ORACLE: Arquitectura

Todo el mundo puede conducir un automóvil sin necesidad de conocer cómo funciona un motor de combustión interna y todos los subsistemas asociados a él. Pero entonces ciertos conceptos como aprovechamiento de la potencia, compresión, endurecimiento de la suspensión, motricidad, etc., le serán ajenos y nunca podrá sacar lo mejor del automóvil. Y si tiene algún problema se quedará tirado en la carretera.

De la misma manera, no podremos aspirar a que nuestras aplicaciones de BD funcionen bien si no conocemos la arquitectura del *motor* de la BD, el servidor. Es indispensable conocer los factores y parámetros que influyen en el funcionamiento de nuestro SGBD para poder solucionar los problemas que se pueden plantear en cuanto nos salgamos de las aplicaciones estándares y básicas de BD, o en cuanto tengamos algún problema.

El siguiente curso aborda la arquitectura del SGBD Oracle y da una visión lo suficientemente profunda del mismo como para que podamos entender cómo funciona.

Si tienes cualquier sugerencia o encuentras una errata escondida dímelo.

Abril de 1998.

Jesús Vegas
Dpto. Informática
Universidad de Valladolid
jvegas@infor.uva.es

Índice

1 Bases de Datos e Instancias

Estos son dos conceptos fundamentales para entender la arquitectura de Oracle.

En términos sencillos, una instancia de BD es un conjunto de procesos del servidor Oracle que tiene su propio área global de memoria y una base de datos asociada a ellos.

1.1 Base de Datos

Una Base de Datos Oracle es un conjunto de datos almacenado y accesible según el formato de tablas relacionales. Una tabla relacional tiene un nombre y unas columnas, su definición. Los datos están almacenados en las filas. Las tablas pueden estar relacionadas con otras.

Una Base de Datos Oracle está almacenada físicamente en ficheros, y la correspondencia entre los ficheros y las tablas es posible gracias a las estructuras internas de la BD, que permiten que diferentes tipos de datos estén almacenados físicamente separados. Está división lógica se hace gracias a los espacios de tablas, *tablespaces*.

1.1.1 Los Espacios de Tablas, *Tablespaces*

Un espacio de tablas es una división lógica de la BD. Cada BD tiene al menos uno (`SYSTEM`). Un espacio de tablas puede pertenecer sólo a una BD. Los espacios de tablas se utilizan para mantener juntos los datos de usuarios o de aplicaciones para facilitar su mantenimiento o mejorar las prestaciones del sistema.

De esta manera, cuando se crea una tabla se debe indicar el espacio de tablas al que se destina. Por defecto se depositan en el espacio de tablas `SYSTEM`, que se crea por defecto. Este espacio de tablas es el que contiene el diccionario de datos, por lo que conviene reservarlo para el uso del servidor, y asignar las tablas de usuario a otro.

Lo razonable y aconsejable es que cada aplicación tenga su propio espacio de tablas.

Hay varias razones que justifican este modo de organización de las tablas en espacios de tablas:

- Un espacio de tablas puede quedarse *offline* debido a un fallo de disco, permitiendo que el SGBD continúe funcionando con el resto.
- Los espacios de tablas pueden estar montados sobre dispositivos ópticos si son de sólo lectura.
- Permiten distribuir a nivel lógico/físico los distintos objetos de las aplicaciones.
- Son una unidad lógica de almacenamiento, pueden usarse para aislar completamente los datos de diferentes aplicaciones.
- Oracle permite realizar operaciones de *backup/recovery* a nivel de espacio de tabla mientras la BD sigue funcionando.

Cuando se crean se les asigna un espacio en disco que Oracle reserva inmediatamente, se utilice o no. Si este espación inicial se ha quedado pequeño Oracle puede gestionar el crecimiento dinámico de los ficheros sobre los que se asientan los espacios de tablas. Esto elimina la posibilidad de error en las aplicaciones por fallos de dimensionamiento inicial. Los parámetros de crecimiento del tamaño de los espacios de tablas se especifican en la creación de los mismos.

Se pueden ver los espacios de tablas definidos en nuestra BD con el comando SQL siguiente:

```
SQL>select* from user_tablespaces;
```

Dentro de cada espacio de tabla se pueden almacenar objetos de distinta naturaleza: tablas, índices, etc. Pero no se pueden mezclar si más. Necesitamos una manera de separarlos, y eso son los *segmentos*.

Se pueden almacenar más de un segmento por espacio de tabla. Un segmento está contenido en su totalidad en un espacio de tabla. Un segmento está constituido por un conjunto de extensiones, que no son más que grupos de bloques de disco ORACLE contiguos. Cuando se borra un segmento, el espacio es devuelto al espacio de tabla.

Todos los datos de la BD están almacenados en segmentos. Y existen 5 tipos de segmentos:

- de datos: almacenan las tablas.
- de índices: permiten un acceso rápido a los datos dependiendo de la cantidad de los mismos (árboles B). Las consultas que sólo referencian a columnas indexadas se resuelven en el índice. Establecen un control de unicidad (los índices son automáticos cuando se definen claves primarias). Cada índice ocupa un segmento independiente del segmento de datos y deberían estar en un espacio de tablas distinto al de los datos, para mejorar el rendimiento.
- de *rollback*: son objetos internos de la BD que permiten efectuar la restauración de las transacciones no validadas asegurando la consistencia en lectura. La estructura de los registros de *rollback* es :
  - Identificador de la transacción.
  - Dirección del bloque donde está la tabla.
  - Número de fila.
  - Número de columna.
  - Valor del dato antiguo (antes de ser modificado).

  Son tan importantes que una BD no puede arrancar si no puede acceder al menos a un segmento de *rollback*. Si la BD tiene múltiples espacios de tablas, deben existir al menos dos segmentos de *rollback* y cada segmento de *rollback* debe tener al menos dos extensiones, reutilizables de manera cíclica. Esto segmentos son un objeto compartido de la BD, aunque se puede asinar un segmento de *rollback* particular a una transacción dada.

- temporales: son creados por Oracle para un uso temporal cuando debe realizar una ordenación que no le cabe en memoria, y en las operaciones: `create index`, `order by`, `group by`, `distinct`, `union`, `intersect`, `minus`. Son eliminados cuando la sentencia finaliza.
- de *bootstrap*: Se crea en `SYSTEM` y contiene definiciones del diccionario para sus tablas, que se cargan al abrir la BD. No requiere ninguna acción por parte del DBA. No cambia de tamaño.

La tabla que guarda la información de los segmentos de usuario es `user_segments`, y se puede visualizar la información sobre los segmentos con la sentencia SQL siguiente:

```
SQL>select* from user_segments;
```

1.1.2 Ficheros

Cada espacio de tablas se compone de uno o más ficheros en disco. Un fichero puede pertenecer sólo a un espacio de tablas. Los ficheros reciben un tamaño fijo en el momento de su creación, y cuando se necesita más espacio se deben añadir más ficheros a espacio de tablas.

Dividir los objetos de la BD entre múltiples espacios de tablas permiten que los objetos sean almacenados físicamente en discos separados, dependiendo de donde estén los ficheros sobre los que se asientan.

1.2 Instancias

Para permitir el acceso a los datos, Oracle utiliza un conjunto de procesos que son compartidos por todos los usuarios. Además, existen estructuras de memoria que son utilizadas para almacenar los datos más recientemente solicitados a la BD.

Una *instancia* de BD es el conjunto de estructuras de memoria y de procesos que acceden a los ficheros de datos.

Los parámetros que determinan el tamaño y composición de una instancia están almacenados en un fichero llamado `init.ora`. Este fichero es leido durante el arranque de la BD y puede ser modificado por el DBA. Cualquier modificación de este fichero no tiene efecto hasta la siguiente vez que se arranque la BD.

