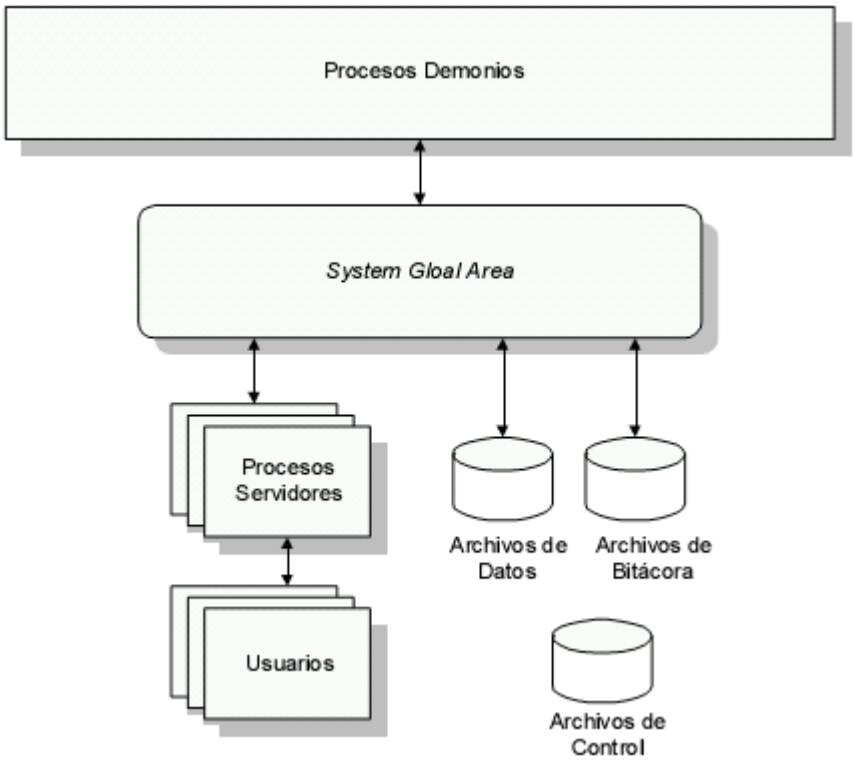
TEMA 1: Conceptos básicos de Oracle

NOTA: Todas las pruebas de este estudio se harán bajo el usuario 'SCOTT´y usando el programa de Oracle "SQL*Plus" como interfaz. Éste se inicia tecleando en una ventana MS-DOS: sqlplusw.exe. Previamente se habrá concedido a este usuario los roles SELECT_CATALOG_ROLE y PLUSTRACE para que pueda ver el diccionario de datos como usuario DBA mediante el comando:
`GRANT select_catalog_role,plustrace TO SCOTT;`

Introducción a la arquitectura de Oracle.-

Una primera y sencilla aproximación a la arquitectura general utilizada por el RDBMS ORACLE para el manejo de base de datos (independientemente de la configuración –single, multi-thread, parallel- utilizada) es la mostrada en la siguiente figura.



Esta arquitectura puede ser divida en dos porciones lógicas:

- estructura de procesos y memoria
- estructura para el manejo de los datos

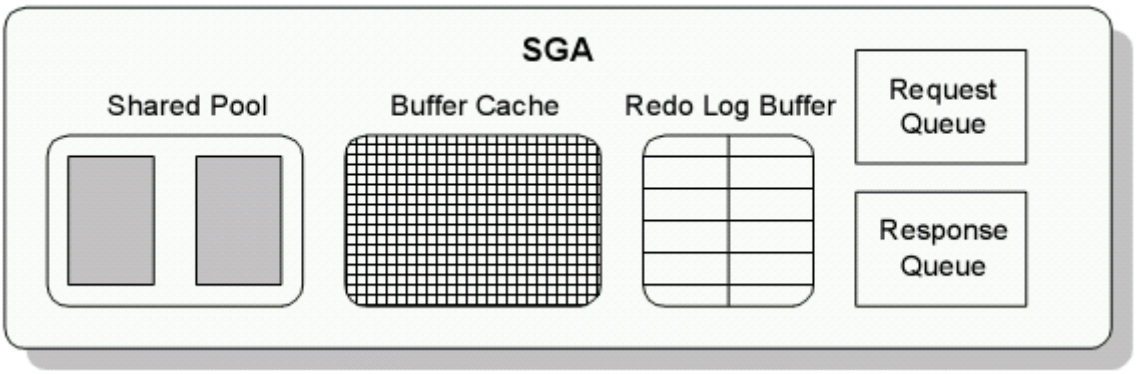
Estructura de procesos y memoria.-

Independientemente de la arquitectura computacional, o de su configuración, cada base de datos dentro del RDBMS ORACLE es asociada a una determinada instancia, y de igual forma una instancia puede abrir y utilizar sólo una base datos ORACLE en cualquier momento de su ejecución. Es posible poseer múltiples instancias ejecutándose concurrentemente dentro de una misma máquina, cada una accediendo su propio espacio físico de datos (su base de datos ORACLE). En el sistema de operación, la variable de entorno ORACLE_SID permite identificar el nombre de la instancia ORACLE a la cual se conectarán, por defecto, las aplicaciones de usuario.

Cada vez que el RDBMS ORACLE es inicializado, tanto el System Global Area (SGA) como los procesos demonios son levantados. El SGA junto con los procesos demonios es lo que se demonina como una instancia ORACLE.

La memoria en Oracle.-

El System Global Area es un conjunto de estructuras de memoria compartida que contienen datos e información de control para una determinada instancia ORACLE. El SGA se mantiene en la memoria virtual del computador en el que reside la instancia ORACLE. Si dentro de la instancia existe la posibilidad de que más de un usuario se encuentren conectados simultáneamente, los datos dentro del SGA de la instancia son compartidos entre todos los usuarios. Es por esto que algunas veces al SGA también se le suele denominar Shared Global Area. La estructura interna de la SGA de un RDBMS "shared server" puede observarse en esta figura:



Veamos que es lo que hace cada componente:

- **Buffer Cache (o Database Buffer Cache):** Su función es mantener bloques de datos leídos directamente de los archivos de datos. Cuando se procesa una consulta, el servidor busca los bloques de datos requeridos en esta estructura. Si el bloque no se encuentra en esta estructura, el proceso servidor lee el bloque de la memoria secundaria y coloca una copia en esta estructura. De esta forma, otras peticiones que requieran de este bloque de datos no requerirán de acceso a memoria secundaria (lecturas físicas).

  El Buffer Cache está organizado en dos listas: la lista de bloques sucios (dirty buffers) y la lista de los más recientemente usados (= L.R.U. "last recent used"). La lista de sucios almacena los bloques sucios, que contienen datos que han sido modificados pero que no han sido escrito todavía a disco. La lista LRU mantiene los buffers libres (no modificados y disponibles), los reservados (pinned buffers) que son lo que actualmente son accedidos y los bloques sucios que todavía no han sido escritos a disco. El número de bloques manejados por el Buffer Cache puede ser configurado para mejorar el rendimiento, así como el tamaño del bloque de datos. En cualquier caso el tamaño de bloque de datos utilizado debe ser el mismo que el que se ha configurado para la instancia como tamaño de bloque de datos utilizado por el RDBMS. No obstante se pueden crear cachés adicionales con tamaños de bloque diferentes (2Kb, 4K, 8Kb, 16Kb ó 32Kb) para que sean usados con tablespaces con un tamaño de bloque diferente. Los parámetros para estas subcachés son del tipo DB_nK_CACHE_SIZE, siendo n uno de los cinco valores indicados antes.

