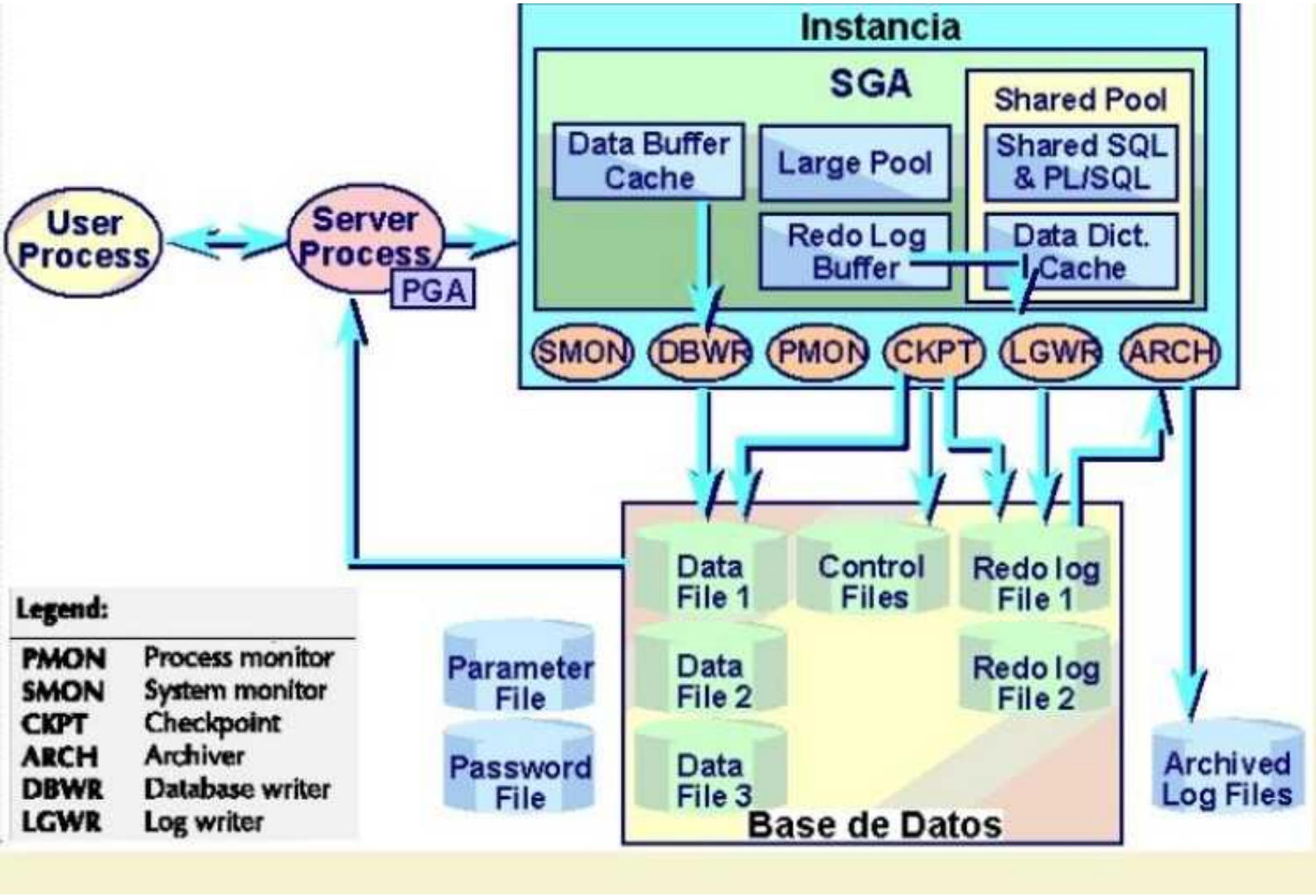
```
SELECT NAME, BYTES FROM V$SGAINFO;
```

Otra vista muy completa es V$MEMORY_DYNAMIC_COMPONENTS, que muestra información sobre la estructura dinámica de la memoria.

La consulta siguiente muestra el nombre de la estructura, su tamaño actual, el tamaño del gránulo y la fecha y hora de la última operación hecha en ella:

```
SELECT COMPONENT, CURRENT_SIZE, GRANULE_SIZE, LAST_OPER_TIME FROM V$MEMORY_DYNAMIC__COMPONENTS;
```

Proceso de una sentiencia DML



Proceso de una consulta

Es importante conocer cuales son las principais etapas en el proceso de consulta de un usuario a los datos que residen en la base de datos.

Parse: En esta etapa, el proceso de usuario envía la consulta al proceso del servidor con una solicitud de compilación de la consulta. El proceso servidor controla la validez del comando y utiliza el área del SGA conocida como Shared Pool para compilar la sentencia. Al finalizar esta fase el proceso servidor devuelve el estado (éxito o fallo) al proceso de usuario.

Execute: Durante esta fase el proceso servidor se prepara para recuperar los datos.

Fetch: Durante esta fase, las filas que son recuperadas por la consulta son devueltas por el servidor al usuario.

Proceso de una transacción

Una sentencia del lenguaje de manipulación de datos como por ejemplo la actualización de un valor por otro pasa por el siguiente proceso:

Parse: Similar a lo que venimos ya para las consultas al principio del tema.

Ejecución: En este caso, la ejecución lleva pareja cinco pasos fundamentales:

1. El proceso servidor lee los bloques de datos y segmentos de rollback del buffer de datos y en caso en que no estén los coge de los archivos de disco.

2. El proceso servidor hace una copia de los bloques leídos en los buffers de datos

3. El proceso servidor sitúa bloqueos en los datos escribiendo esa información en el cache de diccionario de la Shared Pool

4. El proceso servidor graba los cambios que se realizan (valor antiguo y nuevo en el buffer redo)

5. El proceso servidor graba la imagen anterior en el segmento de rollback y el valor nuevo en los buffers.

Caso **commit** (confirmación de la transacción):

a. El proceso servidor sitúa un número de validación de cambio en el sistema en el buffer redolog.

b. LGWR realiza una anotación contigua de todas las entradas en el buffer redolog hasta e incluida la transacción confirmada.

c. El usuario es informado que la confirmación fue realizada correctamente.

d. El servidor graba la información para indicar que la transacción finalizó y desbloquea los recursos.

Caso **rollback** (vuelta atrás de la transacción): La transacción es deshecha, la entrada de rollback es escrita en el buffer de datos.

La base de datos

Una base de datos, identificada por su DB_NAME representa las estructuras físicas y está compuesta por los archivos guardados en disco por el sistema operativo.

Datafiles : Son archivos de datos: Almacenan el diccionario de datos, objetos del usuario e imágenes anteriores de datos que son modificados por las transacciones actuales. Una base de datos contiene por lo menos un archivo de datos.

Archivos de redo log: Contienen una grabación de los cambios hechos en la base de datos para asegurar su reconstrucción en caso de fallos. Una base de datos precisa por lo menos dos. Las anotaciones en los redo logs son más rápidas que las de los datos ya que se hacen secuencialmente. Se graba sólo información mínima ya que se pueden validar varias transacciones de golpe con una sola anotación. El tamaño de la transacción no afecta a la cantidad de tiempo necesaria para una operación de confirmación.

Contolfile: son archivos de control, contienen la información necesaria para mantener y verificar la integridad de la base de datos. Toda base de datos precisa por lo menos uno. El controlfile contiene los path a los datafiles, archivos de redolog e información sobre el último checkpoint realizado.

Archivos de trazas (*.trace) y alertas (alert.log): Da información de los errores y de las trazas (parámetro SQL_TRACE la true o cuando un proceso de background tiene un error) Se lanzan dónde indican los parámetros **BACKGROUND_DUMP_DEST** y **USER_DUMP_DEST**

redo logs

Cada vez que se hace una modificación de la estructura de la base de datos, Oracle la registra en los ficheros de actualización. Estos ficheros también se denominan redo log porque su función es la de restaurar una instancia parada o parar la restauración en caso de que un fichero de datos se dañe. Cuando los ficheros de actualización se llenan, los datos más antiguos son eliminados para hacer espacio a los nuevos. Cuando se detecta que un fichero de datos está dañado, la información de actualización se aplica a la última copia de seguridad estable.

Los archivos de actualización o redo log se organizan en grupos compuestos por una colección de archivos replicados. Asi, un grupo es un conjunto de copias de un fichero redo log. Se recomiendan dos grupos como minimo con un fichero de actualización en su interior. Los ficheros de actualización dentro de un grupo se llaman miembros y tienen el mismo identificador secuencial. Este número es diferente para cada grupo, pero es el mismo para sus miembros, y se registra en el fichero de control y en la cabecera de todos los ficheros de datos.

La definición de cada grupo, tamaño y número de miembros se hace cuando se crea la base de datos. Asi, el número máximo de grupos y el tamaño máximo de cada uno están inicialmente predeterminados. Las modificaciones son registradas en todos los ficheros de un grupo de forma replicada. Cuando se llena un grupo, se pasa al siguiente. Cuando todos los grupos están llenos, se comienzan a reescribir nuevas modificaciones en el primer grupo, eliminando las modificaciones más antiguas, de tal forma que se encuentran sumergidas en ciclos constantes. Se hace necesario establecer un protocolo de copias de seguridad de esta información antes de que sea eliminada. Generalmente, los ficheros de redo log tienen la extensión .log.