Las estructuras de la BD Oracle pueden ser divididas en tres clases:

- aquellas que son internas a la BD,

- aquellas que son internas a las áreas de memoria (incluidas la memoria compartida y procesos),
- aquellas que son externas a la BD.

1.3 Estructuras Internas de la BD

Tablas y Columnas

Los datos son almacenados en la BD utilizando tablas. Cada tabla está compuesta por un número determinado de columnas.

Las tablas propiedad del usuario `SYS` son llamadas tablas del diccionario de datos. Proveen el catálogo del sistema que permite que la BD se gestione a sí misma.

Las tablas se pueden relacionar entre ellas a través de las columnas que las componen. La BD se puede utilizar para asegurar el cumplimiento de esas relaciones a través de la integridad referencial, que se concreta en las restricciones de tablas.

Restricciones de Tablas

Una tabla puede tener asociadas restricciones que deben cumplir todas las filas. Entre las restricciones que se pueden fijar algunas reciben nombres especiales.: *clave primaria, clave ajena.*

La clave primaria de una tabla está compuesta por las columnas que hacen a cada fila de la tabla una fila distinta.

La clave ajena se utiliza para especificar las relaciones entre tablas. De modo que un conjunto de columnas declaradas como clave ajena de una tabla deben tener valores tomados de la clave primaria de otra tabla.

Usuarios

Una cuenta de usuario no es una estructura física de la BD, pero está relacionada con los objetos de la BD: los usuarios poseen los objetos de la BD. Existen dos usuarios especiales: `SYS` y `SYSTEM`. El usuarios `SYS` posee las tablas del diccionario de datos; que almacenan información sobre el resto de las estructuras de la BD. El usuario `SYSTEM` posee las vistas que permiten acceder a las tablas del diccionario, para el uso del resto de los usuarios de la BD.

Todo objeto creado en la BD se crea por un usuario, en un espacio de tablas y en un fichero de datos determinado. Toda cuenta de la BD puede estár unida a una cuenta del S.O., lo que permite a los usuarios acceder a la cuenta de la BD sin dar la clave de acceso.

Cada usuario puede acceder a los objetos que posea o a aquellos sobre los que tenga derecho de acceso.

Esquemas

El conjunto de objetos de un usuario es conocido como esquema.

Índices

Un índice es una estructura de la BD utilizada para agilizar el acceso a una fila de una tabla. Cada fila tiene un identificador de fila, `ROWID`, que determina el fichero, bloque y fila dentro del bloque donde está almacenada la fila.

Cada entrada del índice consite en un valor clave y una `ROWID`. Cada una de estas entradas se almacena en un árbol B^+.

Los índices se crean automáticamente cuando se define una restricción `UNIQUE` o `PRIMARY KEY`.

Clusters

Las tablas que son accedidas juntas frecuentemente pueden ser almacenadas juntas. Para ello se crea un

cluster. De este modo se minimiza el número de E/S.

Las columnas que relacionan las tablas de un *cluster* se llaman clave del *cluster*.

Vistas

Conceptualmente, una vista puede considerarse como una máscara que se extiende sobre una o más tablas, de modo que cada columna de la vista se corresponde con una o más columnas de las tablas subyacentes. Cuando se consulta una vista, esta traspasa la consulta a las tablas sobre las que se asienta. Las vistas no se pueden indexar.

Las vistas no generan almacenamiento de datos, y sus definiciones se almacenan en el diccionario de datos.

Secuencias

Las definiciones de secuencias se almacenan en el diccionario de datos. Son mecanismos para obtener listas de números secuenciales.

Procedimientos y Funciones

Un procedimiento es un bloque de código PL/SQL, que se almacena en el diccionario de datos y que es llamado por las aplicaciones. Se pueden utilizar para implementar seguridad, no dando acceso directamente a determinadas tablas sino es a través de procedimientos que acceden a esas tablas. Cuando se ejecuta un procedimiento se ejecuta con los privilegios del propietario del procedimiento. La diferencia entre un procedimiento y una función es que ésta última puede devolver valores.

Paquetes, *Packages*

Se utilizan para agrupar procedimientos y funciones. Los elementos dentro de los paquetes pueden ser públicos o privados. Los públicos pueden ser llamados por los usuarios, los privados están ocultos a los usuarios y son llamados por otros procedimientos.

Disparadores, *Triggers*

Son procedimientos que son ejecutados cuando se procude un determinado evento en la BD. Se pueden utilizar para mejorar y reforzar la integridad y la seguridad de la BD.

Sinónimos

Para identificar completamente un objeto dentro de una BD se necesita especificar el nombre de la máquina, el nombre del servidor, el nombre del propietario y el nombre del objeto. Para hacer transparente todo esto al usuario se pueden utilizar los sinónimos. Éstos apuntarán a los objetos y si el objeto cambia de lugar o propietario, sólo habrá que modificar el sinónimo.

Existen sinónimos públicos y privados. Los públicos son conocidos por todos los usuarios de una BD. Los privados son locales a un usuario.

Privilegios y Roles

Para que un objeto pueda ser accedido por un usuario debe de tener otorgado ese privilegio. Ejemplos de privilegios son `INSERT`, `SELECT`, `UPDATE`, `EXECUTE`, etc.

Los roles son grupos de privilegios que pueden ser utilizados para facilitar la gestión de los privilegios. Los privilegios se pueden otorgar a un rol, y los roles pueden ser otorgados a múltiples usuarios.

Segmentos, Extensiones y Bloques

Los segmentos son los equivalentes físicos de los objetos que almacenan datos. El uso efectivo de los segmentos requiere que el DBA conozca los objetos que utiliza una aplicación, cómo los datos son introducidos en esos objetos y el modo en que serán recuperados.

Como los segmentos son entidades físicas, deben estar asignados a espacios de tablas en la BD y estarán

localizados en uno de los ficheros de datos del espacio de tablas. Un segmento está constituido por secciones llamadas extensiones, que son conjuntos contiguos de bloques Oracle. Una vez que una extensión existente en un segmento no puede almacenar más datos, el segmento obtendrá del espacio de tabla otra extensión. Este proceso de extensión continuará hasta que no quede más espacio disponible en los ficheros del espacio de tablas, o hasta que se alcance un número máximo de extensiónes por segmento.

Segmento de *Rollback*

Para mantener la consistencia en lectura y permitir deshacer las transacciones, Oracle debe tener un mecanismo para reconstruir la imágen previa a una transacción incompleta. Oracle utiliza los segmentos de *rollback* para esto.

Los segmentos de *rollback* pueden crecer tanto como sea necesario para soportar las transacciones.

1.4 Estructuras de Memoria Internas

Oracle mantiene dos estructuras principales de memoria: el Área Global de Programa, *Program Global Area, PGA*; y el Área Global del Sistema, *System Global Area* o también *Shared Global Area, SGA*.

El PGA es la zona de memoria de cada proceso Oracle. No está compartida y contiene datos e información de control de un único proceso.

El SGA es la zona de memoria en la que la BD Oracle guarda información sobre su estado. Esta estructura de memoria está disponible para todos los procesos, por eso se dice que está compartida.

1.4.1 Área Global del Sistema, SGA

Sirve para facilitar la transferencia de información entre usuarios y también almacena la información estructural de la BD más frecuentemente requerida.

La SGA se divide en varias partes:

Buffers* de BD, *Database Buffer Cache

Es el caché que almacena los bloques de datos leidos de los segmentos de datos de la BD, tales como tablas, índices y clusters. Los bloques modificados se llamas *bloques sucios*. El tamaño de buffer caché se fija por el parámetro `DB_BLOCK_BUFFERS` del fichero `init.ora`.

Como el tamaño del buffer suele ser pequeño para almacenar todos los bloques de datos leidos, su gestión se hace mediante el algoritmo LRU.

