UNIVERSIDAD NACIONAL

AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

ACATLÁN

ADMINISTRACIÓN DE LAS BASES DE DATOS RELACIONALES Y MEJORAMIENTO DE SU DESEMPEÑO

TESINA

QUE PARA OBTENER EL TÍTULO DE

LIC. EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

PRESENTA

DARÍO JAIME LANDEROS FLORES

ASESOR: ING. MARÍA ANDREA SUÁREZ GARCÍA

Dedicatorias

A mis padres Jaime y Yola, que me han tenido la paciencia de darles la gratificación a su esfuerzo y dedicación para educarme y hacerme hombre de bien.

A mi esposa Patty y mis hijos Dieguito y Darío, mi motor mi vida y mi impulso a seguir siendo cada día mejor.

A mis hermanos Edgar, Erick y Aldo que siempre han estado conmigo en las buenas y en las malas.

A mi universidad, mi escuela y mi carrera que me dieron las bases de lo que hoy soy como profesional.

A mis amigos: Germán, Arturo, Angie que me toleraron y me han tolerado desde el primer día en la carrera hasta el día de hoy.

A mis profesores y asesora que durante mi vida de estudiante me regalaron el maravilloso mundo del conocimiento

Índice

Introducción.

Este trabajo tiene la intención de presentar bases sólidas teóricas para la administración de bases de datos. Hoy en día es muy importante tener en cuenta que todo sistema, cada aplicación requieren de una base de datos para explotar la información. La experiencia que he ido obteniendo a lo largo de estos años como DBA me ha indicado el rol tan relevante que tiene la teoría cuando de ellas se utiliza para finalmente poner en la práctica estos fundamentos.

Si bien cada manejador de bases de datos tiene características propias, cada una de las presentadas en este trabajo como Oracle, MySQL y SQL Server han sido muy importantes en el desarrollo de sistemas de información y han sido utilizadas para aplicaciones de diferentes propósitos como la minería de datos, aplicaciones científicas, de servicios, para sistemas corporativos donde los tiempos de respuesta son cada vez de mayor impacto si queremos mantenernos en la vanguardia tecnológica.

Las bases teóricas aquí presentadas sin duda servirán a quienes lean este trabajo y a quienes de manera especial se interesen en la administración de bases de datos. A lo largo de mi experiencia de trabajo con las bases de datos he sabido entender que los estándares de construcción o diseño en las bases de datos son de gran relevancia ya que desde la concepción misma de una base de datos si tenemos el cuidado de construir en base a ellos y a buenas prácticas de negocio nos ahorraran innumerables problemas en el ciclo de vida productivo.

En el capitulo 1 comentaré los detalles relacionados a la arquitectura de las bases de datos, comenzando a partir de las reglas relacionales definidas por Codd, pasando por los tipos de modelos desarrollados a lo largo de la historia para el diseño de las bases de datos y presentando las arquitecturas de cada uno de los RDBMS de los que hablo en este trabajo.

El capitulo 2 mostrará los aspectos importantes de la administración de bases de datos y tocará el tema del estándar OFA definido inicialmente para bases de datos Oracle y que sin embargo bien puede aplicar para la instalación de RDBMS como MySQL y SQL Server. Este estándar es básicamente una serie de lineamientos de construcción, instalación y diseño de bases de datos cuya aplicación práctica esta comprobada en los beneficios que proporcionan al DBA y que se reflejan con mayor provecho en bases de datos distribuidas o en construcción de bases de datos múltiples sobre un solo sistema operativo.

Dentro del capitulo 3 el cual esta enfocado a hablar de las estrategias de respaldo y recuperación en cuyo caso cuando una falla sucede estos lineamientos llegan a ser de tal relevancia que significan en su aplicación adecuada la continuidad del negocio. Muchas veces dependiendo del enfoque del negocio es prioritaria la facilitación del servicio sin dejar de lado que la perdida de información no es viable en ningún caso.

Finalmente la afinación de bases de datos tratada en el capitulo 4 nos apoyará a obtener los mejores beneficios de rendimiento de un sistema, ya que aunque si es de importancia tener recursos suficientes y holgados en un sistema, el tema de la afinación tanto de bases de datos como de aplicaciones nos llevará a tener los mejores resultados para el acceso a la información.

Como podrán observar los aspectos teóricos aquí expuestos serán de un uso cotidiano para quienes se dediquen a la administración de bases de datos donde invariablemente se encontrarán con nuevos retos conforme se encuentren en aplicativos que requieren ir evolucionando, ir creciendo y irse adecuando a las necesidades que vayan naciendo dentro de una empresa.

Para quienes lleguen a ejercer y trabajar dentro del mundo de la administración de bases de datos espero que este trabajo les sirva como una guía informativa y de aplicación práctica en el mundo de los sistemas de información.

Espero sinceramente que mi trabajo apoye y aporte un beneficio para su desempeño en el trabajo o simplemente como documento de estudio base.

Ahora bien, entremos de lleno a la lectura de este trabajo.

1 Conceptos generales y arquitectura de la base de datos.

1.1 Fundamentos de los RDBMS

En el panorama general de generaciones de bases de datos finalmente el esquema que ha logrado permanecer por más tiempo dentro del espectro empresarial es sin lugar a dudas el enfoque relacional. Este esquema ha ido evolucionando paulatinamente y en la actualidad se habla de bases de datos orientadas a objetos, sin embargo aún no han llegado a tener una gran popularidad e incluso hoy se manejan nuevas generaciones de tipo híbrido objeto-relacionales. Esto principalmente ha sido debido a la efectividad y seguridad que ofrece una base de datos relacional.

Si más preámbulos este capítulo hablará de los fundamentos sobre los cuales descansa el esquema relacional.

1.1.1 Reglas de Codd

Uno de los aspectos más importantes que debe considerarse en el concepto relacional, es una serie de reglas que definió el doctor Edgar F. Codd (1) y que es de vital importancia para que un DBMS (Sistema Manejador de Base de Datos) sea considerado como relacional. La ausencia de alguna de estas reglas es suficiente para considerar un DBMS como no relacional. Esto se detalla en la expresión que puede considerarse como la regla 0 de Codd: Para que un sistema se denomine sistema de gestión de bases de datos relacionales, este sistema debe usar (exclusivamente) sus capacidades relacionales para gestionar la base de datos.

Regla 1: Información

Toda la información en una base de datos relacional está representada explícitamente en un nivel lógico en forma única por los valores en las tablas.

Esto significa que toda la información debe estar contenida en tablas. Las tablas deben contener todos los datos de las aplicaciones y todos los datos relevantes del sistema y a su vez los datos están accesibles a todos los usuarios, sin embargo, por razones de seguridad, el acceso a ciertas tablas debe de restringirse. Como ejemplo tenemos los siguientes argumentos:

- Por tanto el diccionario y los catálogos, ambos metadatos (2); se representan exactamente igual que los datos de usuario.
- Puede usarse el mismo lenguaje para acceder a los datos y a los metadatos (regla 4).

- Un valor posible es el valor nulo, con sus dos interpretaciones:
    - Valor desconocido (ejemplo, una dirección desconocida).
    - Valor no aplicable (ejemplo, un empleado soltero no tiene esposa).

.

Regla 2: Garantía de acceso

Todos y cada uno de los datos en una base de datos relacional está accesible lógicamente para ser ordenada por medio de un nombre de tabla, valor de llave primaria y nombre de columna.

Una tabla es la estructura de datos que permite información en una base relacional y se compone de renglones y columnas. Esta garantía representa un esquema de direccionamiento asociativo que es único en el modelo relacional. Con el RDBMS no existen los punteros físicos creados para representar relaciones entre los datos. Las tablas y las operaciones a nivel tabla están bien definidas debido a que la teoría relacional está fundamentada en la teoría de conjuntos, el álgebra relacional y el cálculo relacional.

- Es importante que el concepto de clave primaria, que no es soportado en muchas implementaciones sea considerado sino como concepto si como una estructura similar considerando lo siguiente:
    - Hacer que los atributos clave primaria no puedan ser nulos (NOT NULL).
    - Crear un índice único sobre la clave primaria.
    - No eliminar nunca el índice.

Regla 3: Tratamiento sistemático de valores nulos

Los valores nulos (diferentes de una representación alfanumérica vacía o de una cadena de espacios en blanco, y diferente de cero en los números) son soportados completamente en el RDBMS para representar pérdida de información e información no aplicable.

- Existen tres valores de verdad: Verdadero, Falso y Desconocido (null). Se crean tablas de verdad para las operaciones lógicas:
    - null Y null = null
    - Verdadero Y null = null
    - Falso Y null = Falso
    - Verdadero O null = Verdadero

Regla 4: Catálogo dinámico en línea basado en el modelo relacional

La descripción de la base de datos relacional está representada por datos ordinarios, de tal manera que un usuario puede aplicar el mismo lenguaje relacional para consultar dicha descripción.

El diccionario de datos de Oracle contiene información importante con respecto al sistema en formato de tablas la cual se encuentra disponible a los usuarios autorizados a través de la utilización de sentencias de SQL.

- Esta regla es consecuencia de la regla 1 por lo tanto los metadatos también tienen una representación relacional.

Regla 5: Soporte para otros lenguajes

Un sistema relacional puede soportar varios lenguajes, entre los que se encuentran el lenguaje C, Fortran, Pascal, PL/1, COBOL, ADA y Java. Sin embargo, el SQL es el lenguaje que controla el acceso a la información de la base de datos el cual soporta las siguientes acciones sobre los datos: Definición y manipulación de datos, definición de vistas, constraints (3) de integridad, autorización y control de acceso, y control de transacciones.

Regla 6: Actualización de vistas

Una vista es una ventana lógica de una o más tablas. Todas las vistas que son actualizables también lo son a nivel sistema, no almacena datos, sin embargo despliega datos en formato de tablas ya que la definición de la vista se guarda en el diccionario de datos.

Regla 7: Insert, Update y Delete de alto nivel

La capacidad de establecer una relación base o una relación derivada como un solo operando es aplicada no sólo para recuperación de datos, sino también para inserciones (insert), borrados (delete) y actualizaciones (update) de información.

SQL es un lenguaje no procedural debido a que los procesos en conjunto son registrados como si se tratara de un proceso a la vez y proporciona navegación automática hacia los datos. La utilización de SQL permite a los usuarios trabajar con un nivel alto de estructuras de datos. Con SQL se pueden efectuar una gran variedad de tareas que incluyen: consulta de datos, inserción, actualización y borrado de renglones en una tabla; creación, modificación y borrado de objetos de la base de datos; control del acceso a la base de datos y sus objetos, y la garantía de consistencia en la base de datos. Anteriormente estas actividades eran manejadas por el DBMS por lenguajes diferentes para cada actividad mencionada.

Regla 8: Independencia física de datos

Los programas de aplicación y otras actividades terminales permanecen incomunicados mientras los cambios son efectuados por vía de los métodos de acceso o para propósitos de almacenamiento.

El DBMS debe separar claramente los aspectos lógicos y semánticos en la ejecución física de las tablas base. Cabe hacer notar que las aplicaciones no ingresan directamente a los archivos físicos de la base de datos.

Regla 9: Independencia lógica de los datos

Los programas de aplicación y actividades terminales permanecen aislados cuando se trata de preservar los cambios en la información de manera tal que permiten reflejar estos cambios en las tablas base.

- Un ejemplo de cambios que preservan la información es:
    - Añadir un atributo a una tabla base.
    - Sustituir dos tablas base por la unión de las mismas.

Regla 10: Independencia de integridad

Los constraints de integridad específicos a una base de datos relacional deben estar definidos en un lenguaje relacional y deben de ser almacenados en el catálogo de la base de datos (no en los programas de aplicación).

- Como parte de los limitantes inherentes al modelo relacional (forman parte de su definición) están:
    - Una base de datos relacional tiene integridad de entidad. Es decir, toda tabla debe tener una clave primaria.
    - Una BDR tiene integridad referencial. Es decir, toda clave externa no nula debe existir en la relación donde es primaria.

Regla 11: Independencia de distribución

El DBMS debe tener un lenguaje que habilita a los programas de aplicación y actividades terminales para que permanezcan separados cuando la distribución de los datos se introduce primeramente para después redistribuir los datos.

- Las mismas sentencias y comandos son ejecutados en una base de datos local o en una distribuida o remota.

Regla 12: No subversión

Si un sistema relacional tiene un lenguaje de bajo nivel no puede ser utilizado para manejar las reglas de integridad y los constraints expresados en un lenguaje relacional de alto nivel.

- Normalmente se utiliza SQL inmerso en un lenguaje anfitrión para solucionar estos problemas. Se utiliza el concepto de cursor para tratar individualmente las tuplas (4) de una relación. En cualquier caso no debe ser posible saltarse los limitantes de integridad impuestos al tratar las tuplas a ese nivel.

1.1.2 Evolución del RDBMS Oracle

La evolución de las bases de datos se concentra principalmente desde esquemas en donde Oracle surge a finales de los 70 bajo el nombre de “Relational Software” cuya investigación estuvo a cargo de George Koch.

Oracle ha presentado ya varias generaciones con importantes avances en el manejo de información, los cuales se expresan brevemente a continuación.