- **Redo Log Buffer:** Es un buffer circular que mantiene todos los cambios que han sido realizados sobre la base de datos por operaciones INSERT, UPDATE, DELETE, CREATE, ALTER y DROP. Las entradas de este buffer contienen toda la información necesaria para reconstruir los cambios realizados a la base de datos por medio de calquier sentencia del DDL o del DML (el bloque que ha sido cambiado, la posición de cambio y el nuevo valor). El uso del Redo Buffer es estrictamente secuencial, en tal sentido pueden entrelazarse cambios en los bloques de datos producidos por transacciones diferentes. El tamaño de este Buffer también puede ser configurado para mejorar el rendimiento de la instancia y de las aplicaciones que sobre ellas se ejecutan.

- **Shared Pool:** Esta estructura se utiliza durante el procesamiento de comandos. Su tamaño también es configurable. Contiene dos zonas específicas:

  - **Library Cache**, almacena la siguiente información relacionada con una instrucción de SQL:
    - Texto de la instrucción.
    - Árbol de parseo, es decir la versión compilada de la instrucción.

- Plan de Ejecución, es decir la secuencia de pasos a ser realizados para ejecutar la instrucción a bajo nivel de acuerdo con los resultados producidos por el optimizador de consultas.

Basándose en esta información, si una consulta es ejecutada nuevamente, y su información permanece todavía en el Library Cache, no será necesario compilar de nuevo la instrucción. En tal sentido este componente de la arquitectura permite mejorar el rendimiento de las aplicaciones que se ejecutan periódicamente.

- **Data Dictionary Cache**, también conocido como Dictionary Cache o Row Cache, almacena la información de uso más reciente sobre el diccionario de datos. Esta información incluye definición de columnas, usuarios, passwords y privilegios. Durante la fase de compilación, esta información es necesaria para resolver los nombres de los objetos utilizados en un comando SQL y para validar los privilegios de acceso.

- **Request Queue y Response Queue:** Estas estructuras constituyen los elementos básicos para el manejo de conexiones de usuarios al manejador. Por cada instancia de ORACLE existirá una Request Queue y varias Response Queues si está configurada en el modo servidor compartido (en caso contrario simplemente no existirán). Cuando un proceso usuario desea establecer conexión con el manejador, la solicitud de conexión será encolada en la Request Queue. Los procesos servidores tomarán las solicitudes de conexión, efectuarán las acciones necesarias para que la base de datos complete la solicitud y colocarán la petición en la Response Queue asociada al proceso que atendió la solicitud.

Los procesos en Oracle.-

Los procesos de ORACLE realizan funciones para los usuarios del manejador. En general se puede establecer una partición en los procesos de ORACLE:

- Procesos Servidores (o Shadow Proceses), los cuales ejecutan funciones de interacción con el servidor de datos basándose en las peticiones de los usuarios.

- Procesos Demonio (o Background Proceses), quienes realizan funciones para el RDBMS como un todo.

- Procesos de Red, encargados de proveer la interconexión entre procesos de usuario y el RDBMS o entre RDBMS que establecen un sistema de bases de datos distribuidas.

Procesos Servidores y el Program Global Area (PGA)

Los procesos servidores (Snnn) se comunican con los diferentes procesos de usuario e interactúan con ORACLE para satisfacer las peticiones. Por ejemplo, cuando un proceso de usuario solicita datos que no se encuentran en el SGA (en alguno de los buffer caches, dependiendo del tipo de operación y de la fase de su procesamiento), el proceso servidor que atiende la petición será el encargado de leer los bloques de datos y almacenarlos en el SGA (recuerde que el SGA en un área de memoria compartida).

Puede existir una correspondencia de uno a uno entre procesos de usuario y procesos servidor (por ejemplo en la configuración de RDBMS dedicado); aunque un proceso servidor puede conectarse a múltiples procesos de usuario (como en la configuración de RDBMS multi-threaded). Las conexiones admitidas dependerán en todo momento de la configuración de opciones del RDBMS realizadas durante la instalación del mismo.

El PGA constituye una región de memoria asociada a cada proceso servidor, la cual contiene datos e información de control para cada una de las sesiones que los usuarios mantienen con el RDBMS ORACLE a través de éste proceso servidor. Por lo tanto el PGA no es un área de memoria compartida. Una región de memoria para almacenar un PGA es solicitada cuando un proceso usuario establece una sesión de trabajo con el manejador de datos. El tipo de información que se almacena en el PGA depende de las opciones instaladas para el servidor ORACLE. Por ejemplo, cuando se utiliza una configuración de servidor dedicado, el PGA contiene los siguientes componentes:

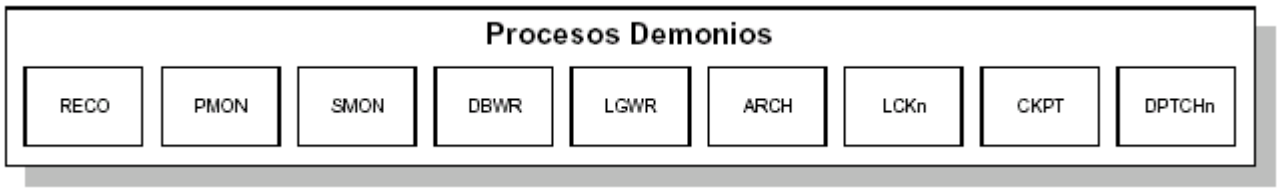
- **Sort Area**, que es utilizada para llevar a cabo los posibles ordenamientos de filas requeridos antes de que las filas sean procesadas o devueltas al usuario como resultado de una consulta.
- **Stack Space**, el cual contiene las variables de sesión de usuario y sus valores.

- **Cursor State**, el cual almacena el estado de los diferentes cursores que están siendo utilizados en la sesión del usuario.
- **Session Information**, la cual mantiene información sobre los privilegios que el usuario que ejecuta la sesión.

> Ejemplo: Cada vez que se invoca SQL*Plus, se crear un proceso usuario. Este proceso usuario se comunicará (bien sea por los mecanismos de IPC –en caso de que el servidor ORACLE y el proceso usuario estén en la misma máquina- o por mecanismos de software de comunicación en Red como SQL*Net –en caso de que el servidor ORACLE y el proceso usuario estén en máquinas diferentes) con el proceso servidor que le proveerá del acceso necesario al servidor ORACLE.