Estos archivos pueden estar en diferentes estados, CURRENT significa que este grupo es aquel en el que se está escribiendo en este momento. Si cada grupo sólo tiene un miembro el log no está multiplexado. Esto está bien para una base de datos pequeña, pero si crece, sería deseable tener los log multiplexados. Es recomendable (aínda que no obligatorio) que todos los grupos tengan el mesmo número de miembros

A continuación, se detalla la creación de los ficheros de actualización al crear una base de datos llamada TiendaInformatica usando el comando CREATE DATABASE.

```
CREATE DATABASE TiendaInformatica
LOGFILE
GROUP l ('d:\oradata\tienda\fich_log0ld.log','e:\oradata\tienda\fich_log01e.log') SIZE 2M,
GROUP 2 ('d:\oradata\tienda\fich_|og02d.log','e:\oradata\tienda\fich_log02e.log') SIZE 2M,
GROUP 3 ('d:\oradata\tienda\fich_log03d.log','e:\oradata\tienda\fich_log03e.log') SIZE 2M,
GROUP 4 ('d:\oradata\tienda\fich_log04d.log','e:\oradata\tienda\fich_log04e.log') SIZE 2M,
MAXLOGFILES 8
MAXLOGMEMBERS 2
LOG_FILES 4
```

Creación de redo logs

Cuando miramos el log de la traza del LGWR y vemos que tiene que esperar demasiado para la escritura significa que tenemos grupos de menos e tenemos que crealos. Para añadir un miembro a un grupo con sqlplus usaremos las sentencias que añaden un grupo nuevo o un nuevo membro a un grupo existente:

ALTER DATABASE ADD LOGFILE GROUP <número> ('<destino1>'[,'<destino2>'...]) SIZE <p.e. 500K>;

ALTER DATABASE ADD LOGFILE MEMBER '<destino>' TO GROUP <número>;

Supongamos que tenemos los redo log multiplexados en dos discos, uno de ellos está fallando o quedando sin espacio y debemos llevar los redo logs a otro disco con más capacidad. En este caso podemos alterar la base de datos renombrando los archivos de log. Primero pararemos la base para que nadie escriba en los redo log. Copiaremos los archivos al nuevo disco. Levantaremos la base de datos en modo mount y usaremos la sentencia siguinte y ya podemos abrir la base:

ALTER DATABASE RENAME FILE '<orixe1>','<orixe2>' TO '<destino1>','<destino2>';

Borrado de redo logs

Podemos borrar un grupo o miembro de los redo logs desde el sqlplus con las sentencias siguientes que sólo podrán ser aplicadas a redo logs que esten en estado inactive. Esto se ve mirando la vista V$LOG :

SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;

ALTER DATABASE DROP LOGFILE GROUP <numero>;

ALTER DATABASE DROP LOGFILE MEMBER <'arquivo'>;

Se el grupo que queremos borrar no está inactivo debemos cambiar el estado haciendo que el LGWR se dedique a otro redo log. Esto lo hacemos con la sentencia :

alter system switch logfile;

Este comando conlleva el switching de los grupos, aunque estos no se hayan almacenado. Si se quiere que este proceso espere a que los registros de actualización sean almacenados antes, se ejecuta el comando

ALTER SYSTEM ARCHIVE LOG CURRENT;.

Para esto el modo ARCHIVE, debe estar activado. Para mostrar información al respecto, se puede ejecutar el comando **ARCHIVE LOG LIST**. Por último, el comando ALTER SYSTEM SWITCH ALL LOGFILE; obliga a que se intercambien todos los grupos de todas las instancias.

Cuando se produce un cambio de log, es decir, se comienza a escribir en un grupo nuevo, el servidor asigna un número de secuencia de log para identificar a dicho grupo de redo log.

Sea automático o forzado, se inicia un evento que se llama punto de control. Estos puntos

de control se producen en cada cambio de log, cuando se cierra una instancia en modo no abortivo o cuando lo fuerzan los parámetros de inicialización LOG_CHECKPOINT_TIMEOUT, FAST_START_IO_TARGET y LOG_CHECKPOINT_INTERVAL. Estos parámetros especifican, respectivamente, el intervalo máximo de tiempo entre puntos de control, el número de bloques que recuperar y el número de bloques de redo log por recuperar ante un fallo. Un punto de control se puede forzar con el comando **ALTER SYSTEM CHECKPOINT;**.

Cuando circularmente Oracle va sobrescribir un arquivo de redo, podemos pedirle que lo archive en los archive log files cuando esté lleno. Así el estado del redo log cambiará a current. Esto es usado para las políticas de backup y recuperación que veremos más adelante. Para archivar el archive log tenemos que tener la base de datos corriendo en modo archivelog.

Consultando información sobre los redo logs

Para obtener información de los grupos, de los miembros e históricos sobre los ficheros de actualización, puedes consultar las vistas VLOG, VLOGFILE y V$BLOG_HISTORY.

La vista V$LOG contiene información muy completa de cada grupo. La siguiente consulta muestra, entre otros, el código del grupo actual, la secuencia en el ciclo actual, el estado, su tamaño y el número de ficheros que contiene:

SELECT GROUP#, SEQUENCE#, STATUS, BYTES, MEMBERS, ARCHIVED, FIRST_CHANGE#, FIRST_TIME FROM VSLOG;

	GROUP#	SEQUENCE#	STATUS	BYTES	MEMBERS	ARCHIVED	FIRST_CHANGE#	FIRST_TIME
1	1	172	CURRENT	209715200	1	NO	13271497	07/11/21
2	2	170	INACTIVE	209715200	1	NO	13148766	06/11/21
3	3	171	INACTIVE	209715200	1	NO	13208574	06/11/21

Los números de secuencia se incrementan cada vez que se pasa de un grupo a otro. Este mecanismo se denomina switch o intercambio. Tienen especial importancia el estado en el que se encuentra el grupo

CURRENT, usándose

INACTIVE, inactivo

el código del grupo y el código de secuencia.

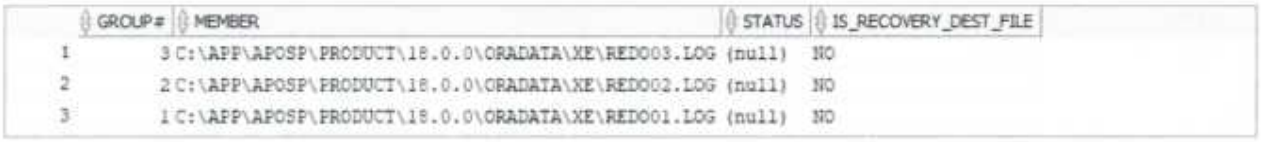
El valor de STATUS es interesante para conocer el estado en el que se encuentra cada grupo. Si contiene el valor ACTIVE, dicho grupo no es el que se está usando, pero si que es necesario para cualquier tarea de restauración de la instancia. Si un grupo está en estado CURRENT, se trata del grupo que se está usando. Si es INACTIVE, significa que el grupo no es necesario para cualquier tarea de restauración de la instancia y, si es UNUSED, es que dicho grupo nunca se ha usado.

Otras columnas interesantes que se pueden seleccionar de la vista V$LOG son BYTES para conocer el tamaño en bytes, MEMBERS para conocer el número de ficheros replicados en cada grupo, ARCHIVED para saber si los grupos están almacenados o no, FIRST_CHANGE# para mostrar el número SCN más pequeño escrito en el grupo y FIRST_TIME para la fecha de este número.

Para obtener información sobre los miembros, se consulta la vista VSLOGFILE.

La siguiente consulta muestra el número del grupo, el nombre del fichero, su estado y si el fichero de actualización se almacena en la zona de recuperación rápida. Los estados posibles en los que puede estar un miembro son DELETED cuando el fichero ha sido eliminado, INVALID cuando no es accesible, STALE cuando está incompleto o NULL cuando el fichero está siendo usado.

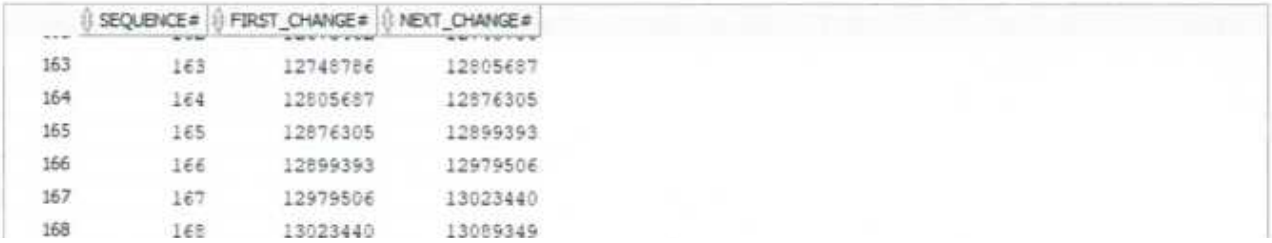
SELECT GROUP#, MEMBER, STATUS, IS_RECOVERY_DEST_FILE FROM VSLOGFILE;



	GROUP#	MEMBER	STATUS	IS_RECOVERY_DEST_FILE
1	3	C:\APP\APOSP\PRODUCT\18.0.0\ORADATA\XE\REDO03.LOG	(null)	NO
2	2	C:\APP\APOSP\PRODUCT\18.0.0\ORADATA\XE\REDO02.LOG	(null)	NO
3	1	C:\APP\APOSP\PRODUCT\18.0.0\ORADATA\XE\REDO01.LOG	(null)	NO

Otra vista sobre ficheros de actualización que almacena información respecto a sus históricos es V$LOG_HISTORY. La siguiente consulta muestra el número de secuencia del grupo sobre el que se muestran históricos, el número más pequeño y el más grande escrito en el grupo.

SELECT SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE# FROM VSLOG_HISTORY;



	SEQUENCE#	FIRST_CHANGE#	NEXT_CHANGE#
163	163	12748786	12805687
164	164	12805687	12876305
165	165	12876305	12899393
166	166	12899393	12979506
167	167	12979506	13023440
168	168	13023440	13089349

El número de cambio en datos es un número que identifica a cada cambio efectuado en los ficheros de actualización. De este modo, se puede consultar cada modificación efectuada en relación con un registro de datos. También se puede hacer durante un periodo de tiempo. Toda esta información se puede obtener consultando la vista del diccionario de datos llamada FLASHBACK_TRANSACTION_QUERY. Las siguientes consultas la usan y ofrecen información al respecto.

SELECT XID, START_SCN, COMMIT_SCN, LOGON_USER, OPERATION, UNDO_SQL FROM FLASHBACK_TRANSACTION_QUERY;

El fichero de control (CONTROLFILE)

Un fichero de control contiene información sobre la base de datos, como su nombre, fecha y hora de creación, la ubicación de ficheros de datos y ficheros de actualización, el número de secuencia de los ficheros de actualización, información de los checkpoint y un largo etcétera.

Cada vez que se lleva alguna acción que modifica la estructura de la base de datos, se registra en el fichero de control, que es el primer fichero que se abre cuando se inicia una instancia. Este permite abrir el resto de ficheros necesarios para el mantenimiento óptimo de la base de datos. En caso de que este fichero falle, la instancia no será arrancada y pasará al modo NOMOUT.