Buffer Redo Log

Los registros *Redo* describen los cámbios realizados en la BD y son escritos en los ficheros *redo log* para que puedan ser utilizados en las operaciones de recuperación hacia adelante, *roll-forward*, durante las recuperaciones de la BD. Pero antes de ser escritos en los ficheros *redo log* son escritos en un caché de la SGA llamado *redo log buffer*. El servidor escribe periódicamente los registros *redo log* en los ficheros *redo log*.

El tamaño del buffer redo log se fija por el parámetro `LOG_BUFFER`.

Área de SQL Compartido, *Shared SQL Pool*

En esta zona se encuentran las sentencias SQL que han sido analizadas. El analisis sintáctico de las sentencias SQL lleva su tiempo y Oracle mantiene las estructuras asociadas a cada sentencia SQL analizada durante el tiempo que pueda para ver si puede reutilizarlas. Antes de analizar una sentencia SQL, Oracle mira a ver si encuentra otra sentencia exactamente igual en la zona de SQL compartido. Si es así, no la analiza y pasa directamente a ejecutar la que mantinene en memoria. De esta manera se premia la uniformidad en la programación de las aplicaciones. La igualdad se entiende que es lexicografica, espacios

en blanco y variables incluidas. El contenido de la zona de SQL compartido es:

- Plan de ejecución de la sentencia SQL.
- Texto de la sentencia.
- Lista de objetos referenciados.

Los pasos de procesamiento de cada petición de análisis de una sentencia SQL son:

- Comprobar si la sentencia se encuentra en el área compartida.
- Comprobar si los objetos referenciados son los mismos.
- Comprobar si el usuario tiene acceso a los objetos referenciados.

Si no, la sentencia es nueva, se analiza y los datos de análisis se almacenan en la zona de SQL compartida.

También se almacena en la zona de SQL compartido el *caché del diccionario*. La información sobre los objetos de la BD se encuentra almacenada en las tablas del diccionario. Cuando esta información se necesita, se leen las tablas del diccionario y su información se guarda en el caché del diccionario de la SGA.

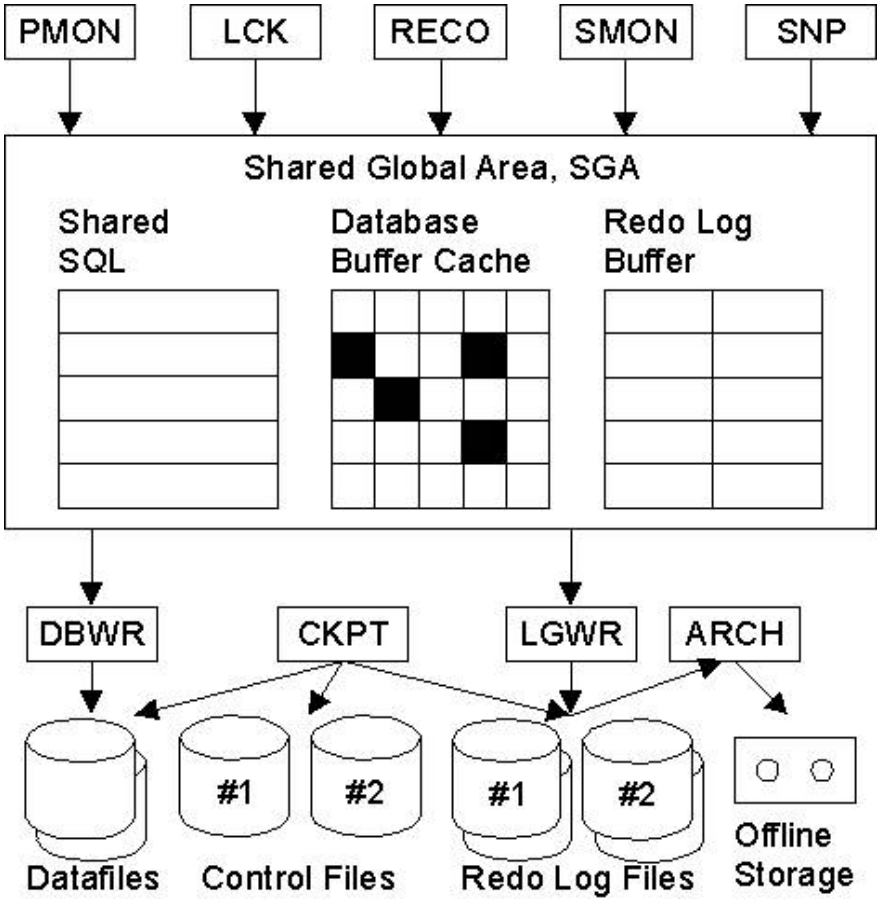
Este caché también se administra mediante el algoritmo LRU. El tamaño del caché está gestionado internamente por el servidor, pero es parte del *shared pool*, cuyo manaño viene determinado por el parámetro `SHARED_POOL_SIZE`.

1.4.2 Área Global de Programa

El *Program Global Area* es un área de memoria utilizada por un proceso Oracle. Esta zona de memoria no se puede compartir.

1.5 Estructuras de Proceso

El servidor se vale de una serie de procesos que son el enlace entre las estructuras físicas y de memoria. A continuación se describen cada proceso y el papel que juega en la gestión de laBD. Todo esto se puede ver en la siguiente figura.



System Monitor, *SMON*

El SMON es el supervisor del sistema y se encarga de todas las recuperaciones que sean necesarias durante el arranque. Esto puede ser necesario si la BD se paró inesperadamente por fallo físico, lógico u otras causas. Este proceso realiza la recuperación de la instancia de BD a partir de los ficheros *redo log*. Además límpia los segmentos temporales no utilizados y compacta los huecos libres contiguos en los ficheros de datos. Este proceso se despierta regularmente para comprobar si debe intervenir.

Process Monitor, *PMON*

Este proceso restaura las transacciones no validadas de los procesos de usuario que abortan, liberando los bloqueos y los recursos de la SGA. Asume la identidad del usuario que ha fallado, liberando todos los recursos de la BD que estuviera utilizando, y anula la transacción cancelada. Este proceso se despierta regularmente para comprobar si su intervención es necesaria.

Database Writer, *DBWR*

El proceso DBWR es el responsable de gestionar el contenido de los buffers de datos y del caché del diccionario. Él lee los bloques de los ficheros de datos y los almacena en la SGA. Luego escribe en los ficheros de datos los bloques cuyo contenido ha variado. La escritura de los bloques a disco es diferida buscando mejorar la eficiencia de la E/S.

Es el único proceso que puede escribir en la BD. Esto asegura la integridad. Se encarga de escribir los bloques de datos modificados por las transacciones, tomando la información del *buffer* de la BD cuando se valida una transacción. Cada validación no se lleva a la BD física de manera inmediata sino que los bloques de la BD modificados se vuelcan a los ficheros de datos periodicamente o cuando sucede algún *checkpoint* o punto de sincronizaión: grabación *diferida*:

- Los bloques del *buffer* de la BD (bloques del segmento de *rollback* y bloques de datos) menos recientemente utilizados son volcados en el disco continuamente para dejar sitio a los nuevos bloques.
- El bloque del segmento de *rollback* se escribe SIEMPRE antes que el correspondiente bloque de datos.
- Múltiples transacciones pueden solapar los cambios en un sólo bloque antes de escribirlo en el disco.

Mientras, para que se mantenga la integridad y coherencia de la BD, todas las operaciones se guardan en los ficheros de *redo log*. El proceso de escritura es asíncrono y puede realizar grabaciones multibloque para aumentar la velocidad.

Log Writer, *LGWR*

El proceso LGWR es el encargado de escribir los registros *redo log* en los ficheros *redo log*. Los registros *redo log* siempre contienen el estado más reciente de la BD, ya que puede que el DBWR deba esperar para escribir los bloques modificados desde el buffer de datos a los ficheros de datos.

Conviene tener en cuenta que el LGWR es el único proceso que escribe en los ficheros de *redo log* y el único que lee directamente los buffers de *redo log* durante el funcionamiento normal de la BD.