- *Oracle 5 y Oracle 6*: Fueron las dos primeras versiones de Oracle, quedando aun rezagadas por las versiones sucesoras y en donde por primera vez se agregó al núcleo la funcionalidad de integridad de la información a través de constraints.

- *Oracle 7*: Se fortalece la interfase de conectividad cliente servidor y se logra integrar el concepto de Oracle Parallel Server el cual ofrece a los usuarios un método seguro de alta disponibilidad y escalabilidad con lo que reincrementa el desempeño de sus bases de datos existentes introduciendo operaciones en paralelo y sincrónicas dentro de sus ambientes informáticos.

- *Oracle 8*: Incluye mejoras de rendimiento y de utilización de recursos. Independiente de que se necesite dar soporte a decenas de miles de usuarios y cientos de terabytes de datos, o se disponga de un sistema mucho más pequeño, pero igualmente crítico, todos se benefician del rendimiento de Oracle8. Esta versión soporta aplicaciones de procesamiento de transacciones en línea (OLTP) y de Data Warehouse (DWH) mayores y más exigentes manejando información a través de dimensiones múltiples. Se utilizan por primera vez tipos de datos más complejos para almacenar capacidades de información superiores a los 2 Gb.

- *Oracle 8i*: Esta versión fue diseñada para soportar grandes requerimientos de aplicaciones OLTP y de aplicaciones con alta concurrencia y uso de Internet así como aplicaciones en ambientes de Data Warehouse. El soporte de Java es otra de las características que fueron reforzadas para ofrecer un buen desempeño sin importar si la aplicación java reside en el mismo servidor de bases de datos, en un cliente o en algún otro servidor de aplicaciones.

- *Oracle9i*: Además de que Oracle 9i continua ofreciendo las mismas prestaciones que su versión antecesora se han agregado nuevas facilidades para hacer más robusto el RDBMS entre las que destacan una integración hacia el sistema operativo de Windows 2000, manejo de estructuras para e-business y módulos para inteligencia de negocios.

- *Oracle 10g*: Esta nueva versión ha sido preparada para desempeñar el papel de servidor de aplicaciones en el cual se facilita la implementación de aplicaciones en un solo ambiente que integra un RDBMS que soporte estos desarrollos sin mayor complejidad. Asimismo esta base de datos soporta los estándares de .NET de Microsoft.

1.2 El concepto relacional en las bases de datos

1.2.1 Panorama general de las bases de datos.

Hace algún tiempo, conforme las necesidades empresariales se fueron haciendo más y más críticas y con base en que el manejo de información fue incrementándose, se vio la necesidad de crear DBMS cada vez más estables y que proporcionaran soluciones en la manipulación de grandes volúmenes de datos. Independientemente del tipo de modelo de base de datos, sea este jerárquico, de red o relacional, el principal objetivo siempre ha sido satisfacer las necesidades de las grandes empresas. Dichos DBMS fueron perfeccionándose integrando herramientas para desarrollo de aplicaciones propias de la base de datos y herramientas de administración y control, explotando con eso las características ofrecidas por estas bases de datos.

Ahora bien, todo cambio ha sido benéfico y cada día podemos contar con mejores productos. Más aún, el avance mostrado no cesará debido a que continuamente surgen nuevas tecnologías para el manejo de los grandes sistemas de flujo de información. Una base de datos es un conjunto de archivos que almacenan información de manera ordenada, dicha información está disponible en cualquier momento en que sea solicitada.

En la figura 1 se muestran los componentes integrales de una base de datos: la información, el equipo, los programas o aplicaciones y los usuarios.

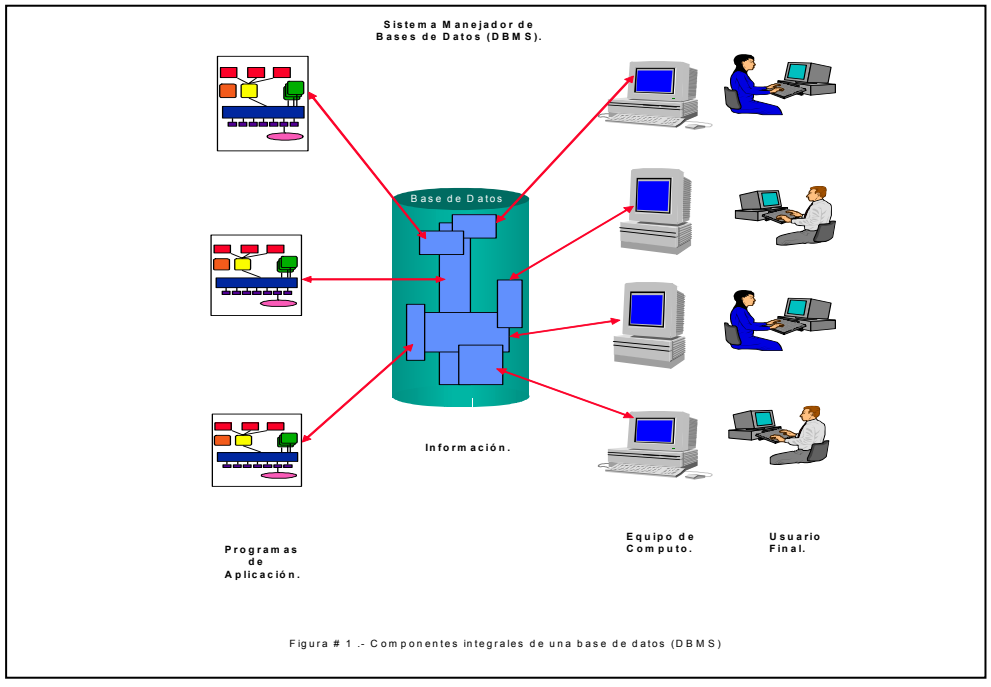


Figura # 1 .- Componentes integrales de una base de datos (DBMS)

GÓMEZ, G. Libro electrónico de bases de datos. Material de libre exposición en internet [en línea]. Medellín, Colombia: Instituto Tecnológico Metropolitano, 2009 [consultado 3 Noviembre 2009]. Disponible en internet: <http://basedatos.site50.net/>.

La información: Actualmente existen bases de datos para máquinas pequeñas como puede ser una PC, hasta bases de datos que se operan a través de grandes redes de computadoras, mainframes u otras microcomputadoras. Dependiendo del tipo de máquina, el flujo de la información es a través de un usuario o a través de un ambiente multiusuario,

donde varios usuarios pueden tener acceso a la misma base de datos en cualquier instante. Uno de los objetivos del ambiente multiusuario es lograr que los usuarios que accedan la base de datos al mismo tiempo, trabajen como si estuvieran en un ambiente monousuario en el cual todos los procesos compartidos se ejecutan de manera transparente, es decir, sin que el usuario se dé cuenta de que está compartiendo los recursos del sistema.

En general, la información en la base de datos estará integrada y será compartida; está integrada porque la base de datos puede considerarse como una unificación de varios archivos de datos que son diferentes, eliminando de esta forma la redundancia entre ellos. Además es compartida, porque los elementos individuales de información en la base de datos pueden ser ocupados por varios usuarios diferentes, en el sentido de que todos ellos pueden tener acceso al mismo elemento de información de manera concurrente. Esta capacidad de compartir se desprende en parte de la integración de la base de datos.

El equipo: Los componentes del equipo que están directamente relacionados con un sistema de base de datos son:

Los dispositivos de almacenamiento secundario: generalmente se utilizan discos magnéticos o unidades de cinta, en los cuales se almacenan los datos.

Los procesadores y la memoria principal: Hacen posible la ejecución de los programas o aplicaciones relacionados con la base de datos, incluso aquellos independientes del sistema.

Los programas de aplicación: En la relación que se establece entre la base de datos y los usuarios se localiza un grupo de programas o aplicaciones que son creados para explotar la información que se encuentra concentrada en la base de datos, también existe un sistema manejador de la base de datos (DBMS), el cual administra todas las peticiones de acceso a la base de datos solicitadas por los usuarios

El usuario final: En este caso, se tienen considerados tres tipos:

- *Desarrollador o programador de aplicaciones:* se encarga de escribir programas de aplicación utilizados en la base de datos, la mayoría de las ocasiones en lenguajes como C, Pascal, Cobol, Visual Basic, PL/SQL, etc.

- *Usuario final:* Es quien interactúa con el sistema a través de alguna terminal. Un usuario final puede tener acceso a la base de datos a través de una de las aplicaciones, o puede ocupar alguna interfaz incluida como parte integral de los programas de la base de datos.

- *Administrador de la base de datos (DBA):* Es la persona o conjunto de personas que se encargan de la administración de la base de datos y sus recursos. Este tipo de usuario y sus actividades serán analizados en el capítulo 2.

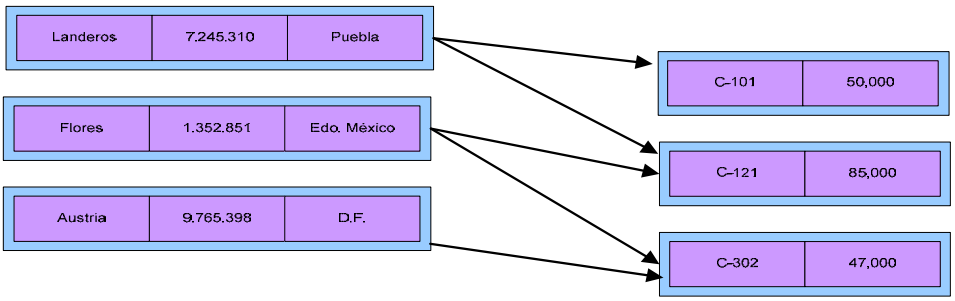
1.2.2 Modelo jerárquico y de red.

Una base de datos jerárquica se compone de un conjunto ordenado de árboles, es decir, un conjunto ordenado que se encuentra formado por múltiples ocurrencias de un solo tipo de árbol. Un tipo de árbol consiste en un solo tipo de registro “raíz” junto con un conjunto ordenado de cero o más tipos de subárbol dependiente (de nivel más bajo). Un tipo subárbol a su vez consiste en un solo tipo de registro junto con un conjunto ordenado de cero o más

tipos de subárbol dependientes de nivel más bajo y así sucesivamente. Por lo tanto el tipo de árbol completo es un arreglo jerárquico de tipos de registros. Este tipo de estructura presenta una relación de: Un registro hijo tiene un solo registro padre.

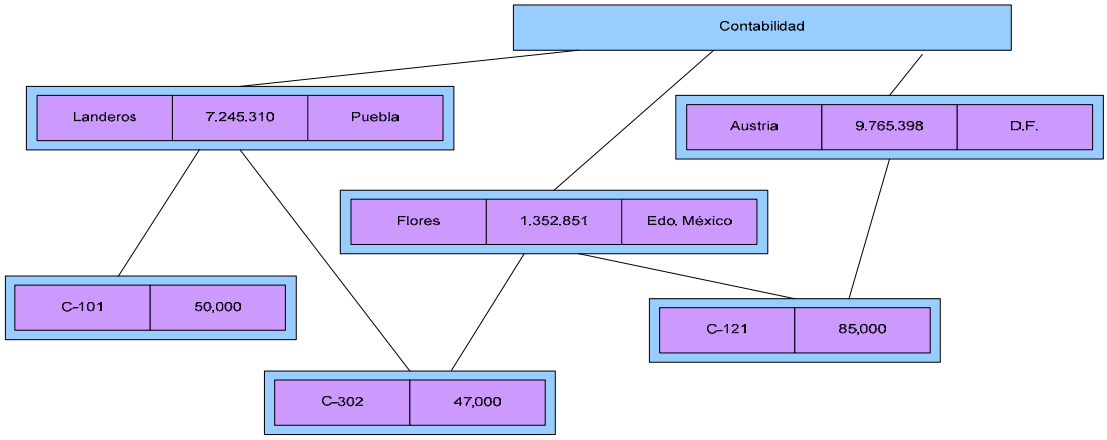
El modelo de red de una base de datos puede considerarse como una forma extendida de la estructura jerárquica. La estructura de red tiene una relación en la cual un registro hijo puede tener múltiples padres, inclusive ninguno. Una base de datos en red se compone de un conjunto de ligas, es decir, un conjunto de ocurrencias múltiples de cada uno de varios tipos de registros, junto con un conjunto de ocurrencias múltiples de cada uno de varios tipos de ligas. Cada liga implica dos tipos de registro, un tipo de registro padre y un tipo de registro hijo. Cada ocurrencia de un tipo de liga dado consiste en una sola ocurrencia del tipo de registro padre, junto con un conjunto ordenado de múltiples ocurrencias del tipo de registro hijo. En el diagrama de abajo se muestra una representación de los modelos de red y jerárquico.

Modelo de Red



Colecciones de registros y las relaciones entre datos se representan mediante enlaces (dirigidos)

Modelo Jerárquico



Los registros se organizan como colecciones de árboles, en lugar de grafos dirigidos

LANDEROS, Darío. Tesina de Administración de las bases de datos relacionales y mejoramiento de su desempeño. México: UNAM FES Acatlán, 2010.