El archivo de control contiene información de control de la base de datos y es de vital importancia para ella. La base de datos no podrá ser abierta si el archivo de control no estuviera o fuera dañado. Por esto,Oracle recomienda no tener sólo un archivo de control, si no haberlo replicado en por lo menos dos copias.

Lo más adecuado es replicar este fichero de control. Existen también metodologías para reiniciar la base de datos usando copias de seguridad del fichero de control. La creación de cada fichero de control replicado se hará en la creación de la base de datos. Se deben crear varios y, a ser posible, en discos separados. Estos ficheros también se pueden agregar una vez creada la base de datos. Generalmente, los ficheros de control tienen la extensión .ctl.

Multiplexar o replicar los controlfiles requiere cerrar la base de datos, ya que se copiarian los ficheros en las otras ubicaciones, por lo que se modifica el archivo de parámetros añadiendo en CONTROL_FILES los nuevos ficheros y se iniciaria de nuevo la base de datos. La siguiente secuencia de comandos indicaria dos supuestos ficheros de control y, evidentemente, los dos ficheros deben estar creados con anterioridad:

STARTUP NOMOUNT;

ALTER SYSTEM SET CONTROL_FILES = ’e:\oradata\tienda\contr0102.ctl’ ,’d:\oradata\tienda\controlel.ctl’, SCOPE=SPFILE;

STARTUP

SELECT name FROM V$CONTROLFILE;

Se puede consultar información de los ficheros de control como su estado, su nombre y ubicación, el tamaño del bloque y el número de bloques. Para ello, se puede ejecutar esta consulta:

SELECT * FROM V$CONTROLFILE;

Otros valores importantes sobre los ficheros de control se pueden obtener con la consulta:

SELECT * FROM V$CONTROLFILE_RECORD_SECTION;

o consultando la vista V$DATABASE.

Debido a la importancia capital que tiene el controlfile vamos a ver unas estrategias de recuperación básicas. El primero que hay que tener en cuenta es que el controlfile es escrito con la base de datos arrancada y de manera constante por lo que nunca debemos copiar un controlfile de una base de datos que esté en funcionamiento.

Backup del controlfile

Podemos hacer dos tipos de backup del controlfile. Un backup binario que será idéntico al controlfile en uso. Para hacer esto o bien paramos la base y copiamos el archivo o bien lo hacemos en caliente con la sentencia

ALTER DATABASE BACKUP CONTROLFILE TO '<destino>';

También podemos hacer un backup de traza

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

Este tipo de backup genera un archivo de traza en udump que contiene la sentencia precisa para recrear el controlfile (ver apartado siguiente).

Perda de una imagen del controlfile

En este caso tenemos dos opciones. Parar la base de datos (si no lo está). Copiar la otra imagen donde teníamos la anterior y levantar la base. También podemos borrar esa entrada en el PFILE pero estamos perdiendo capacidad de recuperación.

Recuperación si el controlfile está corrupto

Si todas las copias se corrompieron, y tenemos un backup del controlfile deberíamos usarlo. En caso de no tener backup del controlfile debemos crear otro como se ve en el apartado siguiente. Esta opción sólo deberiamos tomarla en caso de extrema necesidad pues recrear un controlfile de esta forma puede dañar los archivos de datos y de redo log.

Para recrear un controlfile debemos:

1. Crear una lista de los archivos de la base de datos: controlfiles, datafiles y redo logs. Si tenemos la base abierta podemos usar estas sentencias, si no tenemos que identificarlos a mano:

```
SELECT MEMBER FROM V$LOGFILE;
SELECT NAME FROM V$DATAFILE;
SELECT VALUE FROM V$PARAMETER WHERE NAME = 'CONTROL_FILES';
```

2. Para a base de datos, fai un backup dos arquivos anteriores e arríncaa con modo nomount

3. Crea un nuevo controlfile con la sentencia (es un ejemplo) que depende de como tengamos creada la base de datos. Usaremos la cláusa noresetlogs a menos que perdiésemos también algún redo log, con lo que tendríamos que usar resetlogs. En lugar de saber ésto, é mellor tener un script con la traza que sacamos antes :

CREATE CONTROLFILE REUSE DATABASE "rodeira" NORESETLOGS ARCHIVELOG MAXLOGFILES 16 MAXLOGMEMBERS 3 MAXDATAFILES 100 MAXINSTANCES 8 MAXLOGHISTORY 454
LOGFILE
GROUP 1 ('C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\REDO01.LOG','C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\REDO01A.LOG') SIZE 10M,

GROUP 2 ('C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\REDO02.LOG','C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\REDO02A.LOG') SIZE 10M,

DATAFILE

'C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\SYSTEM01.DBF',

'C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\UNDOTBS01.DBF',

'C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\SYSAUX01.DBF',

'C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\USERS01.DBF',

'C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\INDEX.DBF',

'C:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\TEMP01.DBF'

CHARACTER SET WE8MSWIN1252;

4. Recuperación del control file binario: Parar la base y copiar la copia binaria. Levantar la base.

RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL

Recuperar poniendo los redo log online.

ALTER DATABASE OPEN RESETLOGS

Recuperar el control file creando uno nuevo: Ejecuta las sentencias de la traza del controlfile, arranca la base en modo mount. Si antes paramos la base normalmente no precisaremos RECOVER.

Crea el controlfile con la opción RESETLOGS o NORESETLOGS dependiendo si perdimos o non los redo log online.

ALTER DATABASE OPEN

Creamos el tablespace temporal.

Si creamos el controlfile con la opción RESETLOGS tendremos que abrir la base de datos con:

ALTER DATABASE OPEN RESETLOGS

Ficheros de inicialización

Existen dos ficheros de configuración importantes que se leen en el inicio de una instancia y se llaman init y spfile. Contienen información de los parámetros estáticos de inicialización de la instancia, memoria y también contiene el path al archivo de control para poder abrirlo. El proceso que arranca la instancia lee estos ficheros para configurar la instancia y la base de datos. El fichero init<Sid>.ora contiene los parámetros de inicialización y el fichero spfile<sid>.ora los parámetros del servidor de la instancia. El valor <sid> hace referencia al identificador de la instancia, compuesto por el identificador de la base de datos y a veces el número de la instancia. Asi, un valor para sid puede ser XE1 Los ficheros anteriores se llamarian para este sid, initxe1.ora y spfilexe1.ora, respectivamente.

La diferencia principal entre el fichero SPFILE y el fichero PFILE (el init.ora) es que el primero es un fichero binario y permite modificar sus valores de forma dinámica, es decir, sin parar la base de datos. Los valores de PFILE se leen al iniciar la instancia y estos valores pueden ser modificados, pero en la siguiente vez que se inicie la instancia. El fichero SPFILE sí puede ser incluido en las copias de seguridad; sin embargo, PFILE no.

La ubicación predeterminada de estos ficheros es $ORACLE_HOME/dbs (en Windows %ORACLE_HOME%\database, pero puede que se encuentre en otro lugar si en el proceso de instalación se ha elegido un directorio raíz diferente.

PRÁCTICA

OBTENER INFORMACIÓN DEL SPFILE

a) ¿Cuál es la ubicación del fichero init<sid>.ora? Y la del fichero spfile<sid>.ora?
b) ¿Qué contienen estos ficheros? ¿Contienen un enlace a otro fichero a través del parámetro IFILE?
c) Para que se usa el parámetro SPFILE?
d) Si el fichero abierto contiene un enlace a otro fichero de configuración, abre este nuevo fichero. Una vez abierto el fichero de parámetros de inicialización correspondiente, anota los valores de los parámetros de la tabla que hemos visto de la SGA y los valores de SGA_MAX_SIZE, DB_CACHE_SIZE, SHARED_POOL_SIZE, SGA_TARGET.
e) ¿Qué ocurre cuando se usa PFILE en el comando STARTUP?
f) De aquellos parámetros no definidos en el fichero comprueba su valor actual.

Modificación dinámica de los valores de los parámetros de inicialización

La modificación dinámica de los valores de los parámetros de la base de datos hace referencia a su cambio de valor estando la base de datos abierta. Atendiendo a esta consideración, un cambio de valor puede afectar a la sesión actual, a las sesiones futuras o a las dos a la vez.

Para llevar a cabo esta operación se usan los comandos **ALTER SESSION SET** y **ALTER SYSTEM SET**.

Cuando se cambia el valor de un parámetro de configuración, este nuevo valor se puede anotar en el fichero de configuración, en memoria o en ambos sitios. Si es en memoria el nuevo valor afecta solo a la instancia actual, y cuando se anota en el fichero de configuración a las sucesivas instancias.

La sintaxis del comando ALTER SESSION SET es la siguiente:

ALTER SESSION SET parámetro = valor;

Este comando se usará cuando se desee dar valor a un parámetro para la sesión actual y, por tanto, solo afectará a esta y nunca a sesiones futuras. Esta operación se puede llevar a cabo para hacer tareas de pruebas, ya que cualquier cambio no afectará a la sesión siguiente.

La sintaxis del comando ALTER SYSTEM SET es más completa, ya que permite argumentar un parámetro con más opciones. Dicha sintaxis es la siguiente:

ALTER SYSTEM SET parámetro=valor [DEFERRED] [SCOPE=SPFILE | MEMORY | BOTH];

Este comando usa las opciones DEFERRED y SCOPE para indicar el modo en el que afecta el nuevo valor que se le da a un parámetro de inicialización. Asi, si se usa DEFERRED el cambio solo afectará a las futuras sesiones y solo tendrá sentido si se usa la opción MEMORY en SCOPE.

La cláusula SCOPE admite también el valor SPFILE. Cuando se usa de este modo, la acción modifica el parámetro en el fichero de parámetros SPFILE y, por tanto, afectará a sesiones futuras, pero no a la actual. Evidentemente, esta opción se usa cuando se dispone de ficheros de parámetros del servidor.