Procesos Demonios

Los procesos demonio, también conocidos como Background Proceses, constituyen programas que llevan a cabo funciones específicas de soporte y mantenimiento a la ejecución del servidor de bases de datos. Esto no quiere decir que sean opcionales, por el contrario sin ellos no se podría operar correctamente en un entorno basado en ORACLE. Estos procesos pueden ser observados en esta figura:



- **Database Writer (DBWR):** Encargado de copiar los bloques de datos desde el buffer cache a la memoria secundaria. Cuando una transacción cambia los datos de un bloque de datos, no es necesario que este bloque se escriba inmediatamente en el disco. Debido a esto, el DBWR puede escribir estos datos de forma que se logre mayor eficiencia de la obtenida si se escribiera siempre que una transacción terminara. Usualmente el DBWR escribe sólo cuando se necesitan nuevos bloques datos en el Database Buffer Cache. Los datos se escriben utilizando el algoritmo LRU (Least Recently Used). Si el entorno de memoria secundaria está basado en entrada salida asíncrona (AIO) debe existir un único proceso DBWR. Si el entorno no posee AIO, el rendimiento del sistema puede ser mejorado agregando más procesos DBWR.

- **Log Writer (LGWR):** Encargado de escribir las entradas desde el Log Buffer a disco. La escritura de bloques del Redo Log Buffer a disco ocurre secuencialmente y bajo las siguientes reglas:
  - Cuando el Redo Log Buffer está lleno en un 33% o más.
  - Cuando ocurre un timeout (cada 3 segundos).
  - Antes de que el DBWR escriba algún bloque modificado a disco.
  - Cuando una transacción se confirme (= "COMMIT").

- **Checkpoint (CKPT):** Encargado de notificar al DBWR, para que se escriban en los archivos de datos todos los bloques contenidos en la lista de sucios. Este proceso es invocado en intervalos de tiempo determinados. El CKPT es opcional. Si este proceso no está presente las funciones son realizadas por el LGWR.

- **System Monitor (SMON):** Este proceso es el encargado de hacer un porceso de recuperación rápida cada vez que una instancia es inicializada. Esta labor incluye limpieza de las estructuras de datos de soporte a la ejecución de consultas y llevar la base de datos a un estado estable previo a la ejecución de aquellas transacciones que no hayan culminado exitosamente. De igual forma, el proceso SMON se encarga de desfragmentar el espacio físico de almacenamiento uniendo bloques de datos libres en la memoria secundaria.

- **Process Monitor (PMON):** Es el proceso encargado de llevar la pista de los procesos de la base de datos y efectuar labores de limpieza (liberar los recursos y liberar bloques ocupados en los caches) si alguno de ellos termina prematuramente. En la opción de multi-threading también es el encargado de reiniciar cualquier proceso dispatcher en caso de fallas.

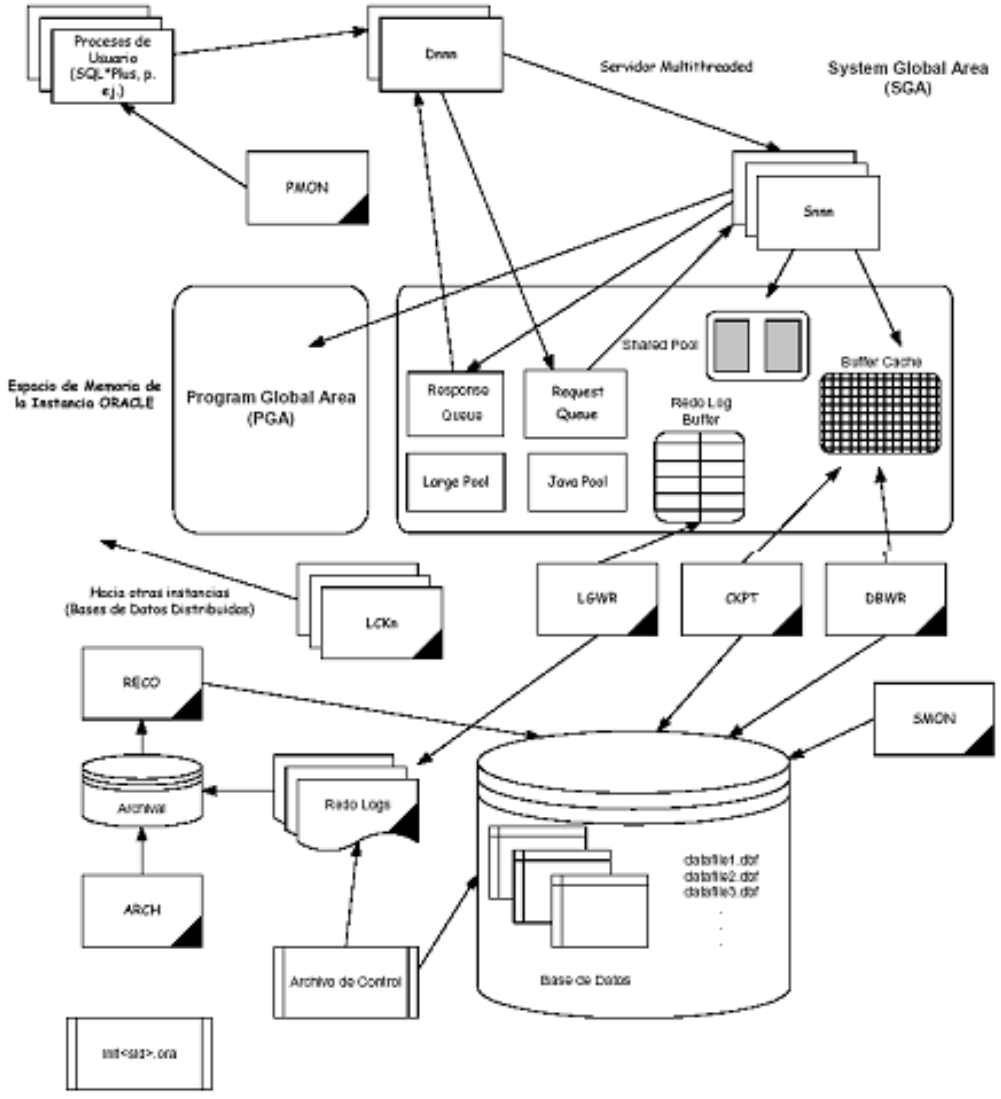
- **Archiver (ARCH):** Proceso encargado de copiar loss "redo log" activos en el sistema cuando estos se encuentran llenos. Este proceso se encuentra activo sólo cuando el RDBMS se encuentra operando en modo ARCHIVELOG, el único modo que admite recuperación de los datos frente a fallas del sistema.

- **Recoverer (RECO):** Encargado de resolver transacciones distribuídas que se encuentran pendientes debido a la red o a fallas ocurridas en la base de datos distribuída.

- **Dispatcher (Dnnn):** Un proceso dispatcher es creado por cada sesión de trabajo activa. Cada dispatcher es responsable de enrutar los requerimientos desde el proceso usuario, al cual se encuentra asociado, hacia los procesos servidores y retornar la respuesta al proceso de usuario adecuado. Los procesos Dnnn se crearán sólo en entornos donde el RDBMS se ejecute con la opción de multi-threading.

Control de Recursos

ORACLE administra el uso de los diferentes recursos que gestiona a través de "latches". Un "latch" es un cerrojo o semáforo mantenido por la instancia de ORACLE que permite administrar una cierta actividad. Los "latches" limitan la cantidad de tiempo y espacio en los que un proceso puede mantener un recurso en un instante dado. De igual forma, los "latches" son utilizados para garantizar el acceso exclusivo a un recurso. El monitoreo de "latches" permite determinar situaciones en la que se crea contención por el acceso a un recurso. El número de "latches" existentes depende de la configuración y la información de los "latches" definidos y manejados por la instancia puede ser extraída del diccionario de datos.