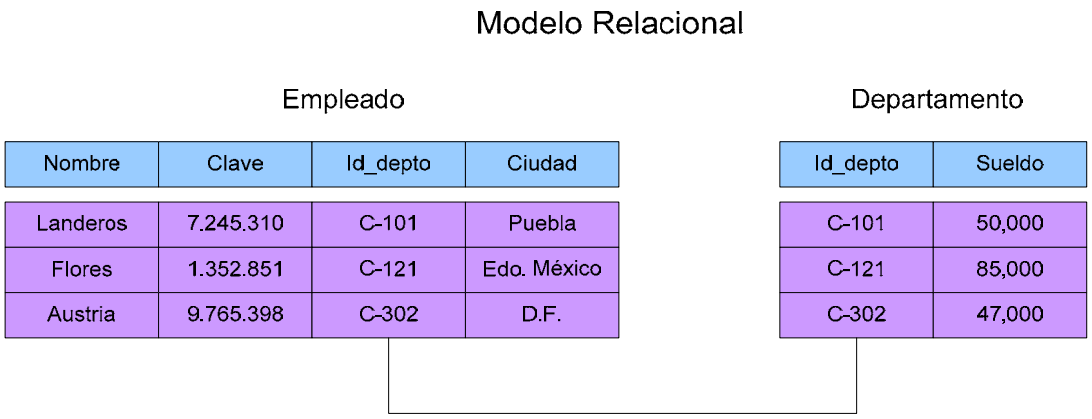
1.2.3 El modelo relacional y sus ventajas.

Una base de datos relacional soporta este modelo para su diseño en el RDBMS. Este clasifica a todos los elementos de un sistema sean estos una entidad (una persona, un lugar o una cosa) o una relación entre entidades.

La aplicación del modelo entidad-relación requiere de los siguientes pasos:

- Identificar las entidades del sistema y construir una tabla para representar la entidad.
- Identificar las relaciones entre las entidades y de esa forma extender la tabla creada o crear nuevas tablas para representar estas relaciones.
- Identificar los atributos de cada entidad y extender las tablas para incluir estos atributos.

Cuando se modela un sistema con el modelo entidad-relación con frecuencia ser incluirá un paso llamado normalización. La normalización es el proceso para asegurar que una tabla tenga una llave primaria completa y que todas las columnas que no son llaves primarias dependan de esa llave primaria.



Representa la información mediante tablas relacionadas entre sí por columnas comunes.

LANDEROS, Darío. Tesina de Administración de las bases de datos relacionales y mejoramiento de su desempeño. México: UNAM FES Acatlán, 2010.

1.2.4 Los modelos orientados a objetos.

Este modelo de bases de datos no acepta accesos a la información sino es a través de los métodos almacenados en la base de datos conservando así los principios de objetos. Estos métodos están listos para entrar en acción al momento de que reciben una solicitud y es entonces cuando los objetos quedan encapsulados.

El objetivo principal es el encapsulado, que permite almacenar datos y métodos. Los datos sólo pueden utilizarse mediante los métodos, los datos están diseñados para ser utilizados por esos métodos

Los objetos son activos, las solicitudes hacen que ejecuten sus métodos, algunos pueden ser complejos como aquellos que utilizan un motor de inferencias.

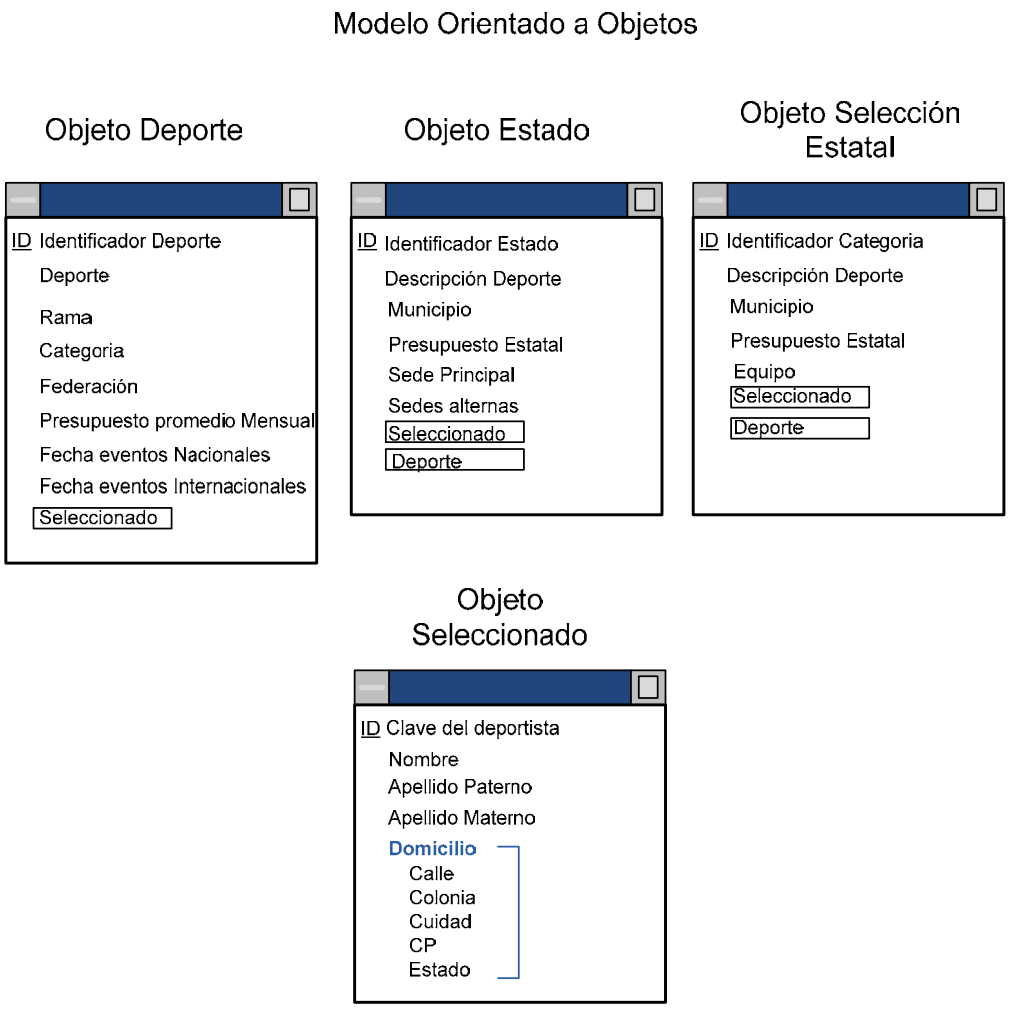
Las clases son diseñadas para una alta utilización y son rara vez modificadas, pudiendo ser reorganizadas sin modificar su forma de uso.

Las estructuras de datos son complejas, esto es transparente al usuario, debido a que estos están encapsulados.

Los datos están ligados entre si, de modo que los métodos logren un mejor entendimiento.

No se busca obtener datos no redundantes, sino métodos no redundantes utilizando el encapsulado y la herencia.

Las solicitudes al objeto provocan la ejecución de sus métodos.



LANDEROS, Darío. Tesina de Administración de las bases de datos relacionales y mejoramiento de su desempeño. México: UNAM FES Acatlán, 2010.

1.3 Arquitectura de una base de datos Oracle

Una base de datos Oracle se compone de dos partes importantes: La instancia de Oracle y los archivos de la base de datos cuyos conceptos son explicados a continuación para identificar las diferencias entre ellos.

1.3.1 Arquitectura física de la base de datos.

La arquitectura física de la base de datos la conforman todos los archivos físicos en el sistema operativo de la máquina, es decir, aquellos conocidos como: Archivos de datos (Data files), archivos de Redo log (Redo log files), Archivos de Control (Control files) y el archivo de Parámetros (Parameter file).

Archivos de datos (Datafiles). Contienen todos los datos almacenados, como pueden ser tablas, vistas, índices, procedimientos, funciones, etc. La información en los Datafiles es leída cuando es requerida durante operaciones normales de la base de datos y posteriormente es colocada en la memoria del RDBMS.

Archivos de redo log (redo log files). Son aquellos archivos utilizados para propósitos de recuperación de información en caso de alguna falla en el sistema o de dispositivos, también almacenan todas las transacciones y cambios efectuados a la información de la base de datos. En la base de datos son requeridos por lo menos dos archivos de redo log que pueden ser colocados en forma multiplexada (5), es decir, dos o más copias de redo log en discos diferentes.

Archivos de control (Control Files). Registran la estructura física de la base de datos al igual que un número de sincronización para cada archivo perteneciente a la misma con el fin de mantener la consistencia e integridad de la información. Al igual que los archivos de Redo log también pueden ser colocados en forma multiplexada.

Archivo de parámetros (Parameter File). Determina las características de la instancia, es decir todos los procesos que se van a levantar cuando damos de alta la base de datos. Comúnmente es denominado como init<NOMBRE_DE_LA_INSTANCIA>.ora.

La figura 2 nos muestra la estructura íntegra de la base de datos de Oracle. Esta estructura debe conocerse perfectamente para actividades de administración.

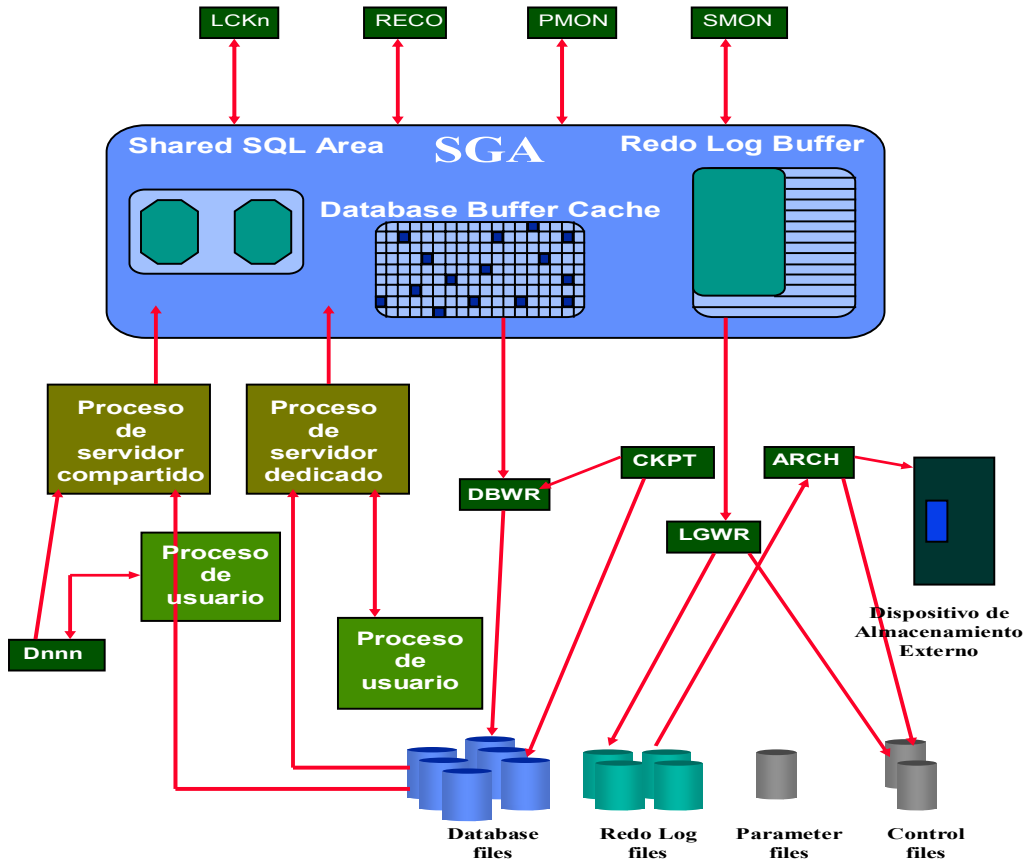


Fig. # 2.- Base de datos Oracle (Estructura física, estructura lógica y procesos de Oracle).

LEVERENZ Lefty, REHFIELD Diana, BAIRD Cathy. Oracle 8i Concepts Release 2 (8.1.6). U.S. December 1999.

1.3.2 Arquitectura lógica de la base de datos.

La instancia de Oracle esta compuesta de un área de memoria conocida como SGA y una serie de procesos denominados procesos de background. Los mecanismos de ejecución del RDBMS de Oracle se realizan mediante el uso de estructuras de memoria y procesos. Todas estas estructuras existen dentro de la memoria real o física del sistema de cómputo que constituye el sistema de la base de datos. Los procesos son tareas o trabajos que se ejecutan en la memoria de estos sistemas. (Ver figura # 2).

El RDBMS utiliza la memoria para almacenar código de programas y datos que son compartidos entre los usuarios de la base de datos. Existen varias estructuras de memoria básicas asociadas con el RDBMS: el área global del sistema (SGA) que incluye los redo log buffers, los database buffers y el área de memoria compartida (shared pool area) y el área global de programas (PGA).

El **SGA** es un conjunto de asignaciones o localizaciones de memoria asignado por el RDBMS que contiene datos e información de control para la instancia de la base de datos. El SGA y la serie de procesos conocidos como procesos de background conforman lo que se conoce como una instancia Oracle. Cuando una instancia es dada de alta se le asigna el

SGA y cuando es dada de baja también se da de baja el SGA. Cada instancia que se levanta consta de su propia SGA.