Si en SCOPE se indica el valor BOTH, la modificación se realiza tanto en memoria como en el fichero SPIFLE. Así, el cambio afecta tanto a la sesión actual como a las sesiones futuras. Por ejemplo, el comando siguiente cambia el valor del parámetro CONTAINER para que afecte solo a la sesión actual:

```
ALTER SESSION SET CONTAINER=XEPDB1;
```

Sin embargo, el siguiente ejemplo afecta a las sesiones futuras y no a la actual:

```
ALTER SYSTEM SET CONTAINER=XEPDB1 SCOPE=SPFILE;
```

Para que afecte, tanto a la sesión actual como a sesiones futuras, se escribe el comando:

```
ALTER SYSTEM SET CONTAINER=XEPDB1 SCOPE=BOTH;
```

Otra opción interesante es la cláusula RESET de ALTER SYSTEM. Esta opción elimina el valor del parámetro en el fichero SPFILE, con el fin de que, en la siguiente sesión, dicho parámetro tome el valor definido por defecto. El ejemplo siguiente elimina de SPFILE el valor definido para el tamaño en bytes del búfer para los ficheros de actualización y que se llama LOG_BUFFER:

```
ALTER SYSTEM RESET LOG_BUFFER;
```

Se pueden modificar parámetros de la base de datos en un PDB?

Con una sola instancia, se tiene solo un archivo de parámetros PFILE o SPFILE.

Con esta afirmación es común pensar que todos los parámetros se definen por igual para todos los contenedores. Sin embargo existen algunos parámetros que pueden ser configurados de manera independiente en cada contenedor. Los parámetros que pueden modificarse pueden ser revisados de la vista **V$PARAMETER** donde el campo **ISPDB_MODIFICABLE** tiene el valor de **TRUE**.

```
SQL> desc V$PARAMETER
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ----------------
 NUM                                                            NUMBER
 NAME                                                           VARCHAR2(80)
 TYPE                                                           NUMBER
 VALUE                                                          VARCHAR2(4000)
 DISPLAY_VALUE                                                  VARCHAR2(4000)
 ISDEFAULT                                                      VARCHAR2(9)
 ISSES_MODIFIABLE                                               VARCHAR2(5)
 ISSYS_MODIFIABLE                                               VARCHAR2(9)
 ISPDB_MODIFIABLE                                               VARCHAR2(5)
 ISINSTANCE_MODIFIABLE                                          VARCHAR2(5)
 ISMODIFIED                                                     VARCHAR2(10)
 ISADJUSTED                                                     VARCHAR2(5)
 ISDEPRECATED                                                   VARCHAR2(5)
 ISBASIC                                                        VARCHAR2(5)
 DESCRIPTION                                                    VARCHAR2(255)
 UPDATE_COMMENT                                                 VARCHAR2(255)
 HASH                                                           NUMBER
 CON_ID                                                         NUMBER
```

Vamos a modificar el parámetro OPTIMIZER_MODE en una de las bases de datos PDB

```
SQL> connect sys@PDB2 as sysdba
Enter password:
Connected.
SQL> alter system set optimizer_mode=FIRST_ROWS scope=spfile;

System altered.
```

Cuando coloco scope=spfile le estoy indicando a la base de datos que active el cambio cuando reinicie el PDB.

```
SQL> show parameter optimizer_mode

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
optimizer_mode                       string      ALL_ROWS
SQL>
SQL> alter pluggable database close;

Pluggable database altered.

SQL> alter pluggable database open;

Pluggable database altered.

SQL> show parameter optimizer_mode

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
optimizer_mode                       string      FIRST_ROWS
```

En el contendor ROOT podemos ver los valores que se le asigna a un parámetro en los diferentes contendores.

```
SQL> select CON_ID, NAME, ISPDB_MODIFIABLE, VALUE
  2   from V$SYSTEM_PARAMETER
  3  where  NAME='optimizer_mode';

    CON_ID NAME                 ISPDB VALUE
---------- -------------------- ----- -------------
         0 optimizer_mode       TRUE  ALL_ROWS
         4 optimizer_mode       TRUE  FIRST_ROWS
```

Aunque le decimos scope=spfile los valores de estos parámetros no se almacenan en el archivo de parámetros PFILE o SPFLE. Se almacenan en el diccionario de datos de cada PDB.

El fichero PFILE

En la siguiente tabla se muestran los parámetros más interesantes que se encuentran en el fichero PFILE

Parámetro	Descripción
COMPATIBLE	Contiene la versión de Oracle compatible con la versión instalada.
CONTROL_FILES	Se usa para indicar el nombre y la ubicación de los ficheros de control.
DB_BLOCK_SIZE	Número de bytes del tamaño del bloque de la caché de datos y de los ficheros de datos.
DB_DOMAIN	Nombre del dominio en el que se encuentra el servidor. Será útil en sistemas que usan un controlador de dominios.
DB_NAME	Nombre de la base de datos.
MEMORY_MAX_TARGET	El límite máximo de memoria que la instancia puede usar.
MEMORY_TARGET	Memoria asignada a la instancia.
PROCESSES	Contiene el número de procesos que se pueden conectar a la instancia a la vez.
SESSIONS	Se usa para indicar el número de sesiones que una instancia puede abrir.
STATISTICS_LEVEL	Para todas las funciones automáticas de Oracle, se puede configurar el nivel de información que se puede obtener. Estos valores pueden ser ALL, BASIC y TYPICAL.

PRÁCTICA

PARÁMETROS DE CONFIGURACIÓN GENERALES

Abre el fichero init.ora y analiza cada uno de los parámetros que se indican en la Tabla anterior, anotando el valor actual. A continuación, haz una copia de seguridad de ese fichero con la notación init.anioMesDia.back. Modifica los parámetros que correspondan con el fin de cumplir los requisitos siguientes:

a) El nombre de la base de datos es XE.

b) El nombre y la ubicación de los ficheros de control son c:\oradata\oradata\xe\ control01 .ctl y c:\oradata\oradata\xe\control02.ctl.

c) El número de procesos que se pueden conectar a la vez a la instancia es 50.

d) El nivel de estadisticas para las funcionalidades automáticas es máximo.

e) Se le asignan a la instancia 4GB

f) El tamaño del bloque es de 8KB

El fichero SPFILE

El fichero SPFlLE es un fichero binario, por lo que no es editable, del mismo modo que se hace con un fichero plano como PFILE. Para crear este fichero a partir del fichero de texto PFILE se usa el comando CREATE SPFILE con la sintaxis que a continuación se detalla:

CREATE SPFlLE [='nombreFiCheroSPFILE'] FROM PFILE [='nombreFiCheroPFILE‘];

Indicar el nombre del fichero, tanto para SPFlLE como para PFILE, es opcional. Cuando no se indican, se utilizan un nombre y una ubicación predeterminados. A continuación se ejemplifica el uso de este comando para leer el fichero c:\oradata\files\init.ora y crear el fichero C:\oradata\fi|es\spfi|e:

```
CREATE SPFILE ='c:\oradata\files\spfile.ora' FROM PFILE='c:\oradata\files\init.ora';
```

Del mismo modo que se puede obtener un fichero SPFlLE usando un fichero plano PFILE, también se puede exportar el contenido de un fichero de parámetros de inicialización de servidor a un fichero PFILE. El comando que hay que usar es CREATE PFILE y tiene el mismo principio que el comando anterior. Su sintaxis es la siguiente:

CREATE PFILE [='nombreFicheroPFILE'] FROM SPFlLE [='nombreFicheroSPFILE'];

El ejemplo siguiente consigue un fichero PFILE desde el fichero SPFlLE y lo almacena en c:\oradata\fi|es:

```
CREATE PFILE ='c:\oradata\Files\init.ora' FROM SPFILE;
```

Es una buena práctica hacer siempre un “backup” del SPFILE en un PFILE que tendremos a buen recaudo.

Estos comandos también se pueden usar desde los valores de la instancia actual. Para ello, se usa FROM MEMORY en vez del nombre de un fichero. Para sustituir algunos de estos ficheros, la instancia debe estar parada o iniciada, pero la base de datos debe estar cerrada. Para obtener información sobre el nombre y la ubicación de estos ficheros, se puede usar la consulta siguiente:

```
SELECT NAME,VALUE,DISPLAY_VALUE FROM V$PARAMETER
WHERE NAME IN ('spfile‘,'pfile');
```

Adviértase que la vista V$PARAMETER contiene el nombre de los parámetros en minúsculas.

Consulta de los valores de los parámetros, instancia y sesiones

En este apartado se analizan las diferentes vistas del diccionario de datos que permiten obtener información importante sobre los parámetros de inicialización, sobre las instancias, sobre las sesiones y sobre la base de datos.

info sobre parámetros

V$PARAMETER y V$SYSTEM_PARAMETER: información de los parámetros de la instancia actual. En concreto el primero muestra los valores modificados en la sesión actual, el segundo muestra los valores con los que la instancia arrancó. No se tiene en cuenta si han sido modificados en la sesión actual. V$SPPARAMETER contiene los parámetros definidos en el fichero SPFILE. Las columnas más interesantes son NAME, VALUE y DISPLAY_VALUE, que muestran el nombre del parámetro, su valor y su valor más legible, respectivamente.

V$PARAMETER			
NUM	NUMBER	ISINSTANCE_MODIFIABLE	VARCHAR2(5)
NAME	VARCHAR2(80)	ISMODIFIED	VARCHAR2(10)
TYPE	NUMBER	ISADJUSTED	VARCHAR2(5)
VALUE	VARCHAR2(4000)	ISDEPRECATED	VARCHAR2(5)
DISPLAY_VALUE	VARCHAR2(4000)	ISBASIC	VARCHAR2(5)
DEFAULT_VALUE	VARCHAR2(255)	DESCRIPTION	VARCHAR2(255)
ISDEFAULT	VARCHAR2(9)	UPDATE_COMMENT	VARCHAR2(255)
ISSES_MODIFIABLE	VARCHAR2(5)	HASH	NUMBER
ISSYS_MODIFIABLE	VARCHAR2(9)	CON_ID	NUMBER
ISPDB_MODIFIABLE	VARCHAR2(5)		

La siguiente consulta muestra el valor más legible de todos los parámetros que empiezan por SGA:

```
SELECT NAME, DISPLAY_VALUE FROM VSPARAMETER WHERE NAME LIKE 'SGA%';
```

Sin embargo, la siguiente consulta lo hace de los valores con los que arrancó la instancia:

```
SELECT NAME, DISPLAY_VALUE FROM V$SYSTEM_PARAMETER WHERE NAME LIKE 'SGA%';
```

Otras columnas interesantes son ISDEFAULT, que indica si su valor es igual al predeterminado; ISSES_MODIFIABLE, que indica si es modificable en una sesión; y ISMODIFIED, que indica si se ha modificado desde el inicio de la instancia.