Estructura Física para el Manejo de Datos.-

Una base de datos ORACLE, identificada por un nombre lógico mantenido en la variable de entorno DB_NAME, representa las estructuras físicas y está compuesta de archivos del sistema de operación. Aunque es posible utilizar un nombre de base de datos diferente al del nombre de la instancia, se recomienda utilizar el mismo nombre para hacer más sencilla la administración.

Los archivos que conforman la base de datos contienen los datos del usuario e información adicional que es necesaria para garantizar el funcionamiento de la base de datos. Una base de datos ORACLE consiste de los siguientes tipos de archivos:

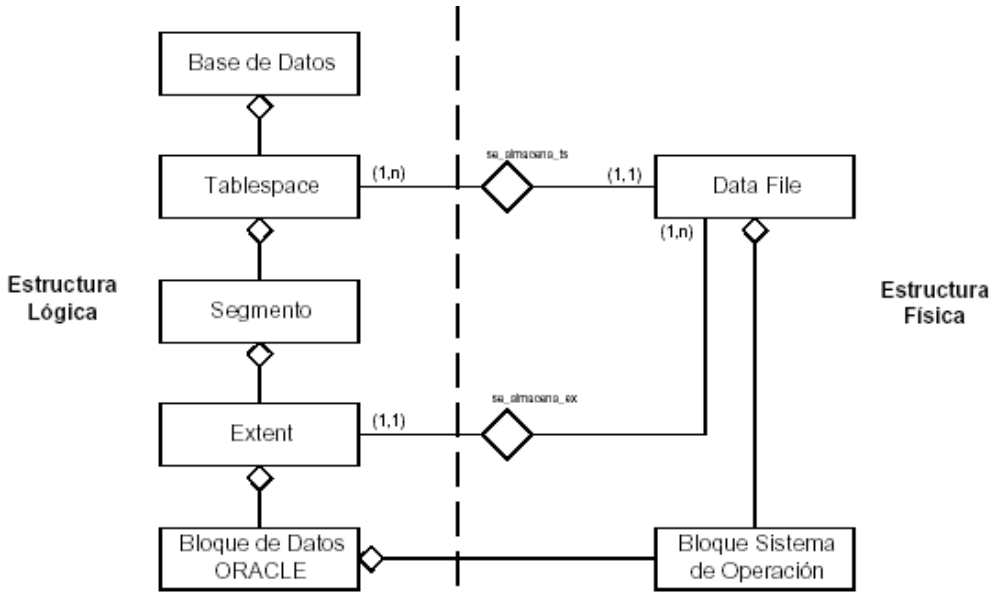
- **Archivos de datos (Data Files):** son aquellos que almacenan el diccionario de datos, los objetos del usuario, y los valores de los datos previos a su modificación. Una base de datos ORACLE debe poseer al menos un archivo de datos. La información de una tabla puede expandirse a muchos Archivos de datos y un Data File puede almacenar múltiples tablas. El número máximo de Archivos de datos que soporta una instancia de ORACLE puede ser configurado en el "Control File".

- **Archivos de bitácora (Redo Log Files o Redo Log):** Mantienen información sobre todos los cambios efectuados sobre la base de datos para asegurar su reconstrucción en caso de fallas. Toda base de datos ORACLE requiere de por lo menos dos Redo Log Files. Es esencial que los Redo Log posee un buen rendimiento y estén protegidos contra fallas de hardware. Si se pierde la información contenida en estos archivos será imposible la recuperación de la base de datos en caso de fallas del sistema. ORACLE provee mecanismos para almacenar múltiples copias de los Redo Logs.

- **Archivos de Control (Control Files):** Contienen la información necesaria para mantener y verificar la integridad de la base de datos, como la ubicación de los datos y los Redo Log. ORACLE requiere de esta información toda vez que se "arranque" la instancia del RDBMS. Toda base de datos ORACLE requiere de por lo menos un Control File. Es esencial que los Control Files estén protegidos ya que en caso de perdida no se podrá reiniciar la instancia de la base de datos con la consecuente pérdida de toda la información. ORACLE provee mecanismos para almacenar múltiples copias de los archivos de control.

Estructura Lógica de Almacenamiento.-

ORACLE es el encargado de manejar el espacio donde van a ser almacenados todos los objetos de una base de datos. Las unidades lógicas de almacenamiento son: bloques de datos, extents, segmentos y tablespaces. La siguiente figura muestra la relación existente entre estas estructuras de datos:



- **Tablespaces:** El nivel más alto en la estructura de almacenamiento de datos en ORACLE es la base de datos. Una base de datos (DATABASE) agrupa las piezas lógicas de mayor nivel de almacenamiento en ORACLE: los tablespaces. Un tablespace es utilizado para agrupar lógicamente los datos. Por ejemplo, puede tenerse un tablespace para almacenar los datos de contabilidad y otro para almacenar los datos de ventas en una empresa de servicios. Al segmentar grupos de datos relacionados en tablespaces diferentes se simplifican las tareas de administración de dichos grupos.

Los tablespaces pueden estar constituidos de varios archivos de datos. Al utilizar más de un archivo de datos por tablespace pueden distribuirse los datos sobre varios discos y balancear la carga de E/S, mejorando así el rendimiento del sistema. Como parte del proceso de crear la base de datos, ORACLE automáticamente crea un tablespace llamado SYSTEM. Aunque sólo una base de datos pequeña puede ser almacenada en SYSTEM, es recomendable que se cree uno (o varios) tablespace(s) para los datos del usuario. El tablespace SYSTEM almacena los datos del diccionario. Los Archivos de datos pueden ser archivos del sistema de operación, y en algunos sistemas de operación se permite que un archivo de datos sea un dispositivo de almacenamiento secundario crudo (o parte de él).

- **Segmentos:** Un segmento es un conjunto de extents que han sido asignados para el almacenamiento de un tipo de datos específico y todos ellos se encuentran ubicados dentro del mismo tablespace. Existen diferentes tipos de segmentos como lo son:

  - de datos: cada segmento de datos almacena los datos correspondientes a una tabla
  - de índice: cada segmento de índice mantiene los datos para un índice definido dentro de la base de datos
  - de Rollback: un segmento de Rollback permite almacenar las acciones de una transacción que deben ser deshechas bajo ciertas circunstancias
  - **temporales:** se crean cuando se requiere de un espacio temporal para procesar una instrucción de SQL y son destruidos una vez que haya culminado el procesamiento de la instrucción. Las instrucciones que hacen uso de estos segmentos son: `SELECT ... ORDER BY...,SELECT ... GROUP BY..., SELECT ... UNION ..., SELECT DISTINCT ..., SELECT … INSERSEC ..., SELECT ... MINUS ..., CREATE INDEX`

- **Extents:** Los extents son las piezas utilizadas para constituir segmentos. Cada extent se compone de una serie de bloques de datos. La razón principal de esta estructura es la de minimizar el espacio desperdiciado (vacío) de un tablespace. A medida que se insertan o eliminan filas de una tabla, los extents del tablespace asociado a la tabla pueden aumentar o disminuir de tamaño. De esta forma el espacio para el almacenamiento de los datos puede ser administrado dependiendo de cómo la tabla va sufriendo modificaciones en el número de filas. Cuando se crea un tablespace se puede especificar el número mínimo de extents a ser asignados, así como el número de extents a ser agregados cada vez que se agote el espacio disponible para almacenar datos.