Los datos en el SGA son compartidos por los usuarios que están conectados a la base de datos. Para un desempeño óptimo, la SGA debe de ser lo suficientemente grande para almacenar los datos solicitados con mayor frecuencia minimizando de esta forma la lectura y escritura a disco. La información almacenada dentro de la SGA esta dividida en varios tipos de estructuras de memoria que incluyen los database buffers, redo log buffers y el shared pool area.

Los **database buffers** almacenan los bloques de datos utilizados más recientemente; estos bloques pueden contener información que no ha sido almacenada permanentemente en disco. Cuando una instancia es creada se le asigna un número estático de bloques que se especifican cuando se da de alta la instancia mediante el archivo de parámetros.

El **redo log buffer** almacenan los cambios hechos a la base de datos. Las entradas almacenadas en los redo log buffers son escritas en línea a los redo log files, que son utilizados para propósitos de recuperación cuando es necesario. Al igual que los database buffers, el redo log buffer es creado cuando se levanta la instancia y tiene un tamaño estático determinado en el archivo de parámetros de la base de datos.

El **shared pool area** contiene construcciones de memoria tales como áreas de código de SQL compartido: Una parte del SQL area es requerido para procesar cada sentencia SQL utilizada en la base de datos. El SQL area contiene el plan de ejecución correspondiente a cada sentencia. Una sola parte del SQL area es utilizada por múltiples aplicaciones que utilizan la misma sentencia.

El **program global area (PGA)** es un buffer de memoria que contiene datos e información para el control de los procesos de servidor. Un PGA es creado por el RDBMS cuando se inicia un proceso de servidor. La información en el PGA depende de la configuración de Oracle.

1.3.3 Procesos de Oracle.

Los procesos de Oracle son mecanismos de control en el sistema operativo de la máquina que ejecutan una serie de pasos comúnmente conocidos como trabajos o tareas. Un proceso normalmente tiene asignado su propia área de memoria cuando se está ejecutando. Un sistema de base de datos Oracle tiene dos tipos generales de procesos: Los procesos de usuario y los procesos del RDBMS. (Ver figura # 2 para referencia).

Los **procesos de usuario** son creados y establecidos para ejecutar el código de un programa de aplicación (tales como programas en Pro*C) o herramientas de Oracle como (SQL*DBA, SQL, Oracle Enterprise Manager -herramientas de administración de base de datos-). Los procesos de usuario también manejan la comunicación con los procesos de servidor.

Los **procesos del RDBMS** son llamados por otros procesos para ejecutar funciones que involucren otros más.

Los **procesos de servidor** son procesos del RDBMS para atender las peticiones de los usuarios conectados a la base de datos. Un proceso de servidor se encarga de comunicarse con los procesos de usuario y de interactuar con el RDBMS para llevar a cabo las peticiones asociadas a los procesos de usuario. Oracle puede ser configurado para variar el número de

procesos de usuario por proceso de servidor. En la configuración de servidor dedicado, un proceso de servidor atiende las peticiones para un solo proceso de usuario. Alternativamente, la configuración de multi-threaded server permite que varios procesos de usuario compartan un número pequeño de procesos de servidor disminuyendo de esta manera la utilización de los recursos disponibles en el sistema.

Los procesos de background se encargan de diferentes tareas en la base de datos para mejorar lo más posible el desempeño de la misma. Estos procesos se habilitan en forma asíncrona cuando son requeridos ejecutando lecturas, escrituras y monitoreo de otros procesos de Oracle obteniendo de esta forma la utilización de tareas en paralelo mejorando el rendimiento de la base de datos. Los procesos de background que son utilizados en la base de datos son los siguientes:

Database Writer (DBWR): Escribe los bloques ocupados desde el database buffer cache hacia los datafiles. El DBWR no necesita escribir los datos a los que se les ha dado commit (almacenar en la base de datos para hacer los cambios permanentes), de tal manera que puede escribir sólo aquellos que han sido menos requeridos, minimizando de esta manera las escrituras a disco y manteniendo el mayor tiempo posible los datos en memoria. Este proceso es obligatorio y se levanta cada vez que se levanta la instancia.

Log Writer (LGWR): Este proceso de background escribe las entradas al redo log buffer hacia disco. Los datos de redo log son generados del redo log buffer como transacciones con commit, entonces el LGWR escribe estas entradas hacia los redo log files habilitados o en línea. Este proceso es obligatorio al levantarse la instancia.

Checkpoint (CKPT): A un tiempo especifico, todos los database buffers modificados en el SGA son escritos a los datafiles por el DBWR; este evento es llamado checkpoint. Este proceso es el responsable de avisarle al DBWR que debe de escribir los datos del database buffer caché a disco y de actualizar el número de secuencia de todos los datafiles y los control files indicando de esta manera el checkpoint más reciente. Este proceso es opcional y se especifica dentro del archivo de parámetros.

System Monitor (SMON): Efectúa la recuperación de la instancia en caso de que fallara algún otro proceso. Es utilizado cuando se levanta la instancia y entonces inicia la recuperación automática de la misma. También se encarga de limpiar los segmentos temporales no utilizados con frecuencia recuperando transacciones que quedaron suspendidas durante fallas de instancia debidas a errores de lectura de archivos cuando estos no están habilitados o en línea. Estas transacciones son recuperadas cuando el tablespace o el archivo son habilitados o puesto en línea. Este proceso es obligatorio.

Process Monitor (PMON): Ejecuta recuperación en caso de que fallara un proceso de usuario. También se encarga de limpiar el caché y liberar los recursos que el proceso estaba utilizando.

Archiver (ARCH): Este proceso copia los archivos de redo log en línea hacia algún dispositivo de almacenamiento cuando estos están llenos. ARCH se activa solamente cuando los redo log en uso son utilizados en modo de ARCHIVELOG. Se inicializa en el archivo de parámetros.

Recoverer (RECO): Resuelve las transacciones distribuidas que quedaron pendientes debido a una falla en la red o de sistema en una base de datos distribuida. En un tiempo específico, el RECO local intenta conectarse a la base de datos remota y automáticamente completa las transacciones distribuidas con commit y da rollback (deshace la transacción) a las inconclusas.

Dispatcher (Dnnn): Es un proceso de background opcional que se presenta cuando se utiliza la configuración de multi-threaded server. Un proceso de dispatcher es creado para cada protocolo de comunicación que se utiliza (D000, D001,...,Dnnn). Un dispatcher es el responsable de encausar las peticiones desde los procesos de usuario hacia un proceso de servidor compartido y regresar la respuesta al proceso de usuario que hizo la petición.

Lock (LCKn): Alrededor de 10 procesos de lock (LCK0,..., LCK9) son utilizados para liberar los candados internos de la instancia cuando se utiliza parallel server.

*1.4 Arquitectura de una base de datos SQL*Server*

Una base de datos SQL*Server Esta compuesta a manera general de dos componentes principales: el motor relacional –Conocido en inglés como Relational Engine (RE)- y el motor de almacenamiento – Conocido en inglés como Storage Engine (SE)-, ambos se comunican a través de API's de base de datos de tipo OLE. Así mismo estos componentes se integran al manejo de memoria interno del núcleo de SQL*Server.

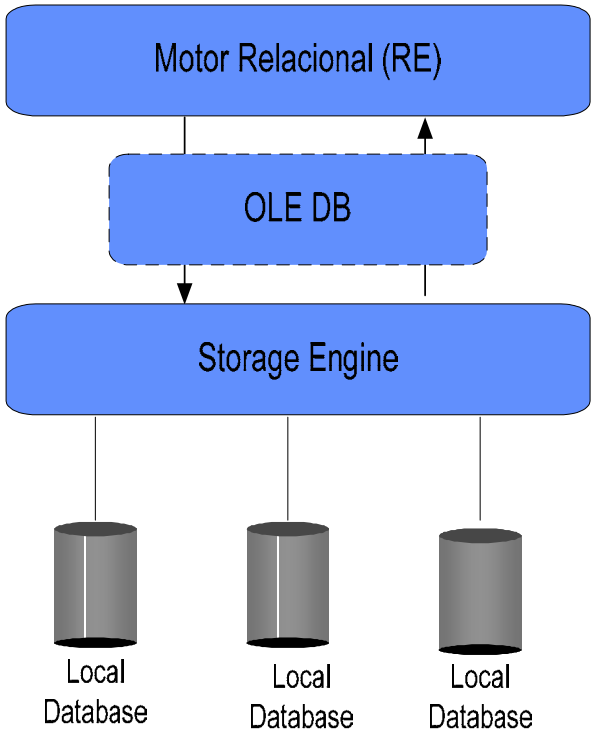


Figura # 3. Base de datos SQL*Server

Microsoft Corporation. Libro electrónico de SQL Server 2005 Books Online, Noviembre 2008, Material de libre exposición en internet [en línea]. Disponible en internet: < http://www.microsoft.com/downloads/en/confirmation.aspx?familyId=be6a2c5d-00df-4220-b133-29c1e0b6585f&displayLang=en >.

1.4.1 El motor relacional (RE).

Este motor tiene cuatro funciones principales:

Análisis del comando SQL.
Optimización del plan de ejecución.
Ejecución lógica de las operaciones.
Procesar los datos y las instrucciones DDL.

- **Análisis del comando SQL:** Se separa e interpreta cada una de las unidades lógicas como son las palabras claves, los parámetros, los operadores e identificadores.

- **Optimización del plan de ejecución:** Determina el plan de ejecución basado en la estimación de costos para manipular y obtener la información.

- **Ejecución lógica de las operaciones:** Esta función es precisamente el tratamiento ordenado de cada una de las operaciones necesarias para obtener la información.

- **Procesar los datos y las instrucciones DDL:** Se realiza la presentación de los datos de acuerdo a los criterios de cada DDL.

1.4.2 El motor de almacenamiento (Storage Engine).

El motor de almacenamiento realiza las siguientes funciones: Manejo de archivos de base de datos, manejo de espacio, construcción y lectura de las páginas físicas utilizadas para guardar la información, manejo de los buffers de datos, control de concurrencia, acceso y recuperación, y finalmente la implementación de funciones de copia, respaldo y recuperación de archivos y datos.

1.4.3 Enlace de Objetos (OLE DB).

Este motor realiza los enlaces a todos los componentes y fuentes de información. En conjunto estos componentes interactúan como una infraestructura que brindan a los programadores una forma para desarrollar aplicaciones con acceso a casi cualquier almacén de datos.

1.5 Arquitectura de una base de datos MySQL

La base de datos MySQL se constituye de varios componentes que siguen un flujo para formar la arquitectura compuesta en 2 capas principales a las cuales se les llama, la Capa de Aplicación: Aplicaciones e Interfases; y la Capa Lógica: Procesador de Consultas, Manejo de Transacciones, Manejo de Recuperación y Manejo de Almacenamiento.

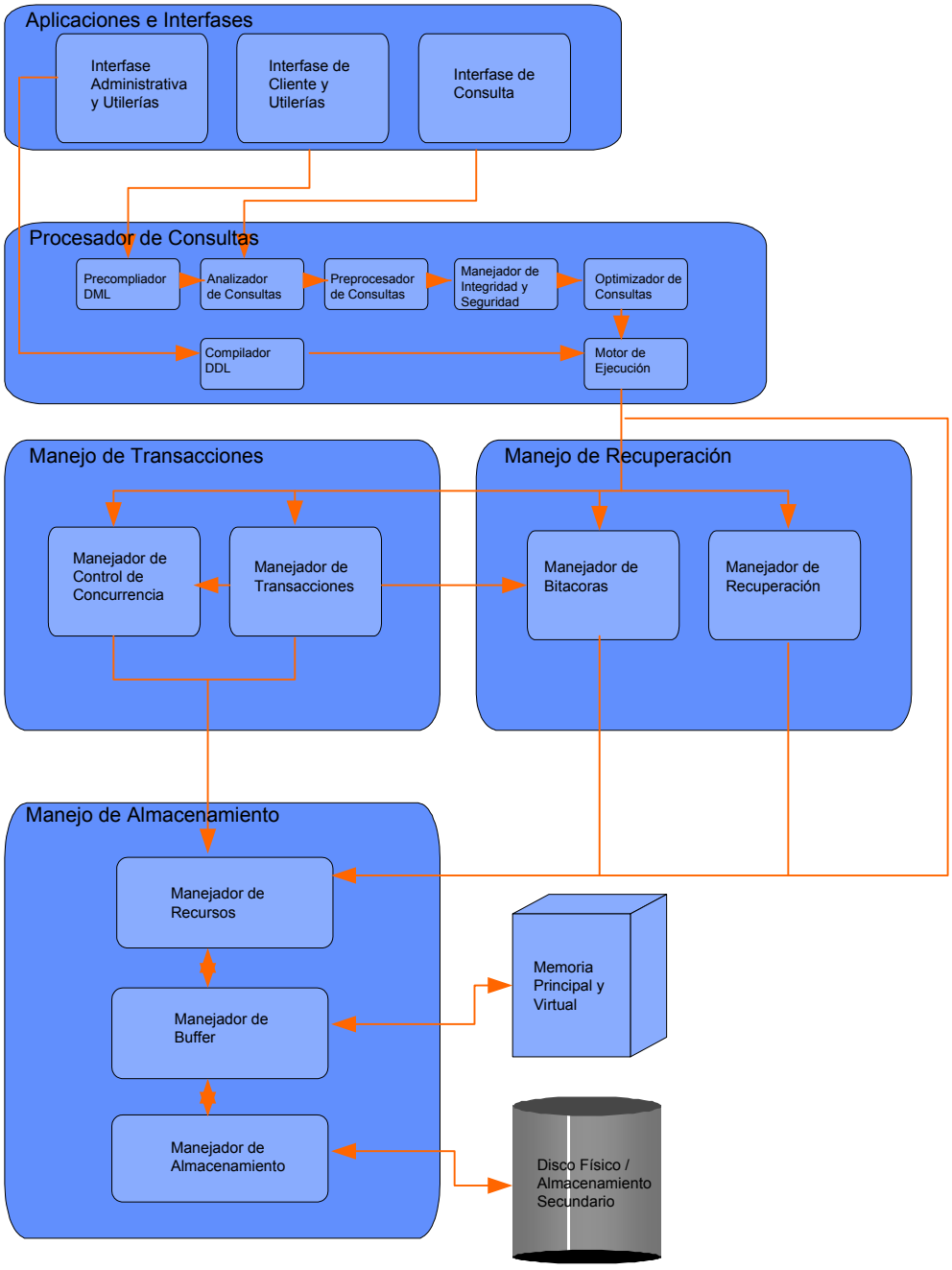


Figura # 4. Base de datos MySQL.