La siguiente consulta muestra el nombre, el valor legible, si es modificable en una sesión y si ha sido modificado desde que la instancia se inició de todos los parámetros de inicialización que comienzan por LOG:

```
SELECT NAME, DISPLAY_VALUE, ISSES_MODIFIABLE, ISMODIFIED FROM V$PARAMETER WHERE NAME LIKE 'LOG%';
```

El comando **SHOW PARAMETERS** permite obtener información de los parámetros. Para mostrar los parámetros de interés basta con anexar el texto contenido en el nombre del parámetro. Asi, el comando siguiente muestra los valores de los parámetros que contienen en su nombre el texto max:

```
SHOW PARAMETERS max
```

PRÁCTICA

MOSTRAR INFORMACIÓN SOBRE LOS PARÁMETROS

Indica el comando o comandos que debes ejecutar para obtener la información siguiente:
a) El valor de los parámetros que contienen el texto SIZE.
b) La cabecera de la vista V$SYSTEM_PARAMETER2. Atendiendo a esta información, mostrar de los parámetros que empiezan por db si el valor es el mismo que el predeterminado.
c) La cabecera de la vista VSPARAMETER2. Atendiendo a esta información, mostrar, de los parámetros que tienen el texto pool, si el parámetro no se usa.

info sobre la instancia

V$INSTANCE: información sobre la instancia actual. Para obtener información de la instancia se puede consultarla vista dinámica VSINSTANCE. Esta vista está disponible en cualquiera de los estados de una base de datos; es decir, iniciada (NOMOUNT), montada (MOUNT) y abierta (OPEN). Generalmente, es una vista que el administrador usa cuando la base de datos está iniciada con el fin de evaluar el estado de la instancia, pero puede usarse en cualquier momento. Las columnas de interés son INSTANCE_NAME, STATUS, STARTUP_TIME, LOGINS e INSTANCE_MODE.

V$INSTANCE			
INSTANCE_NUMBER	NUMBER	LOGINS	VARCHAR2(10)
INSTANCE_NAME	VARCHAR2(16)	SHUTDOWN_PENDING	VARCHAR2(3)
HOST_NAME	VARCHAR2(64)	DATABASE_STATUS	VARCHAR2(17)
VERSION	VARCHAR2(17)	INSTANCE_ROLE	VARCHAR2(18)
VERSION_LEGACY	VARCHAR2(17)	ACTIVE_STATE	VARCHAR2(9)
VERSION_FULL	VARCHAR2(17)	BLOCKED	VARCHAR2(3)
STARTUP_TIME	DATE	CON_ID	NUMBER
STATUS	VARCHAR2(12)	INSTANCE_MODE	VARCHAR2(11)
PARALLEL	VARCHAR2(3)	EDITION	VARCHAR2(7)
THREAD#	NUMBER	FAMILY	VARCHAR2(80)
ARCHIVER	VARCHAR2(7)	DATABASE_TYPE	VARCHAR2(15)
LOG_SWITCH_WAIT	VARCHAR2(15)		

La siguiente consulta muestra el nombre de la instancia, su estado, hora en la que se arrancó, modo restringido para administradores y el modo de la instancia:

```
SELECT INSTANCE_NAME, STATUS, STARTUP_TIME, LOGINS, INSTANCE_MODE
FROM V$INSTANCE;
```

PRÁCTICA

MOSTRAR INFORMACIÓN SOBRE LA INSTANCIA

La siguiente consulta muestra información muy interesante sobre la instancia:
SELECT INSTANCE_NAME, STARTUP_TIME, STATUS, ARCHIVER, LOG_SWITCH_WAIT, DATABASE_STATUS, LOGINS, INSTANCE_MODE FROM V$INSTANCE;
Indica el valor de cada columna y deduce que información muestra la consulta.

INSTANCE_NUMBER: identificador numérico de la instancia.

INSTANCE_NAME: nombre de la instancia.

HOST_NAME: nombre host de la máquina en la que se lleva a cabo la instancia.

VERSION: versión de la base de datos.

STARTUP_TIME: hora en la que se inició la instancia.

STATUS: muestra el estado de la instancia, abierto o cerrado: STARTED, MOUNTED, OPEN u OPEN MIGRATE para opciones de actualización y de desactualización (ALTER DATABASE UPGRADEIDOWNGRADE).

PARALLEL: si la instancia se ejecuta en paralelo con otras bases de datos, usa Clusters.

THREAD#: número del hilo abierto de redo por la instancia.

ARCHIVER: estado del archivado automático (STOPPED, STARTED o FAILED).

LOG_SWITCH_WAIT: evento ocurrido en una basculación (ARCHIVE LOG, CLEAR LOG, CHECKPOINT).

LOGINS: indica si está en modo ALLOWED, o si solo los administradores RESTRICTED.

SHUTDOWN_PENDING: indica si está en espera de una parada de la base de datos.

DATABASE_STATUS: indica el estado (ACTIVE, SUSPEND, INSTANCE RECOVERY).

BLOCKED: indica si todos los servicios están bloqueados.

INSTANCE_MODE: muestra el modo de la instancia actual : REGULAR si usa o no RAC; READ MOSTLY para RAC con pocas escrituras; o READ ONLY para RAC de solo lectura.

DATABASE_TYPE: RAC si usa múltiples instancias, RACONENODE para RAC de un solo nodo, SINGLE si es una instancia normal, o UNKNOWN.

Info sobre la sesión actual

Una conexión a una base de datos tiene una sesión, o más de una sesión. La sesión hace referencia a la comunicación cuando un usuario se autentica y obtiene recursos para esa comunicación. Los datos de una sesión se almacenan en el servidor.

Para obtener información de las sesiones actuales se puede consultarla vista **V$SESSION.** Esta vista es muy completa y casi todas sus columnas se muestran en la Figura.

V$SESSION					
SADDR	RAW(8)	SQL_ID	VARCHAR2(13)	CURRENT_QUEUE_DURATION	NUMBER
SID	NUMBER	SQL_CHILD_NUMBER	NUMBER	CLIENT_IDENTIFIER	VARCHAR2(64)
SERIAL#	NUMBER	SQL_EXEC_START	DATE	SEQ#	NUMBER
AUDSID	NUMBER	SQL_EXEC_ID	NUMBER	EVENT#	NUMBER
PADDR	RAW(8)	PREV_SQL_ADDR	RAW(8)	EVENT	VARCHAR2(64)
USER#	NUMBER	PREV_HASH_VALUE	NUMBER	SECONDS_IN_WAIT	NUMBER
USERNAME	VARCHAR2(128)	PREV_SQL_ID	VARCHAR2(13)	STATE	VARCHAR2(19)
COMMAND	NUMBER	PREV_CHILD_NUMBER	NUMBER	WAIT_TIME_MICRO	NUMBER
OWNERID	NUMBER	PREV_EXEC_START	DATE	TIME_REMAINING_MICRO	NUMBER
TADDR	VARCHAR2(16)	PREV_EXEC_ID	NUMBER	TIME_SINCE_LAST_WAIT_MICRO	NUMBER
LOCKWAIT	VARCHAR2(16)	PLSQL_ENTRY_OBJECT_ID	NUMBER	SERVICE_NAME	VARCHAR2(64)
STATUS	VARCHAR2(8)	PLSQL_ENTRY_SUBPROGRAM_ID	NUMBER	SQL_TRACE	VARCHAR2(8)
SERVER	VARCHAR2(9)	PLSQL_OBJECT_ID	NUMBER	SQL_TRACE_WAITS	VARCHAR2(5)
SCHEMA#	NUMBER	PLSQL_SUBPROGRAM_ID	NUMBER	SQL_TRACE_BINDS	VARCHAR2(5)
SCHEMANAME	VARCHAR2(128)	MODULE	VARCHAR2(64)	SQL_TRACE_PLAN_STATS	VARCHAR2(10)
OSUSER	VARCHAR2(128)	MODULE_HASH	NUMBER	SESSION_EDITION_ID	NUMBER
PROCESS	VARCHAR2(24)	ACTION	VARCHAR2(64)	CREATOR_ADDR	RAW(8)
MACHINE	VARCHAR2(64)	ACTION_HASH	NUMBER	CREATOR_SERIAL#	NUMBER
PORT	NUMBER	CLIENT_INFO	VARCHAR2(64)	SQL_TRANSLATION_PROFILE_ID	NUMBER
TERMINAL	VARCHAR2(16)	FIXED_TABLE_SEQUENCE	NUMBER	PGA_TUNABLE_MEM	NUMBER
PROGRAM	VARCHAR2(64)	TOP_LEVEL_CALL#	NUMBER	SHARD_DDL_STATUS	VARCHAR2(8)
TYPE	VARCHAR2(10)	LOGON_TIME	DATE	CON_ID	NUMBER
SQL_ADDRESS	RAW(8)	LAST_CALL_ET	NUMBER	EXTERNAL_NAME	VARCHAR2(1024)
SQL_HASH_VALUE	NUMBER	PDML_ENABLED	VARCHAR2(3)	PLSQL_DEBUGGER_CONNECTED	VARCHAR2(5)

La siguiente consulta muestra el identificador y el número de serie de cada una de las sesiones:

```
SELECT SID, SERIAL#, USERNAME FROM VSSESSION;
```

y luego podemos matar una sesión de usuario concreto

```
ALTER SYSTEM KILL SESSION '7,5';
```

siendo 7 e 5 los valores de sid y serial# que queremos matar.

Info sobre la base de datos

Para obtener información importante de la base de datos se puede consultar la vista dinámica **V$DATABASE**. Esta vista está disponible solo si la base de datos está abierta.

Las columnas más interesantes son:

DBID: identificador único de la base de datos.

NAME: nombre de la base de datos.

CREATED: fecha de creación de la base de datos.

RESETLOGS_CHANGE#: número de cambios en los reseteos de los registros logs.