- **Bloques de Datos:** Una base de datos se encuentra almacenada en bloques de datos que es el nivel más fino de unidades de almacenamiento. El tamaño de un bloque de datos siempre corresponde a un múltiplo del tamaño de bloque manejado por el sistema operativo. El tamaño del bloque de datos es un valor configurable en el RDBMS.

El Diccionario de Datos de ORACLE.-

El diccionario de datos es una parte fundamental de toda base de datos ORACLE. El diccionario de datos de ORACLE está constituido por una serie de tablas, vistas y ORACLE packages a los que puede accederse para obtener información sobre la base de datos. Las tablas del diccionario de datos son creadas automáticamente durante el proceso de instalación de ORACLE y permiten al administrador conocer, entre otros:

- La estructura lógica y física de la base de datos
- Los usuarios de la base de datos
- Las restricciones de integridad definidas sobre las tablas de la base de datos
- El espacio asociado a cada objeto en la base de datos y cuánto de este espacio está siendo realmente utilizado por los diferentes objetos creados por los usuarios de la base de datos

ORACLE predefine un usuario dueño de todo el diccionario de datos denominado SYS. Este usuario tiene todos los permisos sobre cualquier objeto de la base de datos (incluidos los objetos de cualquier usuario). Debido a que este usuario puede modificar entradas del diccionario de datos es recomendable no utilizarlo ya que cualquier error generado sobre el diccionario de datos puede provocar errores irrecuperables en el RDBMS. Sólo se deberá hacer uso cuando se requiera efectuar modificaciones CONTROLADAS sobre el diccionario para reparar errores en el mismo (cirugía de la base de datos). Sus componentes son:

- **Tablas Base:** La base del diccionario de datos es una serie de tablas a las que el servidor de datos accede cada vez que se procesa una instrucción del DDL de SQL, o en determinados comandos del DML. Estas tablas raramente son accedidas por usuarios ya que la mayor parte de su información está codificada. Por ejemplo, la tabla IND$ provee información sobre los índices definidos en la base de datos, y la tabla OBJ$ provee información sobre los diferentes objetos lógicos (tablas, índices, secuencias, etc.) creados por los usuarios de la base de datos.

- **Vistas Estáticas:** Las vistas estáticas se crean durante la instalación del RDBMS y decodifican y resumen la información contenida en las tablas base. Durante la ejecución del script de creación de estas vistas se generan sinónimos públicos para proveer el acceso a los usuarios del RDBMS. Estas vistas son las que deben ser utilizadas para efectuar labores de administración rutinarias que requieran de información específica sobre la configuración y estado de la base de datos. A estas vistas se les conoce como estáticas porque no mantienen información relacionada con las sesiones que actualmente existen en el sistema de base de datos.

  Estas vistas se dividen en tres categorías de acuerdo con el prefijo de sus nombres:

  - Vistas con prefijo USER: Estas vistas pueden ser utilizadas por cualquier usuario de la base de datos y se refieren a los objetos poseídos por dicho usuario. Por ejemplo si el usuario ci531301 ejecuta el comando SQL:

  ```
  SELECT * FROM USER_TABLES;
  ```

  Se desplegará toda la información de aquellas tablas cuyo dueño sea el usuario ci531301.

  - Vistas con prefijo ALL: Estas vistas pueden ser utilizadas por cualquier usuario e incluyen la columna OWNER adicionalmente a la demás información. Con estas vistas se puede tener acceso a la información de aquellos objetos de los que el usuario es dueño, más todos aquellos objetos públicos y todos aquellos objetos a los que el usuario tiene acceso por pertenecer a cierto grupo de seguridad o por poseer ciertos privilegios sobre la base de datos. Por ejemplo, si el usuario CI531301 posee el privilegio de ver la tabla EMPLEADO creada por el usuario CI531302 (dicho privilegio le fue otorgado por un administrador o por el mismo usuario CI531302), entonces la consulta anterior devolverá la misma información (incluyendo la columna OWNER) mas una fila específica con la información sobre la tabla EMPLEADO del usuario CI531302.

  - Vistas con prefijo DBA: Estas vistas dan información sobre TODOS los objetos de la base de datos, y usualmente incluyen la columna OWNER. Estas vistas sólo pueden ser utilizadas por el administrador de la base de datos o por cualquier usuario que posea el privilegio SELECT ANY TABLE o pertenezca a un rol que incluya tal privilegio.

- **Vistas Dinámicas**: A diferencia de las vistas estáticas, las vistas dinámicas (también conocidas como performance views) incluyen información sobre las condiciones actuales (en un instante dado) de operación

del RDBMS. La mayor parte de estas vistas son creadas durante la instalación y existen algunas que se crean explícitamente para monitorear cierta actividad (el RDBMS provee los archivos necesarios para crear estas vistas).

Todas las vistas dinámicas se identifican por poseer el prefijo V$. Por ejemplo la vista dinámica V$_SESSION incluye información sobre las sesiones actuales y la vista V$SYSSTAT provee información estadística sobre el RDBMS. Para obtener información general sobre las vistas del diccionario de datos puede usarse la consulta:

```
SELECT * FROM DICTIONARY WHERE table_name LIKE ´%indicador%´;
```

Por ejemplo, para ver todas las vistas relacionadas con tablas podríamos ejecutar la instrucción SQL:

```
SELECT * FROM DICTIONARY WHERE table_name LIKE ´%TABLE%´;
```

En el manual ORACLE9i Reference se dispone de toda la información detallada de las vistas estáticas y dinámicas que provee el RDBMS.

Entendiendo las conexiones vía servidor compartido versus servidor dedicado.-

Usar un servidor dedicado es de lejos el método más común para conectar a una base de datos Oracle. Es más fácil de configurar y ofrece muchos beneficios cuando se hace debug sobre un proceso. Un servidor compartido posee una configuración más compleja pero puede ayudar a poder aceptar más conexiones usando el mismo hardware.

Explicaré brevemente como trabajan los dos tipos de conexiones y los pros y contras de cada configuración. También rebatiré algunos mitos que existen sobre los servidores compartidos.

"¿Debería usar el servidor compartido de Oracle?" A menos que se tengan más conexiones concurrentes a la base de datos de las que el sistema operativo puede manejar no se debería usar el "shared server". Hasta que el servidor que aloja la base de datos no pueda aceptar una conexión más se debe usar el servidor dedicado. Sólo entonces, cuando el servidor está a su máximo rendimiento y capacidad, donde no es posible otra conexión o se está afectando al rendimiento general, sería el momento de considerar el cambio a un "shared server". Afortunadamente los clientes no tienen control sobre esto, por lo que deshacer los cambios llegado el caso es trivial y es invisible a la aplicación.

Actualmente hay muchas razones por las que los servidores dedicados deben ser la primera elección, pero la principal es que un sistema en donde un servidor dedicado todavía no lo ha sobrecargado con procesos/threads, será más rápido. Así que a menos que el servidor dedicado sea la causa de fallos de rendimiento (sobrecargas del sistema incluídas) no hay motivos técnicos para pensar que un servidor compartido lo hará mejor.