Coloca la información de los *redo log buffers* en los ficheros de *redo log*. Los *redo log buffers* almacenan una copia de las transacciones que se llevan a cabo en la BD. Esto se produce:

- a cada validación de transacción, y antes de que se comunique al proceso que todo ha ido bien,
- cuando se llena el grupo de *buffers* de *redo log*
- cuando el DBWR escribe *buffers* de datos modificados en disco.

Así, aunque los ficheros de DB no se actualicen en ese instante con los buffers de BD, la operación queda guardada y se puede reproducir. Oracle no tiene que consumir sus recursos escribiendo el resultado de las modificaciones de los datos en los archivos de datos de manera inmediata. Esto se hace porque los registros de *redo log* casi siempre tendrán un tamaño menor que los bloques afectados por las modificaciones de una transacción, y por lo tanto el tiempo que emplea en guardarlos es menor que el que emplearía en almacenar los bloques sucios resultado de una transacción; que ya serán trasladados a los ficheros por el DBWR. El LGWR es un proceso único, para asegurar la integridad. Es asíncrono. Además permite las grabaciones multibloque.

Checkpoint, *CKPT*

Este proceso escribe en los ficheros de control los *checkpoints*. Estos puntos de sincronización son referencias al estado coherente de todos los ficheros de la BD en un instante determinado, en un punto de sincronización. Esto significa que los bloques sucios de la BD se vuelcan a los ficheros de BD, asegurándose de que todos los bloques de datos modificados desde el último *checkpoint* se escriben realmente en los ficheros de datos y no sólo en los ficheros *redo log*; y que los ficheros de *redo log* también almacenan los registros de *redo log* hasta este instante. La secuencia de puntos de control se almacena en los ficheros de datos, *redo log* y control. Los *checkpoints* se produce cuando:

- un espacio de tabla se pone inactivo, *offline*,
- se llena el fichero de *redo log* activo,
- se para la BD,
- el número de bloques escritos en el *redo log* desde el último *checkpoint* alcanza el límite definido en el parámetro `LOG_CHECKPOINT_INTERVAL`,
- cuando transcurra el número de segundos indicado por el parámetro `LOG_CHECKPOINT_TIMEOUT` desde el último *checkpoint*.

Está activo si el parámetro `CHECKPOINT_PROCESS` tiene un valor verdadero.

Archiver, *ARCH*

El proceso archivador tiene que ver con los ficheros *redo log*. Por defecto, estos ficheros se reutilizan de manera cíclica de modo que se van perdiendo los registros *redo log* que tienen una cierta antiguedad. Cuando la BD se ejecuta en modo `ARCHIVELOG`, antes de reutilizar un fichero *redo log* realiza una copia del mismo. De esta manera se mantiene una copia de todos los registros *redo log* por si fueran necesarios para una recuperación. Este es el trabajo del proceso archivador.

Recoverer, *RECO*

El proceso de recuperación está asociado al servidor distribuido. En un servidor distribuido los datos se

encuentran repartidos en varias localizaciones físicas, y estas se han de mantener sincronizadas. Cuando una transacción distribuida se lleva a cabo puede que problemas en la red de comunicación haga que una de las localizaciones no aplique las modificaciones debidas. Esta transacción *dudosa* debe ser resuelta de algún modo, y esa es la tarea del proceso recuperador. Está activo si el parámetro `DISTRIBUTED_TRANSACTIONS` tiene un valor distinto de 0.

Lock, *LCK*

El proceso de bloqueo está asociado al servidor en paralelo.

1.6 Estructuras Externas

Por estructuras externas se entienden los ficheros que utiliza el servidor de BD, de los cuales ya se han ido contanto algunos aspectos, y otros se han ido intuyendo. Estos ficheros guardan información tanto de los datos almacenados en la BD como la necesaria para gobernar la propia BD.

Ficheros de la BD

En estos ficheros reside la información de la BD. Solo son modificados por el DBWR. A ellos se vuelcan los bloques sucios de la SGA cuando se hace una validación o cuando sucede un *checkpoint*. Las validaciones de las transacciones no producen un volcado inmediato, sino lo que se conoce por un *commit diferido*. Toda actualización se guarda en los ficheros de *redo log*, y se lleva a la BD física cuando tenemos una buena cantidad de bloques que justifiquen una operación de E/S. Almacenan los segmentos (datos, índices, *rollback*) de la BD. Están divididos en bloques (Bloque Oracle = c * Bloque SO), cada uno de los cuales se corresponde con un *buffer* del *buffer cache* de la SGA. En el bloque de cabecera no se guardan datos de usuario, sino la marca de tiempo del último *checkpoint* realizado sobre el fichero.

Ficheros *redo log*

En ellos se graba toda operación que se efectue en la BD y sirven de salvaguarda de la misma. Tiene que haber por lo menos 2, uno de ellos debe estar activo, *online*, y se escribe en ellos de forma cíclica. Existe la posibilidad de almacenar los distintos ficheros de *redo log* en el tiempo mediante el modo `ARCHIVER`. Así, se puede guardar toda la evolución de la BD desde un punto dado del tiempo.

Una opción es la utilización de archivos *redo log* multiplexados:

- Permite al LGWR escribir simultaneamente la misma información en múltiples archivos *redo log*.
- Se utiliza para protegerse contra fallos en el disco.
- Da una alta disponibilidad a los archivos *redo log* activos u *online*.

Esto se hace definiendo el número de *grupos* y de *miembros* de archivos *redo log* que van a funcionar en paralelo:

- grupos: funcionan como ficheros *redo log* normales, uno de ellos está activo y el resto espera su turno.
  - Su nombre lleva incorporado una numeración.
  - Deben contener todos el mismo número de miembros.
- miembros: cada escritura de un registro *redo log* se lleva a cabo en todos los miembros del grupo activo en ese momento. Los miembros deben:
  - tener el mismo tamaño y el mismo número de secuencia.
  - deben tener nombres similares y estar en diferentes discos para proteger contra fallos de una manera efectiva.

Cuando se produce algún fallo en los ficheros de *redo log* o en el proceso LGWR:

- Si la escritura en un fichero *redo log* falla pero el LGWR puede escribir al menos en uno de los miembros del grupo, lo hace , ignorando el fichero inaccesible y registrando un fallo en un fichero de traza o alerta.
- Si el siguiente grupo no ha sido archivado (modo ARCHIVELOG) antes del cambio de grupo que lo

pone activo, ORACLE espera hasta que se produzca el archivado.
- Si fallan todos los miembros de un grupo mientras el LGWR trata de escribir, la instancia se para y necesita recupeción al arrancar.

Se pueden visualizar los nombres y estado de los ficheros de *redo log*:

```
SVRMGR>select group# ,status,substr(member,1,60) from v$logfile;
```

También se pueden visualizar estadísticas de los ficheros *redo log*:

```
SVRMGR>select group# ,sequence# ,bytes,members,archived,
    2 status,firstchange# ,firsttime from v$logfile;
```

Ficheros de control

Mantienen la información física de todos los ficheros que forman la BD, camino incluido; así como el estado actual de la BD. Son utilizados para mantener la consistencia interna y guiar las operaciones de recuperación. Son imprescindibles para que la BD se pueda arrancar. Contienen:
- Infomación de arranque y parada de la BD.
- Nombres de los archivos de la BD y *redo log*.
- Información sobre los *checkpoints*.
- Fecha de creación y nombre de la BD.
- Estado *online* y *offline* de los archivos.

Debe haber múltiples copias en distintos discos, mínimo dos, para progerlos de los fallos de disco. La lista de los ficheros de control se encuentra en el parámetro `CONTROL_FILES`, que debe modificarse con la BD parada.