BANNON Ryan, Libro electrónico MYSQL Conceptual Architecture. Enero 2002, Material de libre exposición en internet [en línea] Disponible en internet < http://www.swen.uwaterloo.ca/~mrbannon/cs798 >.

1.5.1 Capa de Aplicación.

En esta capa se logra la interacción de los usuarios y los clientes con el RDBMS MySQL. Dentro de esta se tienen tres elementos básicos denominados como la interfase administrativa, la de clientes y la de usuarios o de consulta.

Los administradores frecuentemente utilizan diferentes herramientas que les ayudan a tener en buenas condiciones las bases de datos; utilerías como *mysqladmin*, *myisanchk* y *mysqldump* son de gran apoyo para un DBA. Mientras tanto los clientes se conectan a los datos a través de diversos programas desarrollados para diferentes lenguajes de programación, tal es el caso de MySQL API's, C API, DBI API, Perl, PHP API, Java API, etc. Finalmente los usuarios interactúan a través de la interfase *mysql* que permite ejecutar sentencias SQL directamente a las tablas.

1.5.2 Capa Lógica.

La capa lógica se explica de acuerdo a las partes que la componen.

1. **Procesador de Consultas:** La gran mayoría de interacción con el sistema ocurre cuando un usuario desea ver o manipular los datos almacenados. Estas consultas SQL son analizadas y optimizadas por el *procesador de consultas*. Cuando una solicitud es recibida el pre-compilador DML extrae las consultas relevantes y las traduce para obtener la información solicitada. El *compilador DDL* procesa las solicitudes de acceso a la base de datos MySQL.

   Después de que las consultas relevantes son extraídas entonces el *analizador de consultas* construye una estructura de árbol para que sea interpretado de mejor manera por los demás componentes como el *pre-procesador de consultas* que revisa la sintaxis y la semántica de la instrucción para catalogarlo como una estructura válida. Adicionalmente el *manejador de integridad y seguridad* revisa los permisos necesarios para determinar si el cliente tiene acceso a los objetos que son consultados. Una vez que los permisos han sido verificados el *optimizador de consultas* busca la forma más rápida de ejecutarla buscando índices que le ayuden a suprimir renglones fuera de las condiciones especificadas. Finalmente el *motor de ejecución* recupera la información.

2. **Manejo de Transacciones:** el *manejador de transacciones* es responsable de asegurar que la transacción sea autenticada y ejecutada de manera lógica y de resolver los problemas de deadlock y decidir los commits y rollbacks de las sentencias. El manejador de control de concurrencias se encarga de ejecutar las transacciones de manera separada e independiente para manejar los candados a nivel registro y a nivel tabla. Si una transacción nueva intenta ingresar al mismo dato el manejador de control de concurrencia lo rechaza hasta que la primera sea terminada.

3. **Manejo de Recuperación:** Esta se logra a través de dos procesos llamados *manejador de bitácoras* y *manejador de recuperación*. El primero se encarga de registrar cada operación que es ejecutada en la base de datos y la almacena en disco con apoyo del *manejador de buffer*, así si se presentara una caída del sistema esa información es aplicada una a una hasta llegar al último estado consistente o estable. El manejador de recuperación va aplicando los cambios de esas bitácoras a la base de datos.

4. **Manejo de Almacenamiento:** Típicamente el espacio físico es almacenado en dispositivos secundarios sin embargo el acceso a ellos no es una tarea simple y se logra a través del *manejador de buffer*. Este trabaja en conjunto con otros manejadores. En el nivel más inferior el *manejador de almacenamiento* es el mediador entre el *manejador de buffer* y el almacenamiento secundario, es

entonces cuando el manejador *de buffer* asigna los recursos de memoria para el uso y manipulación de los datos derivados de una solicitud del *manejador de recursos* que a su vez las acepta del *motor de ejecución* colocándolas el tablas de memoria manejadas por el *manejador de buffer*.

2 La administración de la base de datos.

2.1 Administración de bases de datos

Este capitulo está dedicado en especifico a bases de datos Oracle con el objeto de mostrar de manera mas clara una estructura adecuada para la construcción de una base de datos poniendo especial énfasis en las mejores prácticas de los expertos DBA's. Así mismo se pretende enriquecer este texto con la mayor experiencia del titular de este trabajo en la base de datos oracle aprovechando las ventajas que ofrece el diseño de oracle como la base de datos más fiable en el mercado.

Administrador de la(s) base(s) de datos (DBA): Es la persona encargada de administrar, y dar mantenimiento a la base de datos. Debido a que una base de datos puede tener un gran numero de usuarios, estas tareas de administración pueden ser divididas y asignadas a mas DBA's, encargados cada uno de una labor específica de administración consideradas como sus responsabilidades.

El DBA juega un papel muy importante en el buen desempeño del RDBMS. Es el manager principal del sistema de bases de datos, también es el responsable de verificar y asegurar que el software y hardware funcionen adecuadamente según las peticiones de los usuarios.

Es posible considerar que las actividades principales del DBA se involucran en los siguientes aspectos:

1. Instalación y mantenimiento del software.
2. Diseño de la base de datos.
3. Montaje y desmontaje de la base de datos.
4. Seguridad en la base de datos.
5. Autentificación de usuarios.
6. Límite de recursos y perfiles.
7. Auditoria en la base de datos.
8. Afinación en la base de datos.
9. Planificación del almacenamiento de los datos.
10. Cuidado en el crecimiento de las bases de datos.
11. Elaboración e implementación de esquemas de respaldo y recuperación.

El DBA requiere del buen entendimiento de la operación y funcionamiento de la base de datos, es decir, debe conocer la cantidad de usuarios que van a tener acceso a la base de datos,

los privilegios van a tener y sobre que objetos, los datos que son consultados y almacenados con mayor frecuencia, el tipo de transacciones se efectúan con mayor regularidad, la distribución de los database files a nivel sistema operativo, y en general, todos aquellos tópicos que estén involucrados con el mejoramiento del desempeño de la base de datos.

Para realizar todas estas actividades el DBA cuenta con una serie de herramientas que le ayudan a administrar de la manera más eficiente posible las bases de datos a su cargo.

Las utilerías que son utilizadas con más frecuencia son:

SQL. Es una utilería de Oracle que le permite al administrador la manipulación y el control de una base de datos Oracle; todas las operaciones administrativas pueden realizarse a través de esta herramienta en modo texto e incluso a través de una consola gráfica de administración llamada Oracle Enterprise Manager.

SQL*Loader. Esta herramienta es utilizada por los administradores y usuarios en general para cargar grandes volúmenes de datos desde archivos de sistema operativo a tablas de la base de datos.

Export/Import. Permiten al administrador mover datos en formato Oracle hacia otra localización en la misma base de datos, hacia otra base de datos o migrarlos hacia otra plataforma con diferente sistema operativo. El export solo puede ser leído por la utilería Import.

2.1.1 Instalación y mantenimiento del software

Para poder realizar una instalación del software de Oracle es necesario considerar requerimientos de vital importancia según la plataforma, versión del RDBMS, versión de los productos, y versión de sistema operativo. El considerar todos estos requerimientos asegurará el éxito o fracaso de la instalación.

Existen plataformas con determinados sistemas operativos, versiones de RDBMS desarrollados para determinados sistemas operativos y versiones de productos desarrollados para determinadas versiones de RDBMS. Por tal motivo es importante verificar como una tarea primordial que software está certificado con el hardware que se vaya a considerar.

Otro punto de importancia es el determinar las variables de ambiente que deben ser inicializadas a nivel sistema operativo y que son requeridas por el software de Oracle.

También se necesita planificar el lugar en el cuál se va a instalar el software, el lugar donde van a residir las bases de datos y si se instalarán varias versiones de RDBMS para correr diversas aplicaciones.

2.1.2 Diseño de la base de datos

Antes de que una base de datos pueda ser utilizada por las aplicaciones, esta debe ser creada. La creación de la base de datos involucra la preparación de varios archivos de sistema operativo para que puedan operar junto con una base de datos Oracle. Una base de datos necesita ser creada solo una vez sin importar la cantidad de datafiles que la constituyan ni cuantas instancias tengan acceso a ella. Si se desea, la creación de la base de datos puede utilizarse para borrar información de una base de datos ya existente y crear una nueva base de datos con el mismo nombre y estructura física.

El hecho de crear una base de datos incluye las siguientes operaciones:

- Creación de nuevos datafiles o el borrado de algún dato existente en un datafile creado con anterioridad.

- Escribir los datos iniciales de la base de datos (el diccionario de datos) requeridos por Oracle para acceder y usar la base de datos durante operación normal.

- Crear e inicializar los Control files y los Redo Log files para la base de datos.

La creación de la base de datos es ejecutada por medio del comando de SQL CREATE DATABASE; sin embargo, antes de ejecutar este comando, el DBA debe realizar algunas tareas de preinstalación descritas a continuación:

- La cantidad de espacio requerido por los datos de la base de datos (sizing).

- El esquema de la base de datos.

Dependiendo del sistema operativo que se vaya a ocupar, una estructura de base de datos puede ser creada automáticamente como parte del proceso de instalación del RDBMS de Oracle. El DBA tiene la opción de utilizar la base de datos inicial y adaptarla a los requerimientos de información, o crear una o más bases de datos nuevas que diseñadas previamente según estos requerimientos. Una base de datos Oracle nueva tiene registrados a dos usuarios únicamente, el usuario SYS y el usuario SYSTEM. La clave de estos dos usuarios debe cambiarse inmediatamente después de crear la base de datos por cuestiones de seguridad ya que tienen los privilegios de usuarios administradores con acceso a objetos internos con propósitos de administración y control de la misma.

2.1.3 Montaje y desmontaje de la base de datos

Levantar o montar la base de datos (STARTUP).

Existen tres pasos para montar la base de datos y hacerla disponible a diferentes niveles según las actividades o requerimientos del sistema:

1. **Levantar la instancia:** Este procedimiento incluye la asignación de un SGA, un área de memoria compartida utilizada por la información de la base de datos, y la creación de los procesos de background. La instancia es montada antes de abrir la base de datos; es importante aclarar que solo la instancia ha sido montada y que no existe ninguna base de datos relacionada con ella hasta este punto. El hecho de levantar la instancia también es conocido como poner la base de datos en modo *no mount*.

2. **Montar la base de datos:** Es el proceso de asociar una base de datos con una instancia previamente montada. Después de que la base de datos ha sido montada, esta permanece cerrada a los usuarios y solo es accesible cuando el DBA requiere realizar actividades de mantenimiento.

   Cuando una base de datos es montada, los archivos especificados por el parámetro CONTROL_FILES contenidos en el archivo de parámetros utilizado (init<NOMBRE_DE_LA_INSTANCIA>.ora) son localizados y abiertos para levantar la instancia. Una vez que los Control files son abiertos, el RDBMS de Oracle los

interpreta para obtener de ellos los nombres y las rutas de los Redo Log files y los Database files confirmando de esta manera que existan dentro de la base de datos.

3. **Abrir la base de datos:** Este proceso habilita el uso de la base de datos par todos los usuarios y está disponible para su operación normal. Una vez que ha sido abierta, cualquier usuario autorizado puede conectarse a la base de datos y acceder la información dentro de ella. En la mayoría de los casos, el DBA abre la base de datos hasta que esta se encuentra disponible para el uso general del sistema.

Desmontar la base de datos (SHUTDOWN).

Al igual que en el proceso de levantar la base de datos, también se requieren tres pasos para desmontarla.

1. **Cerrar la base de datos:** Es el primer paso que se realiza durante el proceso de shutdown. Cuando una base de datos se cierra, todos los datos y sus cambios en la base de datos se escriben a los database files y a los Redo Log files respectivamente. Después de esta operación, los Data files y Redo Log files en línea se cierran. Los Data files fuera de línea y los tablespaces fuera de línea también se cierran.

2. **Desmontar la base de datos:** Este paso desmonta o suprime la asociación de la instancia con la base de datos. Después de desmontar la base de datos solo permanecerá la instancia en la memoria del sistema. Los Control files de la base de datos se cierran.