RESET_TIME: fecha de apertura de los resetlogs.

LOG_MODE: modo en el que se generan los registros logs: NOARCHIVELOG, ARCHIVELOG o MANUAL

CHECKPOINT_CHANGE# : numero SCN del último checkpoint.

CONTROLFILE_TYPE: Tipo de fichero de control: STANDBY para modo estándar, CLONE para bases de datos clonadas, BUCKUP para ficheros de control copiados o CREATED para ficheros creados, y CURRENT para el uso general

CONTROLFILE_CREATED: fecha de creación del fichero de control.

CONTROLFILE_TIME: fecha de la última copia del fichero de control.

OPEN_MODE: el modo en el que se abre la base de datos (MOUNTED, READ WrlTE, READ ONLY, READ ONLY WITH APPLY).

PROTECION_MODE: modo de protección (MAXIMUM PROTECTION, MAXIMUM AVAILABILITY, RESYNCHRIZATION, MAXIMUM PERFORMANCE, UNPROTECTED).

DB_UNIOUE_NAME: nombre único de la base de datos.

CDB: si la base de datos es un contener CDB.

V$DATABASE

DBID	NUMBER	REMOTE_ARCHIVE	VARCHAR2(8)	SUPPLEMENTAL_LOG_DATA_ALL.	VARCHAR2(3)
NAME	VARCHAR2(9)	ACTIVATION#	NUMBER	DB_UNIQUE_NAME	VARCHAR2(30)
CREATED	DATE	SWITCHOVER#	NUMBER	STANDBY_BECAME_PRIMARY_SCN.	NUMBER
RESETLOGS_CHANGE#	NUMBER	DATABASE_ROLE	VARCHAR2(16)	FS_FAILOVER_STATUS	VARCHAR2(22)
RESETLOGS_TIME	DATE	ARCHIVELOG_CHANGE#	NUMBER	FS_FAILOVER_CURRENT_TARGET.	VARCHAR2(30)
PRIOR_RESETLOGS_CHANGE#	NUMBER	ARCHIVELOG_COMPRESSION	VARCHAR2(8)	FS_FAILOVER_THRESHOLD	NUMBER
PRIOR_RESETLOGS_TIME	DATE	SWITCHOVER_STATUS	VARCHAR2(20)	FS_FAILOVER_OBSERVER_PRESENT	VARCHAR2(7)
LOG_MODE	VARCHAR2(12)	DATAGUARD_BROKER	VARCHAR2(8)	FS_FAILOVER_OBSERVER_HOST.	VARCHAR2(512)
CHECKPOINT_CHANGE#	NUMBER	GUARD_STATUS	VARCHAR2(7)	CONTROLFILE_CONVERTED	VARCHAR2(3)
ARCHIVE_CHANGE#	NUMBER	SUPPLEMENTAL_LOG_DATA_MIN.	VARCHAR2(8)	PRIMARY_DB_UNIQUE_NAME.	VARCHAR2(30)
CONTROLFILE_TYPE	VARCHAR2(7)	SUPPLEMENTAL_LOG_DATA_PK.	VARCHAR2(3)	SUPPLEMENTAL_LOG_DATA_PL.	VARCHAR2(3)
CONTROLFILE_CREATED	DATE	SUPPLEMENTAL_LOG_DATA_UI.	VARCHAR2(3)	MIN_REQUIRED_CAPTURE_CHANGE#	NUMBER
CONTROLFILE_SEQUENCE#	NUMBER	FORCE_LOGGING	VARCHAR2(39)	CDB	VARCHAR2(3)
CONTROLFILE_CHANGE#	NUMBER	PLATFORM_ID	NUMBER	CON_ID	NUMBER
CONTROLFILE_TIME	DATE	PLATFORM_NAME	VARCHAR2(101)	PENDING_ROLE_CHANGE_TASKS	VARCHAR2(512)
OPEN_RESETLOGS	VARCHAR2(11)	RECOVERY_TARGET_INCARNATION#.	NUMBER	CON_DBID	NUMBER
VERSION_TIME	DATE	LAST_OPEN_INCARNATION#	NUMBER	FORCE_FULL_DB_CACHING	VARCHAR2(3)
OPEN_MODE	VARCHAR2(20)	CURRENT_SCN	NUMBER		
PROTECTION_MODE	VARCHAR2(20)	FLASHBACK_ON	VARCHAR2(18)		
PROTECTION_LEVEL	VARCHAR2(20)	SUPPLEMENTAL_LOG_DATA_FK.	VARCHAR2(3)		

La siguiente consulta muestra el tipo y modo en el que se han abierto las bases de datos:

```
SELECT NAME,CDB,CON_ID,OPEN_MODE FROM V$DATABASE;
```

La siguiente consulta muestra la fecha en la que se crearon los ficheros de control:

```
SELECT CONTROLFILE_CREATED FROM V$DATABASE;
```

La siguiente consulta muestra el modo en el que se archivan los ficheros de actualización:

```
SELECT LOG_MODE FROM V$DATABASE;
```

La siguiente consulta muestra el modo y el nivel de protección de la base de datos:

```
SELECT PROTECTION_MODE, PROTECTION_LEVEL FROM V$DATABASE;
```

PRÁCTICA

ACTIVIDAD FINAL RESUMEN

Indica cuál seria el comando para llevar a cabo cada acción:
a) Iniciar la base de datos en modo instancia.
b) Dimensionar en fichero la memoria caché de datos para un tamaño de bloque de 16K de 1,5 gigabytes.
c) Usando una vista, mostrar el identificador único de la base de datos.
d) Abrir la base de datos XE solo para los usuarios de sesión restringida.
e) Mostrar el tipo de los ficheros de control.
f) Mostrar, usando una vista, si la base de datos usa múltiples instancias.
g) Desactivar de forma dinámica el modo de la base de datos a abierto restringido.
h) Mostrar el modo en el que se generan los registros logs.
i) Parar la base de datos en modo normal.
j) Para la instancia actual y usando el comando ALTER SYSTEM, obligar a que se use
k) el fichero de control c:\oradata\ctlprueba.ctl.
l) Mostrar aquellos parámetros cuyo valor es diferente al valor predeterminado.
m) Mostrar la fecha de la última copia de los ficheros de control.
n) Declarar en fichero el tamaño máximo de la memoria compartida en 1G.
o) Dimensionar en fichero el parámetro del tamaño del pool compartido a 64 megabytes.
p) Mostrar si la base de datos es un contenedor CDB.
q) Desactivar en fichero la configuración automática de la memoria compartida.
r) Mostrar los ficheros de traza de los procesos que tienen un uso de la CPU mayor que la media.
s) Cambiar el nombre de la base de datos a 'PRUEBA' con el comando ALTER SESSION.
t) Asignar en fichero un tamaño del DataBuffer de 256 megabytes.
u) Modificar usando ALTER SESSION, solo para la instancia actual, el parámetro DB__CACHE_SIZE a 128M.
v) Mostrar el estado de la instancia.
w)Limitar en fichero el tamaño máximo para la zona Result Cache a 1 megabyte.
x) Mostrar el nombre de la base de datos con el comando SHOW.

La arquitectura multitenant

A partir de Oracle Database 21c, una base de datos de contenedor multitenant es la única arquitectura admitida. En versiones anteriores, Oracle admitía bases de datos que no eran contenedores (no CDB).

Un contenedor es una colección de esquemas, objetos y estructuras relacionadas en una base de datos de contenedores multitenant (**CDB**) . Dentro de una CDB, cada contenedor tiene un ID y un nombre únicos.

La **root-CDB** , también llamada simplemente raíz , es una colección de esquemas a los que pertenecen todos los PDB. La raíz almacena los metadatos del sistema necesarios para administrar las PDB. Todos los PDB pertenecen a la raíz. La raíz de CDB no almacena datos de usuario. Oracle recomienda no agregar objetos comunes a la raíz ni modificar esquemas proporcionados por Oracle en la raíz. Sin embargo, puedes crear usuarios y roles comunes para la administración de la base de datos. Un usuario común con los privilegios necesarios puede cambiar entre contenedores.

La vista **V$CONTAINERS** nos da información sobre todos los contenedores

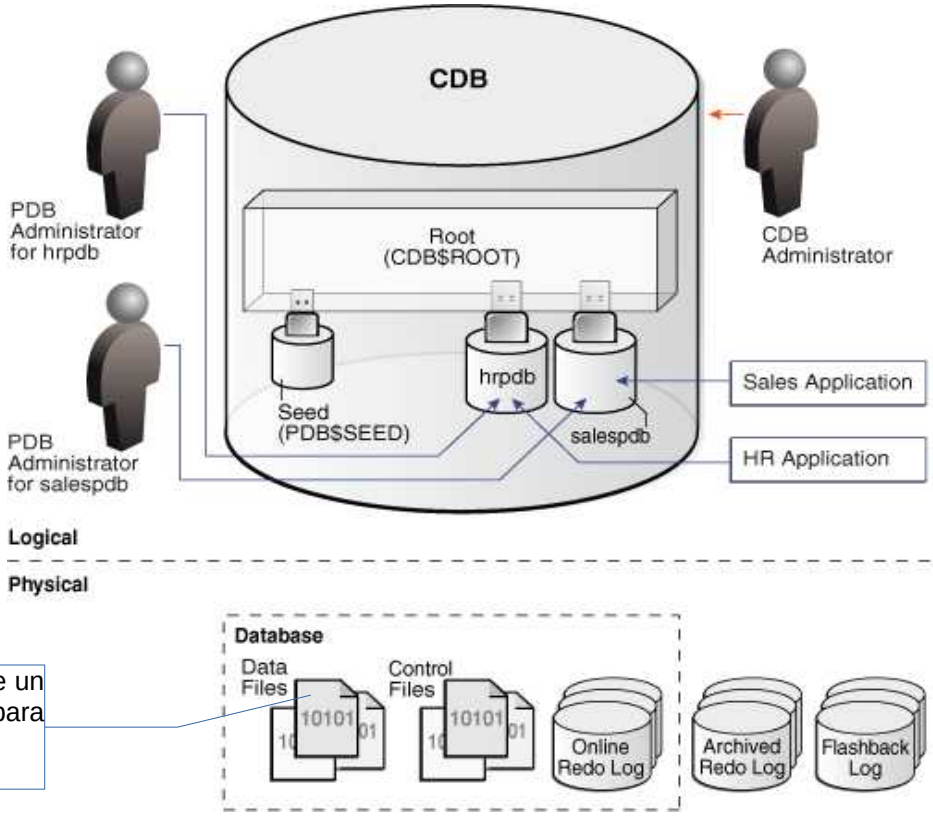
```
COL NAME FORMAT A15
SELECT NAME, CON_ID FROM   V$CONTAINERS ORDER BY CON_ID;
```