Se puede componer una sentencia SQL que nos muestre todos los ficheros asociados a una BD. Esta es:

```
SQL>select 'control' tipo,substr(name,1,70) nombre from v$controlfile
2  union all
3  select 'datos' tipo,substr(name,1,70) nombre from v$datafile
4  union all
5  select 'redo log' tipo,substr(name,1,70) nombre from v$logfile
6  /
```

Hasta aquí los tipos de ficheros que se suelen considerar fundamentales en la arquitectura del SGBD Oracle. Pero existen otros ficheros, que aunque no forman parte de la arquitectura Oracle resultan importantes en el uso del SGBD.

El Fichero `INIT.ORA`

Como parte de la distribución software, Oracle provee de un fichero de parámetros de inicialización llamado `init.ora`. Este fichero contiene los parámetros del sistema Oracle y debe ser utilizado por el DBA para configurar el SGDB y adecuarlo a una determinada explotación. Oracle lee este fichero durante el proceso de arranque para determinar el tamaño de la SGA y encontrar los ficheros de control, entre otros menesteres.

Como el fichero `init.ora` es fundamental para el arranque de la BD, debería ser copiado frecuentemente para protegerlo de posibles prédidas.

Ficheros de Traza

Orace crea ficheros de texto llamados de traza para ayudar en la diagnosis de problemas y en el ajuste del SGBD. Cada proceso del servidor escribe en un fichero de traza asociado cuando es necesario. Los

procesos de usuarios también pueden tener asociados ficheros de traza. La situación de estos ficheros de traza del sistema se especifica por el parámetro `BACKGROUND_DUMP_DEST`, y los de usuario por `USER_DUMP_DEST`. Oracle crea ficheros de traza automáticamente cuando ocurre algún error.

Un parámetro muy frecuentemente utilizado por los desarrolladores Oracle es el SQL_TRACE, que cuando está puesto a TRUE produce que toda sentencia SQL ejecutada genere información en los ficheros de traza. Este parámetro se puede variar con el siguiente comando:

```
SQL>alter session set SQL_TRACE=TRUE;
Session Altered.
```

El directorio donde se depositan los ficheros de traza debe de examinarse con regularidad para controlar el tamaño de los fichero alli depositados.

2 Ciclo de Ejecución

Para ilustrar el funcionamiento del servidor Oracle vamos a ver el ciclo de ejecución de una sentencia de lectura y otra de actualización.

2.1 Ciclo de Lectura

Las sentencias de lectura siguen el siguiente ciclo:

1. El proceso cliente pasa la sentencia SQL (`SELECT`) al proceso servidor por medio de la SGA.
2. Los procesos del servidor buscan en la zona de SQL compartido una versión ejecutable de la sentencia. Si la encuentran no tienen que procesarla.
3. Se procesa la sentencia SQL y su versión ejecutable se coloca en la zona de SQL compartido.
4. El proceso del servidor intenta leer los bloques de datos de la SGA. Si no están, se han de leer del fichero de datos. Si los bloques están en la SGA pero han sido modificados por otro usuario y esa modificación no ha sido validada aún, el proceso de servidor debe reconstruir la imagen de la fila a partir de los segmentos de *rollback*, para conseguir consistencia en lectura.
5. El proceso servidor pasa los datos solicitados al proceso cliente.

2.2 Ciclo de Actualización

Las sentencias de actualización siguen el siguiente ciclo:

1. El proceso cliente pasa la sentencia SQL (`UPDATE`) al proceso servidor por medio de la SGA.
2. Los procesos del servidor buscan en la zona de SQL compartido una versión ejecutable de la sentencia. Si la encuentran no tienen que procesarla.
3. Se procesa la sentencia SQL y su versión ejecutable se coloca en la zona de SQL compartido.
4. El proceso del servidor intenta leer los bloques de datos de la SGA. Si no están, se han de leer del fichero de datos.
5. Se registra el valor antiguo de los datos en un segmento de *rollback* y se crea un registro *redo log*.
6. Se crea una copia de la transacción en un registro *redo log*.
7. Se ejecuta la sentencia SQL modificando los datos, y se crea un registro *redo log* que así lo refleja.
8. El proceso usuario valida la transacción (`COMMIT`), registrandose en un registro *redo log*.
9. El LGWR escribe los *buffers* del *redo log* en el disco.
10. El servidor indica al cliente que la operación ha sido completada de manera satisfactoria.
11. Se registra la terminación de la transacción en un registro *redo log*.
12. Se libera la información del *rollback*, pues ya no va a necesitarse.

Si a partir del paso 6 el usuario cancela la transacción (`ROLLBACK`), se puede utilizar la información de *rollback* para restablecer el valor original.

Si sucede algo que impida que la transacción validada por el usuario pueda llevarse a cabo, se puede utilizar la

información contenida en los registros *redo log* para rehacer la transacción (a partir del paso 6).

Como ocurre con todas las transacciones, en algún momento el DBWR escribe en el archivo de datos la copia de los bloques de datos modificados que se encuentran en el *buffer cache*.

3 Configuración

3.1 El Código Oracle

Cuando el software Oracle se instala en un sistema, se crean subdirectorios y ficheros, dependientes todos ellos del S.O. Por ejemplo, en el S.O. Unix, todo los subdirectorios Oracle se encuentran colgando del directorio principal `ORACLE_HOME`. Todos estos subdirectorios contienen ficheros ejecutables y *scripts* que son cruciales para el funcionamiento y la administración del SGBD, y es lo que se conoce por el código Oracle. Entre ellos, una herramienta nos va a ser fundamental en las tareas de adminstración y puesta en marcha de la BD: *server manager*, `svrmgr`. Con ella son convertiremos en DBA, y para ejecutarla deberemos ser sus propietarios. La sentencia es la siguiente:

```
SVRMGR>connect internal
Connected.
```

Todas las operaciones de administración deben comenzar por conectarse a la BD.

3.2 Arranque y Parada de la BD

Durante el arranque y parada de la BD se sucenden un conjunto de eventos que llevan a la BD por diferentes estados.

Para que los usuarios puedan acceder a la BD el DBA necesita abrir la BD. El siguiente es un ejemplo de apertura de una BD llamada `test`.

```
SVRMGR>startup open test
ORACLE instance started.
Total System Global Area 4512688 bytes.
Fixed Size                39732 bytes.
Variable Size           4055164 bytes.
Database Buffers         409600 bytes.
Redo Buffers               8192 bytes.
Database mounted.
Database opened.
```

cuando se ejecuta el comando `startup open` la BD pasa por tres estados (*nomount*, *mount* y *open*) antes de estar disponible. El DBA puede arrancar la BD hasta uno de los estados con el comando `startup`: `startup nomount`, `startup mount`. A continuación vamos a describir cada uno de los estados por los que pasa la BD en el proceso de arranque.

nomount

```
SVRMGR>startup open test
ORACLE instance started.
Total System Global Area 4512688 bytes.
Fixed Size                39732 bytes.
Variable Size           4055164 bytes.
```

```
Database Buffers        409600 bytes.
Redo Buffers            8192 bytes.
```

Oracle lee el fichero `init.ora`, localiza los ficheros de control, crea e inicializa la SGA, y finalmente arranca todos los procesos Oracle. En este estado la instancia de BD está arrancada. Se deberá llevar la BD al estado *nomount* cuando se esté creando la BD o cuando se está restaurando un fichero de control después de haberlo perdido.

mount

```
SVRMGR>alter database mount;
Statement processed.
```

Oracle abre los ficheros de control para localizar los ficheros de datos y los *redo log*, pero no se realizan ninguna comprobación en ellos en este momento. La instancia monta la BD y la bloquea, verificando que ninguna otra instancia ha montado la misma BD.