¿Cómo funcionan las conexiones a un servidor dedicado?.-

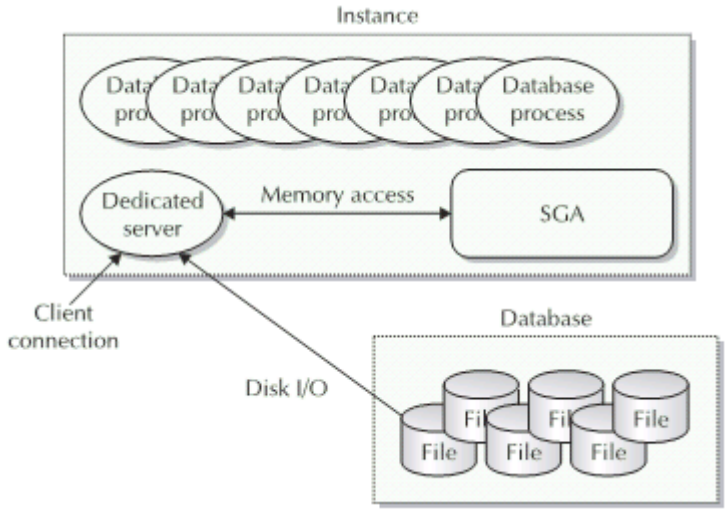
Cuando nos conectamos a una instancia Oracle en realidad nos estamos conectando a un grupo de procesos y memoria compartida. Ya hemos visto que la base de datos es un conjunto complejo de procesos que interactúan juntos para atender a las operaciones solicitadas.

Pasos en una conexión a un servidor dedicado

Cuando nos conectamos a un servidor dedicado a través de la red seguimos estos pasos:

1. Nuestro proceso de cliente se conecta por la red al listener que normalmente está en el servidor de base de datos. Hay situaciones (como en el Oracle Cluster o RAC) en donde el listener no estará en la misma máquina que la instancia de base de datos.
2. El listener crea un nuevo proceso dedicado (Unix) o solicita a la base de datos la creación de un nuevo thread dedicado (Windows) para nuestra conexión.
3. El listener traspasa la conexión al nuevo proceso o nuevo thread.
4. Nuestro cliente envía las peticiones al servidor dedicado el cual las procesa y nos retorna los resultados.

La siguiente figura muestra como funciona la configuración de servidor dedicado. La instancia Oracle será un único proceso con threads (Windows) o cada "burbuja" será un proceso físico separado (Unix). En una configuración de servidor dedicado cada cliente tiene su propio proceso asociado a él.



El servidor dedicado recibirá nuestra sentencia SQL y la ejecutará. Leerá los archivos de datos y pondrá datos en cache o mirará en la cache por si ya están los datos, ejecutará sentencias UPDATE y ejecutará nuestro código PL/SQL. Su único objetivo es responder a las llamadas SQL que le enviemos.

En el modo servidor dedicado hay un mapeo uno-a-uno entre procesos de la base de datos y procesos de cliente. Cada conexión tiene un servidor dedicado designado exclusivamente para ella. Esto no significa que cada sesión tenga un servidor dedicado (es posible para una aplicación tenga muchas conexiones activas a la base de datos). Sin embargo por normal general cuando se trabaja en modo servidor dedicado hay una relación uno-a-uno entre una sesión y un servidor dedicado.

Pros y contras de las conexiones dedicadas

A continuación expondré una pequeña lista de las ventajas de usar un servidor dedicado para conectar a la base de datos:

- es fácil configurarlo. De hecho no requiere ninguna configuración;
- es el modo más rápido de operar en la base de datos al usar el camino más corto;
- debido a que los archivos de traza están ligados a un proceso (por ejemplo a un servidor dedicado) las utilidades como SQL_TRACE se benefician de este modo. De hecho el intentar usar SQL_TRACE en una conexión a un servidor no dedicado será muchas veces una futilidad;
- todas las tareas administrativas están disponibles. (En el modo servidor compartido ciertas características de administración, como STARTUP, SHUTDOWN y recuperaciones, no están disponibles);
- la memoria para el Área Global de Usuario (UGA), que es una zona de memoria específica por sesión, está asignada en la PGA dinámicamente. No necesita ser configurada en la SGA.

Algunas de las desventajas de usar conexiones a servidores dedicados serían las siguientes:

- el cliente tiene un proceso/thread consumiendo recursos (memoria, CPU. etc.) del servidor desde que la sesión empieza hasta que termina;
- al aumentar el número de usuarios el sistema operativo puede sobrecargarse debido al manejo de sus procesos/threads;
- se tiene un control limitado sobre la cantidad de sesiones que pueden estar activas simultáneamente (aunque Oracle9i lo resuelve con el "Resource Manager");
- no se puede usar este tipo de conexiones para concentración de "database links";
- si se tiene un ratio muy alto de conexiones/desconexiones a los datos, los procesos de creación y destrucción de procesos/threads pueden ser molestos. Un "shared server" tendría ventaja aquí. Notad que la mayoría de aplicaciones basadas en web esto no es un problema porque normalmente realizan algún tipo de "connection pooling".

Como podéis ver con estas listas para la mayoría de sistemas los beneficios sobrepasan a las desventajas. En los servidores convencionales (máquinas con 2 CPU) se necesitarían de 200 a 400 conexiones concurrentes antes de considerar un tipo diferente de conexión. En máquinas con más potencia el límite es mucho más alto.

No hay un límite a partir del cual la configuración de servidor dedicado se vuelve inútil. He visto casos en que se dejó de usar esta configuración con 100 usuarios. Por otra parte he visto sistemas funcionando con más de 1.000 usuarios sin problemas. El hardware y el modo en que se use la base de datos dictaminará cuando se debe cambiar al modo "shared server". Mientras se tenga suficiente CPU y un poco de memoria RAM la configuración de servidor dedicado es la más fácil y funcional.

¿Cómo funcionan las conexiones a un servidor compartido?.-

El protocolo de conexión para una conexión a un servidor compartido es muy diferente del usado para una configuración de servidor dedicado. En el modo "shared server" no hay un mapeo uno-a-uno entre clientes (sesiones) y procesos/threads del servidor. En vez de eso hay una "piscina" (= pool) llamada "shared servers" que realiza las mismas operaciones que un servidor dedicado, pero las realizan para múltiples sesiones en vez de para una sesión sólo.