NAME	CON_ID
CDB$ROOT	1
PDB$SEED	2
SAAS_SALES_AC	3
SALESPDB	4
HRPDB	5

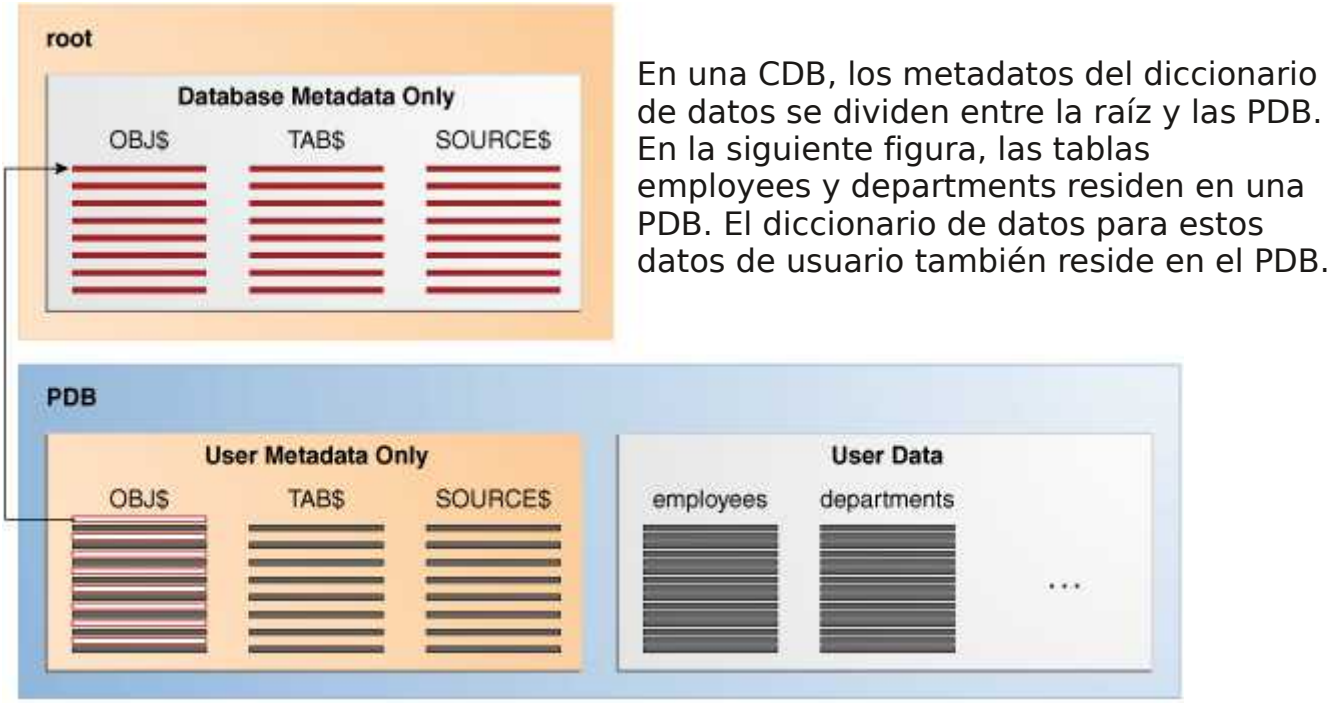
Una **PDB** es un conjunto de esquemas, objetos y estructuras relacionadas creado por el usuario que aparece lógicamente ante una aplicación cliente como una base de datos independiente.
Cada PDB es propiedad de SYS independientemente de qué usuario creó la PDB. SYS es un usuario común en la CDB lo que significa que este usuario tiene la misma identidad en la raíz y en cada PDB existente y futura dentro de la CDB.
Para crear PDB’s hay que tener el privilegio **CREATE PLUGGABLE DATABASE**
Se usan PDB's para

- Almacenar datos de us aplicación en una PDB dedicada
- Mover datos a un CDB diferente
- Migrar rápido. Desconectando una PDB y conectándola en un Oracle superior

Una CDB incluye cero, una o muchas bases de datos conectables (PDB) y contenedores de aplicaciones creados por el cliente. Una PDB es una colección portátil de esquemas, objetos de esquema y objetos que no son de esquema que aparece ante un cliente Oracle Net como una base de datos independiente. .

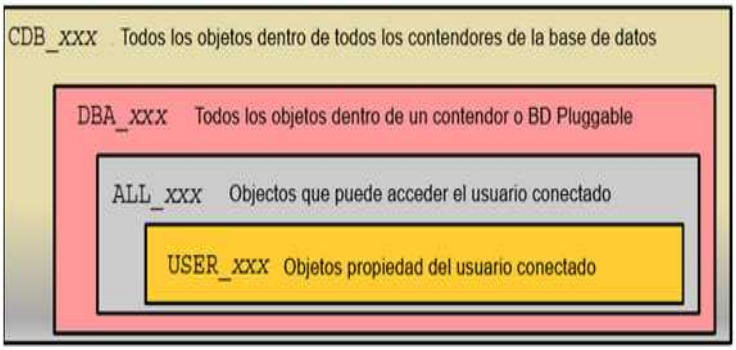


A nivel físico, esta CDB contiene un conjunto de archivos de datos para cada PDB y para la propia CDB.



En una CDB, los metadatos del diccionario de datos se dividen entre la raíz y las PDB. En la siguiente figura, las tablas employees y departments residen en una PDB. El diccionario de datos para estos datos de usuario también reside en el PDB.

Vistas del catálogo en multitenant



Todos estos objetos tienen una columna CON_ID cuyos valores pueden ser:
0 para todo el CDB
1 para CDB$ROOT
2 PDB$SEED
Otros números para PDB's

Cuando el contenedor actual es un PDB, un usuario puede ver información del diccionario de datos solo para el PDB actual. Para una aplicación conectada a una PDB, el diccionario de datos aparece como lo haría para una que no sea CDB. Sin embargo, cuando el contenedor actual es la raíz, un usuario común puede consultar las vistas CDB_ para ver los metadatos de la raíz y de las PDB para las que este usuario tiene privilegios. Si estamos conectados a una pdb, las vistas CDB_ son iguales a las DBA_

Todas las vistas DBA_XXX que conocemos tienen su vista CDB_XXX. En el siguiente cuadro muestro el homónimo (a nivel Multitenant) de algunas vistas conocidas.

Vista de Base de datos	Vista del contendor Multitenant
DATABASE_PROPERTIES	CDB_PROPERTIES
DBA_USERS	CDB_USERS
DBA_TABLESPACES	CDB_TABLESPACES
DBA_DATA_FILES	CDB_DATA_FILES
DBA_ROLES	CDB_ROLES
DBA_OBJECTS	CDB_OBJECTS
DBA_TABLES	CDB_TABLES

```
SQL>  desc CDB_USERS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 USERNAME                                  NOT NULL VARCHAR2(128)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(4000)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(128)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(128)
 EXTERNAL_NAME                                      VARCHAR2(4000)
 PASSWORD_VERSIONS                                  VARCHAR2(12)
 EDITIONS_ENABLED                                   VARCHAR2(1)
 AUTHENTICATION_TYPE                                VARCHAR2(8)
 PROXY_ONLY_CONNECT                                 VARCHAR2(1)
 COMMON                                             VARCHAR2(3)
 LAST_LOGIN                                         TIMESTAMP(9) W
 ORACLE_MAINTAINED                                  VARCHAR2(1)
 CON_ID                                             NUMBER
```

En todas las vistas CDB_XXX encontraremos los mismos campos que en las vistas DBA_XXX más el campo CON_ID que indica a qué contenedor pertenece el elemento.

Cuando se consulta una vista CDB_XXX en el contenedor ROOT, la vista muestra la información de los elementos en todos los contenedores de la base de datos.

```
SQL> select CON_ID, FILE_ID, FILE_NAME, TABLESPACE_NAME
  2  from CDB_DATA_FILES
  3  order by CON_ID, FILE_ID;

    CON_ID    FILE_ID FILE_NAME                                                TABLESPACE_NAME
---------- ---------- ---------------------------------------------------------- ----------
         1          1 /u02/oradata/cdb1/system01.dbf                           SYSTEM
         1          3 /u02/oradata/cdb1/sysaux01.dbf                           SYSAUX
         1          5 /u02/oradata/cdb1/undotbs01.dbf                          UNDOTBS1
         1          6 /u02/oradata/cdb1/users01.dbf                            USERS
         2          2 /u02/oradata/cdb1/pdbseed/system01.dbf                   SYSTEM
         2          4 /u02/oradata/cdb1/pdbseed/sysaux01.dbf                   SYSAUX
         3          7 /u02/oradata/cdb1/pdb1/system01.dbf                      SYSTEM
         3          8 /u02/oradata/cdb1/pdb1/sysaux01.dbf                      SYSAUX
         3         11 /u02/oradata/cdb1/pdb1/data01.dbf                        DATA
         4          9 /u02/oradata/cdb1/PDB2/system01.dbf                      SYSTEM
         4         10 /u02/oradata/cdb1/PDB2/sysaux01.dbf                      SYSAUX
         6         25 /u02/oradata/ORCL/datafile/o1_mf_system_9rqk3bbf_.dbf    SYSTEM
         6         26 /u02/oradata/ORCL/datafile/o1_mf_sysaux_9rqk0nk1_.dbf    SYSAUX
         6         27 /u02/oradata/ORCL/datafile/o1_mf_users_9rqk5zxl_.dbf     USERS
         6         28 /u02/oradata/ORCL/datafile/o1_mf_example_9rqk8owg_.dbf   EXAMPLE
```