Hay varias razones para querer tener la BD en el estado *mount*. En general, todas las sentencias SQL del tipo `alter database` se deben ejecutar en esta etapa. Algunas de las operaciones a realizar cuando la BD está montada son:

- efectuar recuperaciones,
- poner *online*/*offline* un fichero de datos,
- recolocar los ficheros de datos y *redo log*,
- crear un nuevo grupo o miembro *redo log*, o borrar un grupo o miembro *redo log* existente.

open

```
SVRMGR>alter database open;
Statement processed.
```

Durante esta etapa, la instancia abre la BD, bloquea los ficheros de datos, y abre todos los ficheros *redo log*. Si la instancia abre la BD después de una terminación anormal, o después de una caida, se ejecutará automáticamente el proceso de recuperación utilizando los ficheros *redo log*. Al final de esta etapa la BD está dispuesta para su uso normal.

Para parar la BD el comando base es `shutdown` como se puede ver en el siguiente ejemplo:

```
SVRMGR>shutdow
Database closed.
Database dismounted.
ORACLE instance shutdown.
```

Pero este comando se nos presenta con tres opciones: *normal*, *immediate* y *abort*.

shutdown normal

Se impide el acceso a la BD, espera a que todos los usuarios completen todas sus peticiones y se desconecten del servidor. Purga todos los buffers de datos y cachés de *redo log*, actualizando los ficheros de datos y de *redo log*, se eliminan los bloqueos de ficheros, se completan las transacciones en marcha, se actualizan las cabeceras de ficheros, elimina los *threads*, libera los bloqueos de la BD por parte de la instancia, y sincroniza los ficheros de control y de datos. En resumen, la opción normal cierran la BD, desmonta la BD y para la instancia con cuidado y es la opción recomendada para parar la BD.

shutdown immediate

En ciertas ocasiones puede ser necesario parar la BD de modo inmediato. Si es así, las sentencias en proceso son terminadas inmediatamente, cualquier transacción no confirmada (*uncommitted*) es vuelta atrás (*rolled back*) y la

BD es parada. La única desventaja de utilizar esta opción es que Oracle no espera a que los usuarios se desconecten. Sin embargo, la BD será consistenta y no se necesitará recuperación en el siguiente arranque.

shutdown abort

En situaciones de emergencia, y cuando todo lo demás falla, se debe realizar una parada de este tipo. Por ejemplo, cuando un proceso de la instancia muere y la BD no puede pararse de modo normal o inmediato. Cuando se utiliza la opción *abort* las sentencias SQL son terminadas bruscamente, y las transacciones no confirmadas no son vueltas atrás. Parar la BD con la opción *abort* requiere recuperación en la siguiente vez que arranque la BD y esta opción debe ser utilizada sólo cuando no quede más remedio.

3.3 Almacenamiento de Datos

Los datos se almacenan en estacios de tablas, y un espacio de tabla es la entidad lógica que se corresponde con uno o más ficheros físicos. La principal razón de esta organización es el aumento de la flexibilidad a la hora de realizar operaciones con la BD. En esta sección vamos a dar un repaso a las tareas de administración relacionadas con los espacios de tablas y con los ficheros.

3.3.1 Espacios de Tablas y Ficheros

Los espacios de tablas se utilizan para realizar tareas de gestión de espacio, controlar la disponibilidad de los datos y ejecutar copias de seguridad y recuperaciones parciales.

Gestión de Espacio

El primer espacio de tablas es el `SYSTEM`. Este espacio de tablas debe estar disponible siempre durante el funcionamiento normal de la BD porque contiene el diccionario de datos. Después de la creación de la BD, se recomienda la creación de otros espacios de tablas para que los datos de los usuarios puedan ser separados de los del diccionario de datos. Incluso, si varias apliaciones se van a ejecutar sobre la misma BD es recomendable que sus datos estén separados. Para crear un espacio de tablas se puede utilizar el comando `create tablespace`:

```
SVRMGR>create tablespace nombre_tablespace
   2>datafile 'nombre_fichero' size 50Monline;
```

En el ejemplo anterior se ha creado un espacio de tablas de 50 Mb. de tamaño. Cada espacio de tabla tiene un conjunto de parámetros de almacenamiento que controla su crecimiento:

- *initial*: tamaño de la extensión inicial (10k).
- *next*: tamaño de la siguiente extensión a asignar (10k).
- *minextents*: número de extensiones asignadas en el momento de la creación del espacio de tablas (1).
- *maxextents*: número máximo de extensiones.
- *pctincrease*: Porcentaje en el que crecerá la siguiente extensión antes de que se asigne, en relación con la última extensión utilizada.
- *optimal*: Tamaño óptimo declarado para este espacio de tablas.
- *pctused*: porcentaje de utilización del bloque por debajo del cual Oracle considera que un bloque puede ser utilizado para insertar filas nuevas en él.

Si el espacio de tablas necesita más espacio después de su creación se puede puede alterar para añadir uno o más ficheros. Para ello se puede utilizar el comando `alter tablespace`:

```
SVRMGR>alter tablespace nombre_tablespace
   2>add datafile 'nombre_fichero' size 30M;
```

Si se necesitara variar la localización de los ficheros asociados a un espacio de tablas se puede hacer con los comandos `alter tablespace` (el espacio de tables debe estar *offline*) o `alter database` (la BD debe estar montada pero no abierta). Antes de ejecutar los anteriores comandos los ficheros asociados al espacio de tablas

deben de haber sido movidos a su nueva localización utilizando los comandos del SO oportunos.

Poniendo los *tablespaces offline*

Llevar a un espacio de tablas al estado *offline* significa que se impide el acceso a los datos que almacena. El espacio de tablas `SYSTEM` nunca puede estar *offline*. Las razones para poner un espacio de tablas *offline* pueden ser varias: un error de escritura en los ficheros que lo soportan, el mover los ficheros de sitio, etc. Depués de realizar estas operaciones hay que poner otra vez disponible el espacio de tablas, esto es *on line*

Los espacios de tablas se pueden poner *offline* de tres modos: *normal*, *temporary* e *immediate*. Si no existe ningún error lo recomendable es poner el espacio de tablas *offline* usando el modo normal. Así, se colocará un *checkpoint* en el espacio de tablas antes de ponerlo *offline*.

```
SVRMGR>alter tablespace nombre_tablespace offline normal;
```

Si alguno de los ficheros está corrupto, la opción *normal* fallará y se necesitará el modo *temporary*. La opción *immediate* se utilizará sólo cuando la BD está en modo `ARCHIVELOG`, ya que no se produce *checkpoint* alguno.

Poniendo los ficheros *offline*

No es normal poner los ficheros *offline*/*online*. Si un determinado fichero de datos se corrompe, se tendrá que pone *offline*, repararlo y ponerlo *online* de nuevo. Esta operación puede suponer sustituirlo por su copia de seguridad, lo que implicará ejecutar el comando `recover datafile` antes de poner el fichero *online*.

3.3.2 Segmentos, Extensiones y Bloques

Los datos en la BD son almacenados físicamente en bloques Oracle: la mínima unidad de espacio físico, y es un múltiplo del bloque del SO (2 Kb usualmente). El tamaño del bloque Oracle se fija por el parámetro `DB_BLOCK_SIZE` del fichero `init.ora`. Un tamaño grande de bloque mejora la eficiencia del cache de E/S, pero el tamaño de la SGA aumentará para contener los mismos `DB_BLOCK_BUFFERS`, lo que significa un problema de memoria.

Una serie de bloques contiguos es una extensión, que es una unidad lógica de almacenamiento. Una serie de extensiones es un segmento. Cuando un objeto es creado, se reserva una extensión en su segmento. Cuando el objeto crezca, necesitará más espacio y se reservarán más extensiones.

Cada segmento tiene un conjunto de parámetros de almacenamiento que controla su crecimiento:

- *initial*: tamaño de la extensión inicial (10k).
- *next*: tamaño de la siguiente extensión a asignar (10k).
- *minextents*: número de extensiones asignadas en el momento de la creación del segmento (1).
- *maxextents*: número máximo de extensiones (99).
- *pctincrease*: Porcentaje en el que crecerá la siguiente extensión antes de que se asigne, en relación con la última extensión utilizada (50).
- *pctfree*: porcentaje de espacio libre para actualizaciones de filas que se reserva dentro de cada bloque asignado al segmento (10).
- *pctused*: porcentaje de utilización del bloque por debajo del cual Oracle considera que un bloque puede ser utilizado para insertar filas nuevas en él.
- *tablespace*: nombre del espacio de tablas donde se creará el segmento.