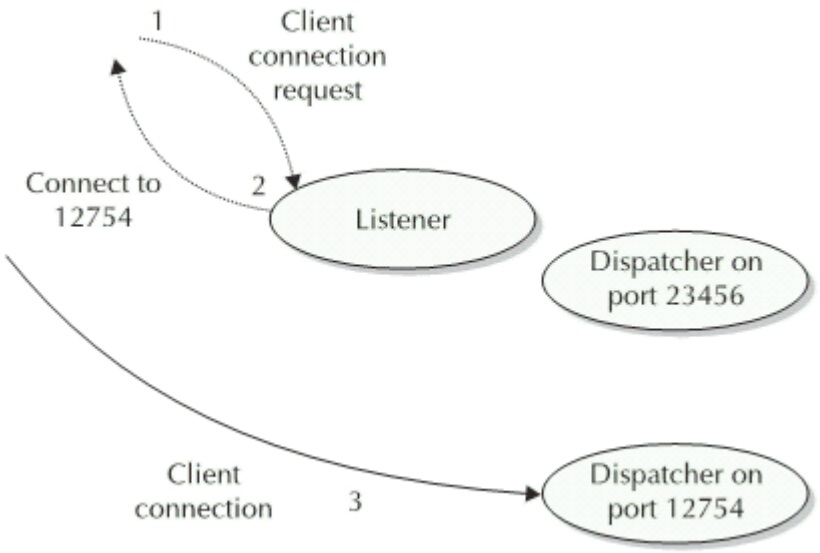
Adicionalmente hay procesos llamados "despachadores" (=dispatchers) que funcionan cuando se está usando una conexión a servidor compartido. Los clientes estarán conectados a un "despachador" durante toda la vida de su sesión y este "despachador" facilitará el manejo de sus peticiones hacia un "shared server" y les retornará la respuesta. Estos "dispatchers" funcionando en la base de datos son conocidos como procesos "listener". Cuando nos conectamos el "listener" nos redirigirá hacia el "dispatcher" disponible.

Pasos en una conexión a un servidor compartido

Cuando nos conectamos vía conexiones de servidor compartido se suceden estos pasos:

1. Vuestro proceso de cliente se conecta a través de la red al "listener" funcionando en la base de datos. Este "listener" elegirá un "dispatcher" desde la piscina de "dispatchers" diponibles para que os conectéis.
2. El "listener" os retornará la dirección del "dispatcher" elegido para que os conectéis a él, o también podéis ser redirigidos directamente al "dispatcher". Esto sucede a nivel TCP/IP y depende del sistema operativo. Si no hemos sido conectados directamente al "dispatcher" nuestro cliente se desconectará del "listener" y se conectará al "dispatcher".
3. Nuestro proceso de cliente envía su petición al "dispatcher".

La siguiente figura ilustra el proceso de conexión en el modo servidor compartido:



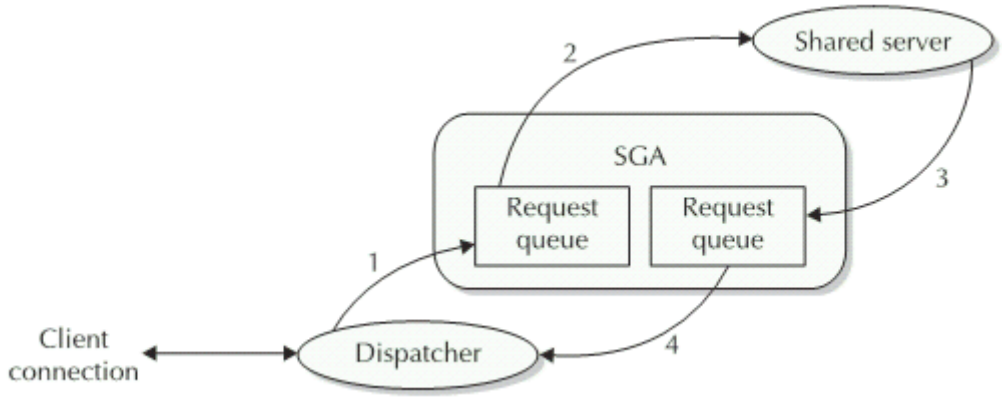
Así como las peticiones de conexión son recibidas, primero el "listener" elegirá que un "dispatcher" desde la "piscina" de "dispatchers" disponibles. Después el "listener" retornará al cliente la información precisa de cómo el cliente puede conectarse al proceso "dispatcher". Esto debe ser hecho porque el "listener" está funcionando en un host con "hostname" y puerto bien conocidos, pero los "dispatchers" aceptarán conexiones aleatoriamente en los puertos asignados en ese servidor. El "listener" es consciente de esas asignaciones aleatorias de puerto y por eso elige un "dispatcher" para nuestra conexión. El cliente se desconecta desde el "listener" y se conecta directamente al "dispatcher". Por último tenemos una conexión física a la base de datos.

Procesamiento de comandos en un servidor compartido

Ahora que estamos conectados el siguiente paso es el procesamiento de comando. Con un servidor dedicado únicamente tenemos que enviar la petición que sea (por ejemplo `SELECT * FROM emp`) directamente al servidor dedicado para su procesamiento. Con las conexiones a servidores compartidos esto es un poco más complejo. No tenemos un único servidor para nosotros. Los pasos son los siguientes:

1. el "dispatcher" coloca la petición en una cola común (compartida por todos los "dispatchers") de la SGA.
2. el primer "shared server" disponible coje y procesa esa petición.
3. el "shared server" coloca la respuesta a esa petición en su cola privada de respuestas de la SGA.
4. el "dispatcher" al que estamos conectados cojerá esa respuesta y nos la retornará.

La siguiente figura ilustra este proceso:



Es interesante notar aquí que estos pasos son realizados para cada llamada a la base de datos. Por lo tanto la petición de parsear una consulta puede ser procesada por el "shared server 1", la petición para recuperar la primera fila de datos puede ser procesada por el "shared server 2", la siguiente fila por el "shared server 3" y finalmente el "shared server 4" puede ser el que cierre el conjunto de resultados.

Pros y contras de las conexiones compartidas

Aquí hay algunas de las ventajas de usar una conexión de servidor compartido a la base de datos:

- Es tan escalable así como necesitemos aumentarlo. Una máquina que pueda físicamente manejar 100 conexiones de servidor dedicado será capaz de manejar 1.000 ó más conexiones a servidor compartido, dependiendo de la aplicación. Si tenemos una aplicación tradicional cliente/servidor donde la inmensa mayoría de las sesiones conectadas están inactivas (por ejemplo de 1.000 conexiones puede que sólo 50 ó 100 estén activas de forma contínua, y el resto estén esperando por las cavilaciones de los usuarios) la configuración de servidor compartido será útil al escalar la aplicación. Notad que este escalamiento se refiere sólo a la CPU (todavía necesitamos memoria suficiente para esas 1.000 conexiones).
- Permite un fino control sobre cuantas sesiones pueden estar activas de forma concurrente; impone un límite "duro" al número total de sesiones activas.
- Usa marginalmente menos memoria que la configuración de servidor dedicado. La diferencia se explica más adelante en esta sección.
- Permite que haya N+1 usuarios conectados cuando en un servidor dedicado sería imposible.
- Si tenemos un alto ratio de conexiones/desconexiones el uso de servidores compartidos puede ser más rápido que servidores dedicados. La creación y destrucción de procesos y threads es muy cara y puede sobrecargar al sistema.

Algunas de las desventajas de la configuración de servidor compartido son como las siguientes:

- Es más compleja de preparar que la configuración de servidor dedicado, aunque las herramientas GUI proporcionadas con la base de datos la ocultan generalmente. Muchos de los problemas de configuración que he visto ha sido por personas que editaban los archivos de configuración en vez de usar esas herramientas.

- En un sistema donde las conexiones de servidor dedicado trabajan bien, las conexiones de servidor compartido pueden ser más lentas. La ruta hacia un "shared server" es por definición más larga que la ruta hacia un servidor dedicado.