3. **Desmontar la instancia:** Cuando la instancia es desmontada, el SGA es removida de la memoria y los procesos de background son terminados por el sistema que ejecuta el RDBMS de Oracle.

2.1.4 Seguridad en la base de datos

El RDBMS de Oracle maneja la seguridad de la base de datos de varias formas diferentes entre las cuales se encuentran los esquemas y los usuarios.

Un esquema es una colección de objetos esquema (tablas, vistas, clusters, procedimientos, paquetes, etc.). Cada base de datos Oracle tiene una lista de esquemas y todos los objetos de un esquema pertenecen exclusivamente a él.

Cada base de datos Oracle tiene una lista de usuarios válidos. Para acceder a la base de datos, un usuario debe correr una aplicación (SQL*Forms, SQL*Plus, algún pre compilador, etc.) y conectarse utilizando un nombre válido con su correspondiente clave de acceso. Cuando se crea un usuario en la base de datos, se crea un esquema del mismo nombre para el usuario.

Por omisión, una vez que el usuario se conecta a la base de datos este puede acceder todos los objetos que contenga el esquema correspondiente. Un usuario de la base de datos puede estar asociado con un esquema de mismo nombre, así, los términos de esquema y usuario en la base de datos son similares; sin embargo, un usuario puede tener permisos para ver objetos de otro esquema.

El derecho de acceso de un usuario es controlado por el diferente dominio de seguridad para el usuario determinado por el DBA.

2.1.5 Autentificación de usuarios

Para prevenir el uso no autorizado de un usuario de la base de datos, El RDBMS de Oracle provee de una validación de usuario utilizando dos métodos diferentes:

- Autentificación por sistema operativo
- Autentificación por medio de la base de datos Oracle.

Por simplicidad, solo un método es utilizado para autentificar a todos los usuarios de la base de datos; sin embargo, El RDBMS de Oracle permite la utilización de los dos métodos en la misma base de datos.

Tablespaces y Cuotas para los usuarios.

Como parte de la seguridad, pueden inicializarse varios parámetros sobre el uso de los tablespaces. Un tablespace es un área de almacenamiento lógico de datos. Dichos tablespaces pueden ser asignados al usuario como sigue:

- **Un tablespace de default.** Que servirá para almacenar todos los objetos que se crearán bajo el esquema del usuario.
- **Un tablespace temporal.** Que servirá para realizar todas las operaciones de ordenamiento en caso de que el espacio en memoria no sea suficiente para esta tarea.

Por otro lado, también se le puede asignar una cuota sobre otros tablespaces de la base de datos.

Una cuota es un espacio asignado al usuario dentro del tablespace.

2.1.6 Límite de recursos y perfiles

Otro método que sirve al DBA para dar seguridad a la base de datos es el limitar la utilización de los recursos del sistema. Explícitamente, se limitan los recursos (tiempo de CPU, tiempo ocioso, etc.) a cada usuario del sistema para que el administrador pueda controlar su uso irracional.

La característica de limitar los recursos del RDBMS de Oracle es muy utilizada en sistemas con grandes cargas de trabajo o en sistemas con gran cantidad de información. En la mayoría de los ambientes, los recursos del sistema son muy caros, y el exceso el consumo de estos puede afectar la actividad de otros usuarios en el sistema. En sistemas monousuarios o de pequeña escala, los recursos requieren de menor administración ya que no disminuyen considerablemente la actividad de otros usuarios.

Tipos de recursos de sistema y sus límites.

El RDBMS de Oracle puede limitar el uso de varios tipos de recursos del sistema. En general, cada recurso puede controlarse a nivel de sesión, a nivel de llamada o en ambos niveles.

Nivel sesión. Cada vez que un usuario se conecta a la base de datos, una sesión es creada. Cada sesión consume una cantidad mínima entre tiempo de CPU y memoria en la máquina en donde es ejecutado el RDBMS de Oracle. El RDBMS permite limitar varios recursos a nivel sesión.

Si los límites de recursos a nivel sesión son excedidos, la sentencia que es ejecutada en el momento de pasar esos limites es terminada y se le da ROLLBACK (Deshace la transacción en curso), en ese momento aparece un mensaje que indica que el limite ha sido excedido. En este punto, todas las sentencias previas en la transacción quedan intactas, y las únicas operaciones que el usuario puede ejecutar son el COMMIT, el ROLLBACK, o desconectarse (en este caso, se le da commit a la transacción), y las demás operaciones producen un mensaje de error. Después de que a la transacción se la da commit o rollback el usuario tendrá que salirse de la sesión actual y abrir una nueva para continuar su trabajo.

Nivel llamada. Cada vez que una sentencia de SQL es ejecutada, se efectúan varios pasos para procesar el argumento. Durante este proceso se hacen varias llamadas a la base de datos en diferentes fases de ejecución. Para prevenir que una llamada utilice excesivamente el sistema, el RDBMS permite limitar los recursos a nivel llamada.

Si un recurso a nivel llamada es excedido, el procesamiento de la sentencia se detiene y se efectúa el proceso de rollback regresando un mensaje de error. Sin embargo, todas las sentencias previas dentro de la transacción permanecen intactas y la sesión del usuario sigue conectada al servidor.

Algunos de los recursos que pueden limitarse son:

Tiempo de CPU. Cuando se hacen llamadas al RDBMS por parte de sentencias SQL y otros tipos de llamadas, una cierta cantidad de tiempo de CPU es requerida para procesarlas. La gran mayoría de llamadas requieren de una pequeña cantidad de tiempo de CPU. Sin embargo, una sentencia SQL que involucra una gran cantidad de datos a ser consultados puede consumir una gran cantidad de tiempo de CPU, reduciendo este recurso para otros procesos en el sistema.

Para prevenir el uso descontrolado de tiempo de CPU, el RDBMS puede limitarlo por llamada o ya sea por una duración determinada por sesión. Estos límites se miden en centésimas de segundo.

Lecturas lógicas. Las entradas/salidas son de las operaciones que representan mayor costo en un sistema de base de datos. Las sentencias con entradas/salidas intensivas pueden acumular peticiones en el uso de memoria y de disco. Las operaciones de la base de datos en estas circunstancias ocasionan la competencia por estos recursos provocando uno de los problemas más graves en el desempeño de la base de datos conocido como *contención*.

Para prevenir los excesos de entrada/salida, el RDBMS puede limitar las lecturas lógicas de bloques de datos por llamada o por sesión. Este tipo de lectura en los bloques de datos se efectúa en memoria y disco. Para limitar este recurso es necesario inicializar estas medidas en números de bloque de datos por llamada o por sesión.

Otros recursos que pueden limitarse a nivel sesión son:

- El número de sesiones concurrentes por un solo usuario.
- El tiempo ocioso en la actividad por sesión.
- El tiempo de conexión por sesión.
- La cantidad de SGA utilizada privada (SQL área).

Estos recursos pueden limitarse en forma individual, es decir que por cada usuario se limitarán los recursos que el administrador considere necesarios según las actividades del mismo en la base de datos. También se pueden limitar los recursos por medio de perfiles. Un perfil es un objeto de la base de datos que contiene una serie de recursos asignados a un grupo de usuarios.

Otorgar privilegios sobre los objetos de la base de datos.

Los privilegios sobre objetos de la base de datos pueden otorgarse o revocarse para todos los usuarios de la base de datos o para los roles creados en la base de datos. Los roles son un conjunto de privilegios que se otorgan a un esquema de usuario, con la ventaja principal de que facilitan la administración de privilegios sobre objetos.

Los privilegios de objetos pueden otorgarse o revocarse utilizando los comandos de SQL GRANT y REVOKE.

Para permitir la seguridad en las tablas de la base de datos, son permitidos los privilegios de objetos en dos niveles:

Operaciones DML (Lenguaje de Manipulación de Datos). Son los privilegios que permiten las operaciones de DELETE, INSERT, SELECT y UPDATE en las tablas de la base de datos. Estos privilegios deben ser otorgados a los usuarios o roles que necesitan manipular datos en una tabla.

Operaciones DDL (Lenguaje de Definición de Datos). Son los privilegios que permiten operaciones con las sentencias de ALTER, CREATE, REFERENCE, etc. Sobre los objetos de la base de datos. El privilegio de REFERENCE se utiliza solo para las dependencias entre tablas, ya sea por llave primaria o una clave única.

2.1.7 Auditoria en la base de datos

Esta es una actividad conceptualizada en la seguridad de la base de datos por medio de un monitoreo a nivel sesión, a nivel sentencia, a nivel recursos, a nivel privilegios o a nivel objetos. Consiste en la investigación de los siguientes puntos:

- Investigación de actividades sospechosas en la base de datos.
- Monitorear y recolectar información con respecto a las actividades de la base de datos.

Para llevar a cabo la auditoria de la base de datos se habilita un parámetro conocido como AUDIT_TRAIL, inicializado con el valor de TRUE. Con este parámetro el RDBMS infiere que debe llenar una serie de tablas de administración mediante las cuales se obtienen las estadísticas necesarias para llevar a cabo esta actividad.

2.1.8 Afinación en la base de datos

El proceso de afinación de la base de datos es una tarea ardua y que requiere la experiencia del DBA para determinar los movimientos adecuados en las bases de datos, por tal motivo esta actividad se analizará en el capitulo 4 con los detalles necesarios y que serán considerados para lograr interpretar las estadísticas obtenidas para la afinación de la base de datos. A nivel general hablaremos de cómo poder obtener el mejor desempeño de las sentencias de SQL construidas para la manipulación de datos, como poder asignar los recursos de memoria adecuados para un funcionamiento óptimo; también hablaremos de cómo evitar problemas de contención de recursos y como aprovechar adecuadamente los recursos para un uso efectivo de los accesos de lectura y escritura en disco.

2.1.9 Planificación del almacenamiento de los datos

Mediante los estándares de arquitectura de flexibilidad óptima (OFA) de los cuales hablaremos posteriormente se contempla la planificación del crecimiento de la base de datos y en general del sistema. Utilizando dicha arquitectura veremos que el control del almacenamiento en disco puede llegar a tener un buen control para separar datos e índices manteniendo un buen nivel de administración de los recursos de almacenamiento.

2.1.10 Cuidado en el crecimiento de las bases de datos

La arquitectura de flexibilidad óptima también cubre el aspecto de cuidar el crecimiento de la base de datos, ya que considera la planificación de files system y de tablespaces, que son las estructuras por parte de sistema operativo y por parte de la base de datos respectivamente que se encargan de contener toda la información.

2.1.11 Elaboración e implementación de esquemas de respaldo y recuperación

Este tema lo revisaremos con mayor detalle en el capitulo 3 de este trabajo. El contar con un buen esquema de respaldos considerando los recursos existentes e incluso llegando a proponer mejores herramientas, hardware y software de última generación y un nivel de experiencia y preparación por parte del DBA más depurado en situaciones críticas nos dará el punto de retorno más adecuado para las fallas que puedan suscitarse en los sistemas; asegurando tiempos de inactividad lo más cortos posibles y con ello ayudar a mantener los niveles de disponibilidad de las bases de datos en umbrales adecuados a las necesidades del negocio.

2.2 OFA (Arquitectura de Flexibilidad Óptima)

El estándar de **OFA** identifica los requerimientos de configuración para mejorar considerablemente el desempeño a bajo costo. En 1991 Oracle introdujo OFA, un método organizacional de bajo mantenimiento para garantizar un alto desempeño y controlar el crecimiento de las bases de datos. El principal objetivo fue ayudar a que los usuarios

entendieran con mayor claridad y facilidad las relaciones de los productos de Oracle con el sistema operativo.

Una propuesta personal es que este estándar de construcción de bases de datos es que sin importar que fue creado para bases de datos Oracle también puede ser implementado en otras bases de datos como lo son SQL*Server y MySQL utilizados como referencia durante este trabajo de investigación.

Los problemas de configuración que motivaron el desarrollo de OFA son:

- Una pobre distribución de los recursos disponibles que provocaban un bajo desempeño de las aplicaciones del sistema.
- Al intentar administrar varias bases de datos Oracle frecuentemente se tenían problemas de corrupción de datos.
- Una administración no adecuada de los segmentos de crecimiento que provocaban fallas recurrentes en las aplicaciones.

Entrando de lleno en los puntos más importantes se propone que primero se hable del tema general y en base a ello se coloquen las reglas de OFA que aplican a ese punto del que hablamos. Sin más preámbulo aquí comenzamos.

2.2.1 Configuración del sistema operativo

Si se planea instalar un RDBMS, probablemente tenga que instalarse sobre él una aplicación muy grande. Antes de que el sistema pueda servir para la aplicación de Oracle, debemos tomar en cuenta las configuraciones de los dispositivos periféricos, los discos, el espacio para swap, el almacenamiento de los datos, el nombre de los files system, la red y la preparación del kernel para las demandas específicas de la aplicación, además de la creación de cuentas de acceso.

Puntos de montaje.

La definición de los puntos de montaje es la primer idea a considerar para configurar un sistema UNIX/LINUX e incluso Windows, con ello lograremos seleccionar el tamaño y el nombre del punto de montaje para el file system. Un *punto de montaje* es el nombre del directorio que denota la posición donde será asociado el file system con una sola partición de disco dentro del sistema operativo. La siguiente tabla, muestra el mapeo de dos puntos de montaje en un equipo Sun Solaris.