Cuando se diseña una BD se ha de tener mucho cuidado a la hora de dimensionar la BD y prever el crecimiento de las tablas. A continuación se hacen algunas consideraciones sobre la gestión del espacio para los diferentes segmentos.

Segmentos de Datos

El espacio del diccionario de datos se suele mantener más o menos constante, aunque es crítico que tenga suficiente espacio para crecer en el espacio de tablas `SYSTEM`. Así, hay que tener cuidado de colocar las tablas de

usuario, los índices, segmentos temporales y los segmentos de *rollback* en otros espacios de tablas. Además, es recomendable que el espacio de tablas `SYSTEM` esté al 50% o 75% de su espacio disponible. Finalmente, asegurarse que los usuarios no tienen privilegios de escritura en el espacio de tablas `SYSTEM`.

Las tablas crecen proporcionalmente con el número de filas, ya que se puede suponer que la longitud de las filas es constante.

Segmentos de Índice

Los índices crecen en tamaño en mayor proporción que las tablas asociadas si los datos en la tabla son modificados frecuentemente. La gestión del espacio es mejor si se mantienen los índices de tablas grandes en espacios de tablas separados.

Segmentos de *Rollback*

Los segmentos de *rollback* almacenan la imagen anterior a una modificación de un bloque. La información en el segmento de *rollback* se utiliza para asegurar la consistencia en lectura, el *rollback* (el valor en el segmento de *rollback* se copia en el bloque de datos) y la recuperación.

Es importante comprender cual es el contenido de un segmento de *rollback*. No almacenan el bloque de datos modificado entero, sólo la imagen previa de la fila o filas modificadas. La información del segmento de *roolback* consiste en varias entradas llamadas *undo*. Por ejemplo, si se inserta una fila en una tabla, el *undo* necesitará sólo el *rowid* de la fila insertada, ya que para volver atrás la insercion sólo hay que realizar un `delete`. En las operación de actualización, se almacenará el valor antiguo de las columnas modificadas. El segmento de *rollback* asegura que la información *undo* se guardan durante la vida de la transacción.

Un segmento de *rollback* como cualquier otro segmento consiste en una serie de extensiones. Sin embargo, la mayor diferencia entre un segmento de datos y otro *rollback* es que en este último las extensiones se utilizan de manera circular. Así, habrá que tener cuidado a la hora de fijar el tamaño del segmento de *rollback* para que la cabeza no pille a la cola.

Segmentos Temporales

Los segmentos temporales se crean cuando se efectuan las siguientes operaciones:

- `Create Index`
- `Select` con `distinct`, `order by`, `union`, `intersect` y `minus`.
- uniones no indexadas.
- Ciertas subconsultas correlacionadas.

Si las tablas a ordenar son pequeñas la ordenación se realiza en memoria principal, pero si la tabla es grande se realiza en disco. El parámetro `SORT_AREA_SIZE` determina el lugar donde se hace la ordenación. Incrementándole se reduce la creación de segmentos temporales.

3.4 Configuración de la BD

Mientras se diseña la BD hay que considerar la posible recuperación de una caida, y las prestaciones de la BD, relacionando todo esto con las necesidades de la implantación y los medios disponibles. La configuración de la BD está relacionada con los ficheros de control, los ficheros *redo log* activos y los archivados.

3.4.1 Gestionando los Ficheros de Control

Los ficheros de control contienen el esquema de la BD. Es uno de los más importantes ficheros e imprescindible para el uso normal de la BD. Así que daremos alguna pista para su gestión.

El parámetro `CONTROL_FILES` del fichero `init.ora` contiene la lista de todos los ficheros de control. Cuando se arranca la BS, Oracle lee el fichero `init.ora` para determinar cuántos ficheros de control se usan en la BD y dónde están. Durante la fase de montaje, se abren los ficheros de control para leer el esquema de la BD. Aunque Oracle escribe en todos los ficheros de control, sólo lee el primero listado en el parámetro `CONTROL_FILES`.

Para protegerlos contra fallos de almacenamiento, se sugiere que al menos existan dos ficheros de control, cada uno en un disco diferente, aunque es buena idea mantener más copias en diferentes discos. Esto es una política de *espejado* que protege frente a fallos en disco. Si un disco falla y se pierden todos los ficheros en él, se puede seguir utilizando los ficheros de control de otros discos. Esto supone una pequeña sobrecarga al sistema, ya que cada vez que se porduce un *checkpoint* o cambia el esquema de la BD, todos los ficheros de control son actualizados.

Cuando se produce un fallo en algún disco y algún fichero de control se pierde hay que parar la BD con la opción *abort*, copiar el fichero de control que queda en otro disco, editar el fichero `init.ora` para reflejar este cambio, y volver a levantar la BD.

Si un fallo ha producido la pérdida de todas las copias de los ficheros de control habrá que recrearlos con el comando `create controlfile`. Si algunos de los parámetros `MAXLOGFILES`, `MAXLOGMEMBERS`, `MAXLOGHISTORY`, `MAXDATAFILES` y `MAXINSTANCES` varía habrá que utilizar también el comando `CREATE CONTROLFILE`.

3.4.2 Gestionando los Ficheros *Redo Log* Activos

Oracle proporciona la posibilidad de *espejar* los ficheros *redo log* activos. Mecanismo conocido como ficheros *redo log* multiplexados. Oracle necesita al menos dos grupos de fricheros *redo log*, cada uno con un miembro como mínimo. Oracle efectua escrituras en paralelo a cada miembro, pero si están en el mismo disco, realmente la escritura se serializa.

Otro aspecto a tener en cuenta es el tamaño de los ficheros *redo log*. Si son muy pequeños, el LGWR deberá cambiar de ficheros demasiado frecuentemente, lo que reduce su rendimiento. Por otro lado, si los ficheros *redo log* son demasiado grandes, se necesitará mucho tiempo en las recuperaciones, ya que se tendrán que recuperar muchas transacciones.

Otro aspecto muy importante es la elección del número correcto de grupos, ya que disponer de demasiados pocos grupos puede acarrear problemas cuando estámos en modos `ARCHIVELOG` y tenemos una tasa de transacciones muy alta. Esto puede suponer que un grupo que todavía está archivando por el proceso `ARCH` se convierta en el grupo en el que el LGWR necesite escribir, lo que produciría que la BD se parara, ya que el LGWR tienen que esperar a que el grupo esté disponible, una vez que su contenido ha sido archivado. Para la mayoría de las implantaciones, tener entre 2 y 10 grupos puede ser suficiente. El número de grupos no puede exceder de `MAXLOGFILES`, ni el número de miembros puede ser mayor que `MAXLOGMEMBERS`.

---

4 Creación de una BD Ejemplo

A continuación se muestran los *scripts* de creación de una BD llamada *demo*. Este conjunto de *scripts* es:

`crdbdemo.sql`
: *Script* de creación de la BD llamada demo. Antes de ejecutarlo hay que asegurarse que los *path* en él referenciados sean consecuentes con la implantación. Además la variable `ORACLE_SID` debe ser puesta a `demo`

`configdemo.ora`
: *Script* con los parámetros de configuración básicos. Estos parámetros no suelen cambiar. Este fichero es incluido en `initdemo_0.ira` y en `initdemo.ora`.

`initdemo_0.ora`
: *Script* con los parámetros de configuración iniciales. Se utilizará sólo en la fase de creación de la BD.

`initdemo.ora`
: *Script* con los parámetros de configuración.