Cuando se consulta la misma vista CDB_XXX, conectado a algún contenedor en particular (por ejemplo un PDB) la vista muestra solo la información de dicho contendor.

```
SQL> connect sys@PDBORCL as sysdba
Enter password:
Connected.
SQL> select CON_ID, FILE_ID, FILE_NAME, TABLESPACE_NAME
  2  from CDB_DATA_FILES
  3  order by CON_ID, FILE_ID;

    CON_ID    FILE_ID FILE_NAME                                                TABLESPACE_NAME
---------- ---------- ---------------------------------------------------------- ----------
         6         25 /u02/oradata/ORCL/datafile/o1_mf_system_9rqk3bbf_.dbf    SYSTEM
         6         26 /u02/oradata/ORCL/datafile/o1_mf_sysaux_9rqk0nk1_.dbf    SYSAUX
         6         27 /u02/oradata/ORCL/datafile/o1_mf_users_9rqk5zxl_.dbf     USERS
         6         28 /u02/oradata/ORCL/datafile/o1_mf_example_9rqk8owg_.dbf   EXAMPLE
```

CDB_PDBS

Dentro del grupo de vistas CDB_XXX encontramos una nueva vista llamada CDB_PDBs que entrega más información sobre cada PDB en la base de datos.

```
SQL> select CON_ID, PDB_ID, PDB_NAME, STATUS
  2  from CDB_PDBS
  3  order by CON_ID;

    CON_ID     PDB_ID PDB_NAME                                 STATUS
---------- ---------- ---------------------------------------- --------
         1          2 PDB$SEED                                 NORMAL
         1          6 PDBORCL                                  NORMAL
         1          4 PDB2                                     NORMAL
         1          3 PDB1                                     NORMAL
```

Ejemplos:

SQL> SELECT COUNT(*) FROM CDB_USERS WHERE CON_ID=1; COUNT(*) -------- 41	El usuario system, que es común y está conectado al raíz consulta el número de usuarios comunes en el CDB
SQL> SELECT COUNT(DISTINCT(CON_ID)) FROM CDB_USERS; COUNT(DISTINCT(CON_ID)) ---------- 4	System consulta el número de distintos contenedores que hay el el CDB
SQL> CONNECT SYSTEM@hrdb Enter password: ******** Connected.	System se conecta ahora a una pdb llamada hrpdb
SQL> SELECT COUNT(*) FROM CDB_USERS; COUNT(*) ---------- 45	System consulta los usuarios cdb, no da el mismo valor porque al no estar conectada a la raíz, las vistas CDB_ son iguales que las vistas DBA_

Con la nueva arquitectura se han desarrollado nuevas Vistas de diccionario que nos permiten revisar el estado de los contendores, así como también, el estado de los elementos dentro de cada contenedor.
También encontramos modificaciones a vistas ya conocidas para poder soportar la nueva arquitectura.

V$DATABASE

Ya la hemos estudiado antes. Se le ha aumentado con nuevos campos para poder diferenciar entre una base de datos CDB y Non-CDB

```
CDB                            VARCHAR2(3)
CON_ID                         NUMBER
PENDING_ROLE_CHANGE_TASKS      VARCHAR2(512)
CON_DBID                       NUMBER
```

```
SQL> select NAME, CDB, CON_ID, PENDING_ROLE_CHANGE_TASKS, CON_DBID
  2  from v$database;

NAME      CDB     CON_ID PENDING_ROLE_CH   CON_DBID
--------- --- ---------- --------------- ----------
CDB1      YES          0 NOT APPLICABLE   821677457
```

V$PDBS

Tal vez la vista que más se utiliza en esta arquitectura. Esta vista muestra las características y el estado de los PDBs dentro de un CDB.

Es importante anotar la particularidad en el nombre de la vista. Es una vista V$, sin embargo, se define en plural.

```
SQL> desc V$PDBS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CON_ID                                             NUMBER
 DBID                                               NUMBER
 CON_UID                                            NUMBER
 GUID                                               RAW(16)
 NAME                                               VARCHAR2(30)
 OPEN_MODE                                          VARCHAR2(10)
 RESTRICTED                                         VARCHAR2(3)
 OPEN_TIME                                          TIMESTAMP(3)
 CREATE_SCN                                         NUMBER
 TOTAL_SIZE                                         NUMBER
```

Es importante tomar atención al campo CON_ID ya que es el código identificador de cada PDB. Es el campo por el que se relaciona cada elemento con su contendor.

```
SQL> select CON_ID, NAME, OPEN_MODE, TOTAL_SIZE
  2  from V$PDBS;

    CON_ID NAME                           OPEN_MODE  TOTAL_SIZE
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY   393216000
         3 PDB1                           READ WRITE  414187520
         4 PDB2                           READ WRITE  393216000
         6 PDBORCL                        READ WRITE 1053818880
```

Pueden darse cuenta que en la vista V$PDBS no muestra al contendor ROOT, es porque el contendor ROOT es considerado como el contendor CDB.

Para poder ver la información del contendor ROOT utilizamos la siguiente vista

V$CONTAINERS

Vista que nos muestra la información de todos los contendores en la base de datos. Es muy parecida a V$PDBS pero también contiene la información del contendor ROOT.

```
SQL> desc V$CONTAINERS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CON_ID                                             NUMBER
 DBID                                               NUMBER
 CON_UID                                            NUMBER
 GUID                                               RAW(16)
 NAME                                               VARCHAR2(30)
 OPEN_MODE                                          VARCHAR2(10)
 RESTRICTED                                         VARCHAR2(3)
 OPEN_TIME                                          TIMESTAMP(3)
 CREATE_SCN                                         NUMBER
 TOTAL_SIZE                                         NUMBER


SQL>  select CON_ID, NAME, OPEN_MODE, TOTAL_SIZE
  2  from V$CONTAINERS;

    CON_ID NAME                           OPEN_MODE  TOTAL_SIZE
---------- ------------------------------ ---------- ----------
         1 CDB$ROOT                       READ WRITE 1525678080
         2 PDB$SEED                       READ ONLY   393216000
         3 PDB1                           READ WRITE  414187520
         4 PDB2                           READ WRITE  393216000
         6 PDBORCL                        READ WRITE 1053818880
```

Como conectarme a una CDB y PDB

Conexión a la CDB

El CDB (container database) es administrado desde el contenedor ROOT y para poder conectarse podemos usar dos formas:

Conexión local

c:\> sqlplus / as sysdba

Conexión usando el Listener

```
C:\Users\manuel>lsnrctl status

LSNRCTL for 64-bit Windows: Version 21.0.0.0.0 - Production on 01-SEP-2023 10:23:46

Copyright (c) 1991, 2021, Oracle.  All rights reserved.

Conectßndose a (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=DESKTOP-SF84IHC)(PORT=1521)))
ESTADO del LISTENER
------------------------
Alias                     LISTENER
Versi¾n                   TNSLSNR for 64-bit Windows: Version 21.0.0.0.0 - Production
Fecha de Inicio       22-AGO-2023 20:50:12
Tiempo Actividad   9 dŸas 13 hr. 33 min. 35 seg.
Nivel de Rastreo        off
Seguridad               ON: Local OS Authentication
SNMP                      OFF
Servicio por Defecto           XE
Parßmetros del Listener   C:\app\manuel\product\21c\homes\OraDB21Home1\network\admin\listener.ora
Log del Listener          C:\app\manuel\product\21c\diag\tnslsnr\DESKTOP-SF84IHC\listener\alert\log.xml
Recibiendo Resumen de Puntos Finales...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=DESKTOP-SF84IHC)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROC1521ipc)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=127.0.0.1)(PORT=5500))(Security=(my_wallet_directory=C:\APP\MANUEL\PRODUCT\
21C\admin\XE\xdb_wallet))(Presentation=HTTP)(Session=RAW))
Resumen de Servicios...
El servicio "CLRExtProc" tiene 1 instancia(s).
  La instancia "CLRExtProc", con estado UNKNOWN, tiene 1 manejador(es) para este servicio...
El servicio "XE" tiene 1 instancia(s).
  La instancia "xe", con estado READY, tiene 2 manejador(es) para este servicio...
El servicio "XEXDB" tiene 1 instancia(s).
  La instancia "xe", con estado READY, tiene 1 manejador(es) para este servicio...
El servicio "a2ea3afa02d94c9694c15e29c71636ea" tiene 1 instancia(s).
  La instancia "xe", con estado READY, tiene 2 manejador(es) para este servicio...
El servicio "xepdb1" tiene 1 instancia(s).
  La instancia "xe", con estado READY, tiene 2 manejador(es) para este servicio...
El comando ha terminado correctamente
```


c:\>sqlpus sys@xe as sysdba

Hay que tener en cuenta que independientemente de cual usemos, si ejecutamos el sqlplus con el usuario del SO con el que instalamos Oracle, entraremos como sys aunque no sepamos la password.

Conexión a la PDB

Para conectarse a la PDB no podemos usar la conexión local. Siempre usamos el listener. Además hay dos formas: en remoto (poniendo toda la cadena de conexión) o usando el service name que debemos dar de alta en el tnsnames.ora. Para usar esto debemos tener el privilegio CREATE SESSION en este contenedor.

c:\>sqlplus sys@xepdb1 as sysdba

Otra forma de conectarse a los contenedores es mediante el parámetro de sesión CONTAINER Para poder utilizar este método es necesario estar conectado con un usuario COMUN para todos los contenedores (en nuestro caso sys o system)

ALTER SESSION SET CONTAINER=contenedor;

```
SQL> show con_name;

CON_NAME
------------------------------
CDB$ROOT
SQL> alter session set CONTAINER=PDB1;

Session altered.
```

Mira en qué contenedor estoy

Cambia contenedor a la pluggable database

Ejemplo:

`SQL> CONNECT SYSTEM@prod`
`Enter password: ********`
`Connected.`
	El usuario system, que es común se conecta al raíz usando el servicio llamado prod

`SQL> SHOW CON_NAME`
 	
`CON_NAME`
`--------`
`CDB$ROOT`
	Lista el nombre del contenedor al cual el usuario está actualmente conectado. CDB$ROOT es el nombre del contenedor raíz.

`SQL> ALTER SESSION`
`  SET CONTAINER = hrpdb;`
 	
`Session altered.`
	System se conecta ahora a una PDB llamada hrpdb