- Algunas de las características de la base de datos no están disponibles cuando nos conectamos vía "shared server". Debemos conectarnos a la base de datos usando un servidor dedicado para usar muchas de ellas. Por ejemplo usando conexiones "shared server" no podemos detener o iniciar la instancia, realizar recuperaciones o usar "Log Miner".

- La memoria total de todas las sesiones concurrentes UGA debe ser computada y configurada como parte del parámetro LARGE_POOL. Debemos establecerla adecuadamente. En Oracle9i ver.2 tenemos la posibilidad de ajsutar el tamaño LARGE_POOL de forma dinámica.

- El traceado (usando SQL_TRACE) no es algo aplicable de forma cómoda a las conexiones "shared server". Los archivos de traza son específicos de un proceso, y en un servidor compartido un proceso no es específico de una sesión. No sólo nuestras sentencias SQL serán escritas en diferentes trazas, sino que las sentencias SQL de otras sesiones serán escritas en las mismas trazas. Por lo tanto se deberá incluir alguna marca (o "hint") en las sentencias para reconocerlas fácilmente.

- Existe el peligro de "deadlocks" artificiales. Ocurren porque una vez que un "shared server" inicia el procesamiento de una petición no retornará nada hasta que haya acabado con ella. Por lo tanto si el "shared server" está enviando una petición UPDATE y la fila ya está bloqueada, se bloqueará. Supongamos que la sesión bloqueante estuvo sin hacer nada durante un período de tiempo (sin hacer ningún trabajo en la base de datos y no teniendo por tanto ningún "shared server" asignado) y el resto de los otros "shared servers" están bloqueados por esta sesión (o por cualquier otra), entonces cuando la sesión manteniendo el bloqueo intenta realizar un COMMIT o ROLLBACK (lo cual liberaría los bloqueos), se colgará porque no existen más "shared server" libres. Esta es una situación de "deadlock" artificial donde el bloqueante no puede liberar a los bloqueados porque éstos tienen copados al resto de servidores compartidos.

- Hay un riesgo potencial de monopolización del "shared server". Esto ocurre por la misma razón que los "deadlocks" artificiales. Si usamos un "shared server" con transacciones muy largas nuestras sesiones monopolizarán un recurso compartido, previniendo a los otros de usarlo. Si tenemos muchas de estas transacciones largas el rendimiento del sistema caerá de forma abismal porque las transacciones pequeñas deberán esperar a que las largas se completen.

Como podemos ver para la mayoría de usos la configuración de servidor compartido no es la más indicada. Sólo cuando la base de datos no pueda manejar físicamente la carga se debería considerar usar la configuración de servidor compartido.

Falsos mitos sobre los servidores dedicados

A través del sitio web "AskTom" y otros grupos me he dado cuenta de algunos mitos sobre la configuración "shared server". Estos mitos poco a poco van siendo echados abajo por las siguientes generaciones de programadores Oracle, pero todavía algunos quedan:

- **Los servidores compartidos usan muchísima menos memoria.** Este es uno de los motivos más extendidos para el uso de "shared server": reduce la cantidad de memoria. He visto gente que activa el "shared server", cambia su 'tnsnames.ora' para solicitar conexiones compartidas y no hace nada más (no configura memoria extra en la SGA). Estos sistemas caen cuando la SGA se queda sin memoria. Sus DBAs olvidaron que la UGA sería asignada al 100% y ahora tomará la memoria desde la SG en lugar desde un proceso local. Mientras la cantidad de memoria usada por los procesos del sistema operativo será significativamente reducida, la SGA deberá ser aumentada.

  Cuando activamos el "shared server" debemos ser capaces de determinar de forma precisa la necesidades estimadas de memoria UGA y asignar esa memoria en la SGA vía el parámetro LARGE_POOL. Esto derivará en que los requerimientos de SGA para "shared server" típicamente serán muy grandes. Esta memoria debe ser preasignada para que pueda ser usada sólo por la base de datos. Es cierto que en Oracle9i ver.2 ser puede ajustar dinámicamente "en línea" con el parámetro LARGE_POOL. Comparad con una configuración de servidor dedicado, conde cualquiera puede usar cualquier memoria no asignada a la SGA.

  Pero si la SGA es mucho más grande debido a que la UGA está localizada en ella, ¿de dónde viene el ahorro de memoria? Proviene del hecho de tener menos PGAs asignadas. Cada servidor dedicado/compartido tiene una PGA para información de proceso (áreas de ordenación, "hash areas" y otras). Son los requerimientos de memoria que se eliminan del sistema al usar el modo servidor compartido. Si vamos de usar 5.000 servidores dedicados a 100 servidores compartidos la suma de las 4.900 PGAs que no necesitamos es lo que ahorramos con los servidores compartidos. Sin embargo incluso esto es cuestionable desde que los modernos sistemas operativos emplean sofisticados mecanismos de paginamiento que desalojan la memoria que no está en uso por las 4.900 PGAs.

  La configuración "shared server" puede ahorrar memoria pero no tanta como mucha gente cree.

- **Se debe usar siempre el servidor compartido con servidores de aplicaciones.** Sólo si tenemos una granja inmensa de servidores de aplicación, donde cada servidor de aplicación tiene un gran "pool" de conexiones conectando a una única instancia de base de datos, debemos considerar usar la configuración "shared server". Aquí tenemos las mismas consideraciones que con otros sistemas (el servidor de aplicaciones no hace ninguna diferencia). Si tenemos un gran número de conexiones y estamos sobrecargando el servidor de base de datos, puede que haya llegado el momento de usar conexiones "shared server" para reducir el número de procesos físicos.

  > "Estamos usando connection pooling en la capa intermedia y servidor compartido en la capa de base de datos. El conjunto va lento, ¿por qué?" Bueno probablemente esté yendo lento porque se esté "cacheando la cache". Esto es como usar la cache de un sistema operativo y la cache de la base de datos. Si una cache es buena dos deberían ser mejor, ¿no? ¡Pues no! Si estamos usando una piscina de conexiones en la capa media las probabilidades indican que queremos usar un servidor dedicado por cuestiones de velocidad. Una piscina de conexiones asignará un número fijo y controlable de conexiones persistentes. En ese caso no tenemos una gran cantidad de sesiones y no necesitamos rápidos ratios de conexión/desconexión, lo cual es la receta que mejor guisan los servidores dedicados.

- **Los servidores compartidos sólo son para aplicaciones cliente/ servidor.** Aquí el DBA nunca consideraría usar "shared server" para un servidor de aplicaciones dado que él ya hace un "connection pooling". El problema con esta teoría es que el servidor de aplicación es sólo un cliente. Tenemos clientes (la piscina de conexiones del servidor de aplicaciones) y un servidor (la base de datos). Desde la perspectiva de la base de datos el paradigma de ir desde el modelo cliente/servidor al modelo de N-capas de aplicación nunca se había dado. Está claro que el programador las ve diferentes pero para la base de datos la diferencia no es visible: todo son conexiones.

  Debemos mirar el número de conexiones concurrentes a la base de datos. Si se han generado demasiados procesos de los que el sistema puede manejar las conexiones "shared server" serán la solución.