`crdbdemo.sql`

```
REM#
REM# scriptde creacion para bd DEM
REM# fichero:  crdbdemo.sql
```

```
REM* localizacion:  /opt/app/oracle/admin/demo
REM* ORACLE_HOME:  /opt/app/oracle/product/7.3.2
REM*

set termout on
set echo on
spool crdbdemo.log

REM*
REM* creacion inicial de la BD
REM* Paso 1:  crear la BD, con los ficheros de control
REM* especificados en el fichero initdemo.ora
REM* ORACLE_SID debe ser igual a demo
REM*
REM* arrancar la BD demo con el fichero initdemo.ora
REM*

connect internal
startup pfile=/opt/app/oracle/admin/demo/initdemo.ora nomount

REM*
REM* crear la BD demo
REM* Guia de configuracion del tablespace SYSTEM
REM*  ORACLE RDBMS  5M
REM*  Espacio adicional del diccionario para aplicaciones 10-50M
REM* Guia de configuracion de los Redo log:
REM*  Utilizar 3+ ficheros redo log para disminuir las esperas.
REM*  Utilizar 2Mb por redo y por conexion para reducir checkpoints.
REM*

create database "demo"
maxinstances 1
maxlogfiles 16
character set "WE8DEC"
datafile
  '/export/home/oradata/demo/system01.dbf'  size 20M reuse
logfile
  '/export/home/oradata/demo/redodemo01.log'  size 2M reuse,
  '/export/home/oradata/demo/redodemo02.log'  size 2M reuse,
  '/export/home/oradata/demo/redodemo03.log'  size 2M reuse;

disconnect

REM*
REM* la BD debera estar arrancada en este momento.
REM*

connect internal

REM*
REM* Paso 2:  crear el diccionario
REM* el fichero catalog.sql viene con la instalacion del software Oracle.
REM*

@/opt/app/oracle/product/7.3.2/rdbms/admin/catalog.sql

REM*
```

```
REM* Paso 3:  crear un segundo segmento rollback (r0) en SYSTEM
REM*

connect internal
create rollback segment r0 tablespace system
storage (initial 16k next 16k minextents 2 maxextents 20);

REM*
REM* Paso 4:  Alterar el nuevo segmento de rollback para hacerlo disponible
REM* Utilizar ALTER ROLLBACK SEGMENT ONLINE para poner r0 online
REM* sin tirar la BD y volverla a arrancar.
REM*

alter rollback segment r0 online;


REM*
REM* crear nuevos tablespaces
REM*
REM* Paso 5:  crear el tablespace RBS para los segmentos de rollback
REM* Guia de configuracion de los segmentos rollback:
REM*   1 segmento rollback por cada 4 acciones concurrentes.
REM*   No mas de 50 segmentos rollback.
REM*   Todos los segmentos rollback del mismo tamano.
REM*   Entre 2 y 4 extensiones de igual tamano por segmento rollback.
REM*


create tablespace rbs datafile
  '/export/home/oradata/demo/rbs01.dbf' size 8M reuse,
  '/export/home/oradata/demo/rbs02.dbf' size 8M reuse,
  '/export/home/oradata/demo/rbs03.dbf' size 8M reuse,
default storage (
      initial       128k
      next          128k
      pctincrease     0
      minextents      2
      maxextents      50
);

REM*
REM* Paso 6:  crear el tablespace (TEMP) para segmentos temporales.
REM* Guia de configuracion del tablespace temporal:
REM*   Initial y next extensiones = k * SORT_AREA_SIZE, k en {1,2,...}.
REM*

create tablespace temp datafile
      '/export/home/oradata/demo/temp01.dbf'  size   1M reuse
default storage (
      initial    500k
      next       500k
      pctincrease 0
);

REM*
REM* Paso 7:  crear el tablespace (TOOLS) para las herramientas de bd
REM*
```

```
create tablespace tools datafile
  '/export/home/oradata/demo/tools01.dbf' size 15M;


REM#
REM# Paso 8:  crear el tablespace para los usuarios.
REM#

create tablespace users datafile
     '/export/home/oradata/demo/users01.dbf' size  1M;

REM#
REM# Paso 9:  crear segmentos rollback
REM# crear 2 segmentos rollback en el tablespace RBS
REM#

create rollback segment r01 tablespace rbs;
create rollback segment r02 tablespace rbs;

REM#
REM# Paso 10:  desactivar el segundo segmento rollback en el
REM# tablespace SYSTEM
REM# Utilizar ALTER ROLLBACK SEGMENT ONLINE para poner los
REM# segmentos rollback online sin tirar la BD y volverla a arrancar.
REM#

alter rollback segment r01 online;
alter rollback segment r02 online;

REM#
REM# Como ya hay 2 segmentos rollback creados y online,
REM# no necesitamos el segmento rollback en el tablespace SYSTEM
REM#

alter rollback segment r0 offline;
drop rollback segment r0;

REM#
REM# Modificar los usuarios SYS y SYSTEM
REM# designar tablespace temporal a TEMP
REM# designar tablespace por defecto para todos los usuarios TOOLS
REM#

alter user sys temporary tablespace temp;
alter user system default tablespace tools temporary tablespace temp;

REM#
REM# Paso 11:  Para cada usuario DBA, ejecutar el script de
REM# creacion de sinonimos.  No olvidar ejecutarlo para cada
REM# usuario DBA creado en el futuro.
REM#

connect system/manager
@/opt/app/oracle/product/7.3.2/rdbms/admin/catdbsyn.sql

spool off
```

configdemo.ora

```
# Fichero de Configuracion Oracle 7
# fichero:  configdemo.ora
# localizacion:  /opt/app/oracle/admin/demo

control_files = (/export/home/oradata/demo/control01.ctl,
           /export/home/oradata/demo/control02.ctl,
           /export/home/oradata/demo/control03.ctl)
background_dump_dest= /opt/app/oracle/admin/demo/bdump
core_dump_dest= /opt/app/oracle/admin/demo/cdump
user_dump_dest= /opt/app/oracle/admin/demo/udump
log_archive_dest= /opt/app/oracle/admin/demo/arch/arch.log
db_block_size  = 2048
db_name = demo
```

initdemo_0.ora

```
#
# Fichero initdemo_0.ora, utilizado en la creacion de la BD
# localizacion:  /opt/app/oracle/admin/demo
#

# incluir los parametros de configuracion de la BD
ifile = /opt/app/oracle/admin/demo/configdemo.ora

rollback_segments = ()

db_files = 20

db_file_multiblock_read_count= 8
db_block_buffers = 200
shared_pool_size = 3500000
log_checkpoint_interval = 10000
processes = 50
dml_locks = 100
log_buffer = 8192
sequence_cache_entries = 10
sequence_cache_hash_buckets = 10
# audit_trail = true         # si quieres auditar
# timed_statistics = true     # si quieres estadisticas de tiempo
max_dump_file_size = 10240     # limita tamano del fichero traza a
5Mb each
# log_archive_start= true     # si quieres archivado automatico
compatible = 7.1.0.0
global_names = TRUE
```

initdemo.ora

```
#
# Fichero initdemo.ora
# localizacion: /opt/app/oracle/admin/demo
#

# incluir configuracion de los parametros de la BD
ifile = /opt/app/oracle/admin/demo/configdemo.ora

rollback_segments = (r01,r02)

db_files = 20

db_file_multiblock_read_count= 8
db_block_buffers = 200
shared_pool_size = 3500000
log_checkpoint_interval = 10000
processes = 50
dml_locks = 100
log_buffer = 8192
sequence_cache_entries = 10
sequence_cache_hash_buckets = 10
# audit_trail = true          # si quieres auditar
# timed_statistics = true     # si quieres estadisticas de tiempo
max_dump_file_size = 10240    # limita tamano del fichero traza a
5Mb each
# log_archive_start = true    # si quieres archivado automatico
compatible = 7.1.0.0
global_names = TRUE

remote_os_authent = TRUE
os_authent_prefix = ""
```