# device # to mount	device to fsck	mount point	FS type	fsck pass	mount atboot	mount options
/dev/dsk/c0r3d0s0	/dev/rdsk/c0r3d0s0	/	ufs	1	yes	-
/dev/dsk/c0r3d0s6	/dev/rdsk/c0r3d0s6	/usr	ufs	2	yes	-

Mapeo de 2 puntos de montaje, / y /usr en dos particiones de un solo disco.

Para seleccionar los puntos de montaje debemos considerar el equilibrio de los siguientes requerimientos para la configuración planeada:

Requerimiento 1. Los files system debe estar organizado para facilitar la administración del crecimiento del mismo, tomando en cuenta las bases de datos existentes en el sistema, el nuevo ingreso de usuarios, la creación de futura bases de datos y la habilitación de hardware nuevo.

Requerimiento 2. Debe ser posible la distribución de cargas de I/O a través de un número suficiente de controladoras de disco para prevenir la contención.

Requerimiento 3. Debe disminuir el costo de hardware.

Requerimiento 4. Requiere aislar el impacto de fallas de disco a través del menor número de aplicaciones posible.

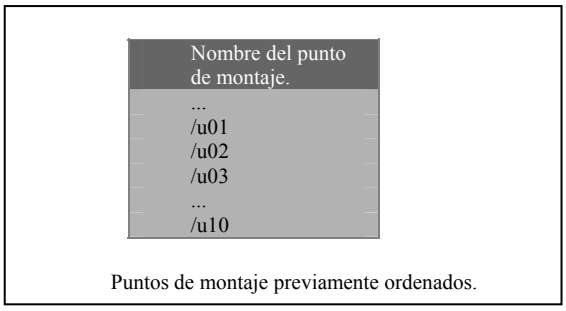
La forma de balancear los requerimientos 2 y 3 se logra a través de nombrar los puntos de montaje de tal forma que sea posible colocar un archivo de una base de datos sin violar el requerimiento 1.

Ahora bien, una vez que se ha definido el contexto, explicaré las reglas OFA que aplican hacia este tema de estructura en el sistema operativo.

OFA # 1.

Nombrar todos los puntos de montaje de tal manera que se permita identificar el lugar específico de los datos mediante el siguiente patrón: ***/pm***, donde ***p*** es una cadena constante y m es una llave de identificación única que distingue un punto de montaje de otro.

Proporcionar puntos de montaje que contengan varios nombres que hagan posible la sencilla identificación de un punto de montaje de otro, como pueden ser */u01* y */u02*, utilizando ceros con el propósito de que la longitud del nombre sea fija y pueda efectuarse un ordenamiento de los puntos de montaje.



Puntos de montaje previamente ordenados.

Desafortunadamente, el almacenamiento de archivos de dos o más bases de datos en el mismo controlador contradice el requerimiento 4. La única forma para satisfacer los requerimientos 2 y 4 es la instalación de más discos en el sistema. Por lo regular, los administradores prefieren sacrificar el requerimiento 4 y darle más importancia al requerimiento 2 por considerarlo más propio para efectos de administración. El grado en que pudiera ser resuelto este conflicto de hardware determina la estrategia que debe tomarse para optar por los nombres de los puntos de montaje.

Directorios Home para acceso a sistema operativo.

En sistemas UNIX anteriores, los directorios home eran colocados en */usr*. Esto trabajaba bien en sistemas monousuario, pero el tener los directorios home en */usr* causaba retos innecesarios. Por ejemplo, el tener los directorios home en */usr* aumentaba el riesgo de una pérdida accidental del archivo cuando se remplazaba el subdirectorio en caso de realizar una actualización; también, si el File system */usr* se llenaba, entonces se provocaba la caída del sistema operativo. Los sistemas UNIX de actualidad recomiendan que los directorios home se coloquen en un directorio llamado */u* o */home*. El uso de */u* o */home* beneficia el trabajo de los sistemas con un alto promedio de volumen de datos. Sin embargo, hay un inconveniente si los archivos almacenados en esos directorios son muy grandes para una sola partición de disco; por ejemplo, las aplicaciones de Oracle llamadas Oracle Financials y Oracle manufacturing consumen casi un gigabyte, que al momento de instalar dichas aplicaciones se asume por default que toda esta información estará almacenada en un directorio llamado *applmgr*, pero una sola partición no puede permitir tal cantidad de información concentrada en un solo directorio. Entonces es necesario satisfacer lo siguiente:

Requerimiento 5. Debe ser posible distribuir a través de dos o más controladores de disco toda la colección de directorios home y el contenido de un solo directorio home individual.

La siguiente regla OFA es una forma de satisfacer el requerimiento 5:

OFA # 2.

Nombrar los directorios home mediante el patrón ***/pm/h/u***, donde ***pm*** es el nombre del punto de montaje, ***h*** es seleccionado de un conjunto de nombres de directorios estándar, y ***u*** es el identificador del usuario dueño del directorio.

Colocar todos los directorios home al mismo nivel en un file system de UNIX, significa que podemos poner los directorios home en diferentes puntos de montaje y hacer referencia a ellos como una colección de logins home mediante un solo patrón. Por medio de esta explicación, es posible observar que la primera parte del requerimiento 5 se cumple sin violar el requerimiento 1 al colocar 2 subdirectorios home muy grandes al mismo nivel en diferentes puntos de montaje.

Para explicar la segunda parte del requerimiento 5, tomaremos en cuenta a los dueños del software de las aplicaciones de Oracle Financials y Oracle Manufacturing llamado applmgr, que puede llegar a ocupar más de un gigabyte de un File system UNIX y cuya partición de disco es de 600 megabytes. Una solución muy sencilla para cubrir este requerimiento es la utilización de ligas simbólicas para hacer que los directorios aparezcan en un solo subdirectorio aunque físicamente estén alojados en diferentes puntos de montaje. La tabla de abajo presenta un listado donde se observa una liga simbólica requerida para habilitar el software de la aplicación Oracle GeneralLedger que reside en puntos de montaje separados. También aparecen otras aplicaciones montadas *en /u02/app/applmgr*. Todos los archivos de *applmgr* son identificables como residentes de los subdirectorios cuyos nombres son marcados por el patrón */*/app/applmgr*.

$ ln -s /u03/app/applmgr/gl /u02/app/applmgr/gl							Creación de la liga simbólica.
$ ls -l /u02/app/applmgr							Lista el contenido del directorio.
- rw -r --r- -	1	applmgr	1119	Jul	5	01:16	AOL.env
drwxrwxr-x	2	Applmgr	2048	Jul	5	01:16	alr
drwxrwxr-x	2	applmgr	2048	Jul	5	01:16	fnd
lrwxrwxr-x	1	applmgr	5	Jul	5	01:16	gl -> /u03/app/applmgr/gl
$ ls -l /u03/app/applmgr							
drwxrwxr-x	1	applmgr	2048	Jul	5	01:16	gl
Liga simbólica a través de varios discos.							

Los valores diferentes de *h* en la regla OFA # 3 que se explicará más adelante, permiten al administrador del sistema simplificar los procedimientos de respaldo utilizando la ruta específica para los diferentes tipos de usuarios.

Mínima administración de mantenimiento.

Con frecuencia al realizar visitas a clientes, es posible encontrar que sus programas de respaldo no están actualizados con los cambios de estructura de files system. En vario lugares, un ejercicio de balanceo de rutinas de I/O que pudieran consumir diez minutos actualmente consumen varias horas de tiempo en la base de datos. Por tal motivo se debe cumplir el siguiente requerimiento.

Requerimiento 6. Debe ser posible agregar o mover los directorios home de acceso al sistema sin necesidad de revisar los programas a los que hacen referencia.

Para satisfacer este requerimiento se tiene la siguiente regla de OFA.

OFA # 3.

Referirse explícitamente a los nombres de path solamente en los archivos designados para almacenarlos, tal caso es la ruta de UNIX para el archivo */etc/passwd* y el archivo de Oracle */etc/oratab*; haciendo referencia a miembros de un grupo determinado en la ruta */etc/group*.

2.2.2 Estándar de archivos Oracle

Antes de la implementación de OFA, los administradores colocaban los archivos de la base de datos bajo la ruta $ORACLE_HOME/dbs, esto provocaba grandes problemas de contención, fragmentación y distribución. Uno de los objetivos principales de OFA fue el solucionar estas dificultades con un conjunto de recomendaciones que permita distribuir los archivos a través de múltiples discos evitando la desorganización de los mismos permitiendo de esa forma que el software de los productos instalados no interfiriera con los datos almacenados por las aplicaciones. Esta separación es un caso específico del siguiente requerimiento:

Requerimiento 7. Los archivos deben separase en subdirectorios independientes según la categoría a la que pertenezcan.

La clasificación de los Oracle files mencionada a continuación permite a los administradores construir un estándar que satisfaga este requerimiento:

Nombre del Oracle file.	Descripción.
Archivo de productos.	Contiene el software del RDBMS y sus herramientas.
Archivos necesarios para la administración.	Contiene datos con referencia a la administración de la instancia y/o la base de datos, incluyendo los Redo Log archivados, diagnósticos de salida de los procesos de servidor, scripts de creación de las bases de datos, exports en línea, y los archivos de parámetros de la instancia.
Software local.	Contiene el software utilizado con Oracle y que ha sido comprado por separado.
Archivos de la base de datos.	Contiene los Control files, Redo Log files y los Datafiles.

Archivos Oracle estándar.

2.2.3 Software de Oracle y datos de administración

La figura # 5 es una representación gráfica del estándar de OFA para el almacenamiento de los productos de Oracle, archivos administrativos y archivos de software local instalados en subdirectorios cuyo dueño es el usuario Oracle. El nombre del directorio principal es el inicializado en la variable de ambiente ORACLE_BASE.

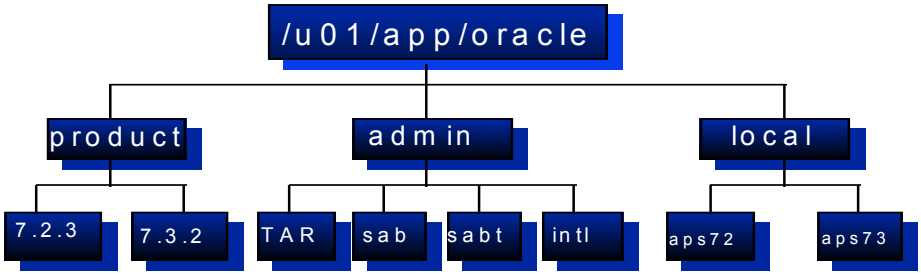


Figura # 5. Representación OFA de la estructura de directorios.

MILLSAP, Cary. Oracle Corporation White paper "The OFA Standard – Oracle for Open Systems".Septiembre 1995, Material de libre exposición en internet [en línea]
Disponible en internet < http:// www.lifeaftercoffee.com/downloads/oracle_ofa_whitepapaer.pdf >.

Archivos del producto.

Este directorio requiere de evaluar futuras integraciones de nuevas versiones de software de Oracle sin impactar la operación diaria. Tal acción es enunciada a continuación:

Requerimiento 8. Debe ser posible la ejecución de varias versiones de software de aplicación en forma simultánea. Esta actualización debe ser simple para el administrador y tan transparente para los usuarios como sea posible.

La regla OFA asociada a este requerimiento se enuncia como sigue:

OFA # 4.

Almacenar cada versión del software de Oracle en un directorio con el patrón ***h/product/v*****,** donde ***h*** es el directorio home del propietario del software de Oracle, ***product*** especifica que dentro de ese directorio estarán almacenados los productos y, ***v*** es la versión del software.

La mayoría de las implementaciones de UNIX utilizan el valor de ***v*** como 10.0.2. Los parches de Oracle involucran cambios solo en el número de la versión a la derecha del tercer punto decimal (por ejemplo: 10.0.2.1 a 10.0.2.3).

Archivos administrativos.

Una de las actividades clave del DBA es la habilidad de manejar grandes cantidades de datos en un sistema Oracle. En operaciones normales como pueden ser la instalación de scripts que realizan la creación de las bases de datos Oracle crea sus propios archivos traces y los administradores guardan la estructura de registros, los parámetros de la instancia, las estadísticas de desempeño, respaldos, Redo Log archivados y en general una bitácora de las transacciones de cada una de las bases de datos. El volumen de datos que son administrados aumenta conforme al número de las bases de datos en el sistema.

Requerimiento 9. La información administrativa relacionada con cada una de las bases de datos debe separarse de la información de otras bases de datos; debe ser una estructura razonable para la organización y almacenamiento de los datos administrativos.

En este ambiente puede determinarse la siguiente regla de OFA.

OFA # 5.

Para cada base de datos con ***db_name=d***; los archivos de administración de la base de datos listados en la figura # 6 deben almacenarse en el subdirectorio **h/admin/>d**, en donde *h* es el directorio home del propietario del software de Oracle:

Software local.

UNIX es un sistema operativo diseñado para ambientes abiertos que permiten agregar capacidades extras al sistema operativo. De manera muy similar OFA proporciona a los DBA’s algunas capacidades dentro de subdirectorios residentes bajo el directorio home del propietario del software local (software de Oracle). La descripción de este tipo de software a continuación.

Directorio de administración.	Descripción.
adhoc	Scripts SQL adaptados a una base de datos específica.
adump	Archivos de auditoria de la base de datos.
arch	Archivos de Redo Log respaldados.
bdump	Archivos trace de los procesos de background.
cdump	Archivos de la estructura interna de la base de datos.
create	Programas utilizados para crear la base de datos.
exp	Archivos Export de la base de datos.
logbook	Archivos históricos de la base de datos.
pfile	Archivos de parámetros de la instancia.
udump	Archivos trace de las sentencias SQL de los usuarios.

Software y archivos de administración local

2.2.4 Archivos de la base de datos

Los tiempos de vida de los archivos de la base de datos difieren del resto de los archivos del sistema, también requieren de estrategias diferentes de respaldo. Tal situación debe ser apegada al cumplimiento del siguiente requerimiento.

Requerimiento 10. Deben nombrarse los archivos de la base de datos de forma tal que sean identificables del resto de los archivos del sistema; los archivos de una base de datos son distinguibles fácilmente de los archivos de otra base de datos; Los Control files, Redo Log files y datafiles son distinguibles perfectamente de otros archivos de su tipo; y la asociación de un datafile con su respectivo tablespace también es identificable fácilmente.

La siguiente regla contempla este requerimiento:

OFA # 6.

Los nombres de los archivos de la base de datos deben considerar el siguiente patrón:

Nombre	Tipo de archivo
/***pm***/***q***/***d***/control***n***.ctl	control file
/***pm***/***q***/***d***/redo***n***.log	redo log files
/***pm***/***q***/***d***/***tn***.dbf	datafiles

Donde:
pm es el punto de montaje,
q es la cadena que denota la separación de los archivos de datos de Oracle,
d es el nombre de la base de datos,
t es el nombre del tablespace asociado al datafile,
n es el caracter de longitud fija que identifica al tipo de datafile.

NOTA: Nunca guardar algún otro archivo diferente a los mencionados en el directorio ***/pm/q/d***.

Estándar para nombrar a los archivos de la base de datos.

2.2.5 Tablespaces

Un tablespace es la estructura a través de los cuales se relaciona la estructura lógica de la base de datos con archivos a nivel sistema operativo. El concepto de tablespace es un término introducido desde Oracle6 para nombrar a una entidad que relaciona varios segmentos de la base de datos a múltiples archivos de sistema operativo.

Partición de segmentos.

Los factores que afectan la decisión de separar los segmentos de Oracle en diferentes tablespaces incluyen:

- **Fragmentación:** El borrado continuo de segmentos ocasiona la fragmentación del espacio libre del tablespace provocando con ello la degradación del desempeño en la base de datos y otros problemas asociados.

- **Distribución de I/O:** A nivel tablespace el DBA puede determinar que conjunto de segmentos de archivos de sistema operativo se ocuparán. Para segmentos cuyos requerimientos de I/O son muy competidos, los creadores de una base de datos pueden hacer posible el balanceo de las cargas de I/O a través de componentes de hardware.

- **Necesidades de administración:** A nivel tablespace el administrador puede especificar la colección de segmentos a ser respaldados y puede restringir un privilegio de usuario para administrar el espacio de la base de datos.

El siguiente requerimiento fundamenta la regla OFA a continuación descrita:

Requerimiento 11. El contenido del tablespace debe separarse para *(a)* Disminuir la fragmentación del espacio libre del tablespace, *(b)* Disminuir la contención por requerimientos de I/O, y *(c)* Aumentar la flexibilidad para la administración.

El equivalente de este requerimiento es la siguiente regla OFA:

OFA # 7.

Separar los grupos de segmentos según su tiempo de vida, demandas de I/O y frecuencia de respaldo entre diferentes tablespaces. Para cada base de datos Oracle, crear los siguientes tablespaces especiales agregándolos a aquellos requeridos por las aplicaciones.

Tablespace	Tipo
SYSTEM	Solo para segmentos del diccionario de datos.
TEMP	Solo para segmentos temporales.
RBS	Solo para segmentos de Rollback.
TOOLS	Solo para segmentos de propósito general.
USERS	Solo para segmentos de usuarios.

Tablespaces para el estándar de OFA.

Nombre para los tablespaces.

El estándar de OFA obliga a nombrar los tablespaces en base al nombre de los datafiles asociados, esto significa que la restricción en la longitud del nombre para los datafiles también afecta a los tablespaces. Para hacer compatible esta característica con otros sistemas UNIX se recomienda una longitud no mayor a 14 caracteres y en casos de Windows por convención hacerlo no mayor a 8 posiciones.

El nombre estándar para los tablespaces se define como sigue:

OFA # 8.

El nombre de los tablespaces debe de tener 8 o menos caracteres. El número total de tablespaces en una base de datos es generalmente de 100 o menos. Un DBA competente debe ser capaz de distinguir perfectamente un tablespace de cualquier otro objeto de la base de datos.

2.2.6 Dispositivos raw y la Lectura/Escritura en buffers

El dispositivo raw es una partición desmontada de disco en el que Oracle puede leer y escribir sin incurrir en desbordamiento del buffer de I/O manejado por UNIX. El dispositivo raw tiene un tamaño fijo ya que permite asignar una partición completa de un disco. A continuación se presenta una serie de beneficios que aporta la utilización de dispositivos raw:

- Si se utiliza Oracle Parallel Server en un cluster de UNIX en varios nodos que manipulan una sola base de datos de un sistema de discos compartidos, se deben de utilizar raw devices.

- Algunas plataformas ofrecen la capacidad de I/O asíncronas para mejorar el desempeño. Las I/O asíncronas generalmente están disponibles en dispositivos raw independientes del file system de Oracle.

- El uso de dispositivos raw ayudará a aumentar el flujo de los procesos que con frecuencia esperan I/O o que utilizan constantemente tiempo de CPU.

- Si se cuenta con particiones de disco variables, el uso de dispositivos raw para los archivos de Redo Log son benéficos debido a que el acceso de escrituras muy intensas son efectuadas con mayor rapidez.

En contraparte también se tienen definidos una serie de costos involucrados con el uso de dispositivos raw:

- Un dispositivo raw requiere de mayor administración que un file system de UNIX.

- Las I/O en un dispositivo raw solo mejoran un porcentaje muy pequeño, por lo que es necesario tomar en cuenta los puntos siguientes:

    - Diseñar y construir la aplicación lo mejor posible para disminuir I/O.

    - Configurar el sistema operativo y los parámetros de la instancia para que Oracle pueda operar lo más eficientemente posible.

- Balancear las cargas de I/O en discos suficientes.

Para la utilización de dispositivos raw es necesario tomar en cuenta el siguiente requerimiento:

Requerimiento 12. Debe ser posible afinar las cargas de I/O a través de todos los discos disponibles en el sistema, incluyendo aquellos discos que almacenan datos de Oracle en dispositivos raw.

La siguiente regla OFA enuncia este requerimiento:

OFA # 9.

Elegir un tamaño pequeño de raw devices para ser utilizados por los archivos de la base de datos.

Cuando se decida la utilización de dispositivos raw, no debemos ignorar la posibilidad de usarlos solamente en algunos archivos de la base de datos, un sistema híbrido puede ofrecer la mejor combinación de bajo costo con mayores ganancias.

2.2.7 Archivos de administración para el Oracle parallel server

El uso de parallel server agrega mayores tareas administrativas al DBA, debido a que el ambiente del parallel server necesita de la identificación constante de las instancias involucradas en esa configuración. De esta característica se deriva el siguiente requerimiento:

Requerimiento 13. Los archivos específicos de la administración de la base de datos deben almacenarse en un lugar central accesible a todos los administradores de las instancias configuradas en parallel server, y los datos administrativos específicos de cada instancia deben distinguirse por el nombre del archivo asociado a la instancia sobre la cual se esta trabajando.

La principal complicación de los ambientes en parallel server se derivan de la administración simultánea de dos o más instancias a partir de algún nodo en el cluster. A partir de este requerimiento se deriva la siguiente regla OFA.

OFA # 10.

Si se utiliza parallel server, seleccionar exactamente un nodo N para actuar como home de administración del cluster para almacenar la estructura de directorios mencionada en la regla OFA # 5. Tomando nuevamente a h como el directorio home del propietario del software de Oracle en un nodo N, crear un directorio para cada instancia que acceda a la base de datos d dentro de los directorios adump, bdump, cdump, logbook, pfile y udump.

2.2.8 Puntos de montaje para VLDB

Una *VLDB (Very Large DataBase)* es una base de datos que cuenta con un tamaño considerablemente grande a medida que se va haciendo más compleja la administración. En este caso se pueden hacer varias sugerencias con respecto a los puntos de montaje. La mayoría de los sistemas deben de cumplir la regla *OFA # 1.* Solamente una VLDB puede considerar el hecho de no tomar en cuenta la regla *OFA # 1* y considerar la siguiente regla:

OFA # 11.

Si se cuenta con suficiente hardware para garantizar que cada controladora de disco (no particiones de disco) contendrá archivos de la base de datos para una sola aplicación, y si se puede dedicar un número suficiente de controladoras para cada base de datos asegurando que no se presentarán cuellos de botella, nombrar a los puntos de montaje mediante el siguiente patrón */qdm*, donde *q* es la cadena que denota que tipo de dato Oracle será almacenado en él, *d* es el valor del parámetro del init<NOMBRE_DE_LA_INSTANCIA>.ora llamado db_name y, *m* es el identificador único de longitud fija que distingue un punto de montaje de otro.

3 Fallas comunes en las bases de datos y la elaboración de planes de contingencia.

3.1 Respaldos

Nuevamente regresando a la administración de bases de datos relacionales con SQL*Server y MySQL así como con Oracle, se aborda el tema de fallas en las mismas, así como las estrategias de respaldos y recuperación que podemos seguir en caso de afectación a nivel lógico o físico.

3.1.1 Tipos de fallas en la base de datos

Una de las responsabilidades del administrador de la base de datos (DBA) es prepararse para una posible falla de hardware, software, proceso, red o sistema. Si alguna falla afecta la operación normal de la base de datos es necesario efectuar un proceso de recuperación de la base de datos a la operación normal tan pronto como sea posible.

Muchos problemas pueden afectar y/o detener la operación de la base de datos. A continuación se describen los tipos de fallas más comunes, para algunos de estos problemas la recuperación es automática y requieren poca o ninguna participación por parte del DBA.

- **Errores de usuario.** Un administrador puede hacer muy poco para prevenir los errores de usuario; por ejemplo, el borrado de registros, truncar tablas o borrarlas en forma accidental. En la mayoría de los casos, los errores de usuario pueden prevenirse con una buena capacitación y/o administración de herramientas de uso de base de datos.

- **Fallas de sentencia.** Ejecutar un comando que demande una gran cantidad de espacio en la base datos y no haber previsto con anterioridad que este no sería suficiente para alojar la información, al intentar realizar un insert, el RDBMS marcará un error debido a que no hay espacio para realizar esa operación. Por lo tanto, existe una falla de sentencia. Las fallas de sentencia normalmente no requieren de recuperación por parte del administrador, usualmente los sistemas son capaces de manejar estos errores y se pueden evitar con una estrategia de buenas prácticas de administración.

- **Fallas de proceso.** Una falla de proceso de la base de datos, por ejemplo una desconexión anormal o la terminación de un proceso por alguna causa ajena a la operación es resuelta por procesos controlados por la misma base de datos aplicando rollback a todas aquellas transacciones que intervinieron en el proceso de tuvo la falla. Si el proceso que falló es un proceso crítico de control de la instancia la operación normal se ve afectada y entonces la base de datos puede colapsar inmediatamente o incluso podría ser necesario intervención humana para dar de baja manualmente la base de datos y recuperarla lo más pronto posible.

- **Fallas en la base de datos.** Estas fallas normalmente se deben a problemas de hardware, fallas de corriente o problemas del software de sistema operativo. Al reiniciar

la operación de la base de datos por parte del administrador en automático se inicia un proceso de recuperación para mantener la consistencia de la información.

- **Fallas de discos.** Esta falla se presenta cuando un disco esta dañado físicamente e impide la lectura o escritura de un archivo de la base de datos. Todos los archivos de la base de datos son vulnerables a fallas de disco. La operación de la base de datos después de una falla de disco requiere usualmente un proceso de recuperación y restauración por parte del DBA.

Si la falla se deriva en uno o más discos y se dañan archivos los cuales tienen algún tipo de espejo (mirroring) (6), la base de datos puede seguir operando sin interrupción en la mayoría de los casos. Si no se tiene una copia en espejo, la base de datos se detiene perdiendo los datos y se requerirá entonces un proceso de recuperación en primera instancia de los discos haciendo una restauración de su configuración y formato por un administrador de sistema operativo y posteriormente se requerirá la intervención de un DBA que ejecutará las tareas necesarias para regresar la base de datos a su funcionamiento normal.

Existen archivos que se mantienen como copias por ser los que se encargan de tener la información de la constitución de una base de datos, usualmente cuando se hace la creación de una de ellas los archivos son creados como copias que ayudan a reemplazar los archivos dañados por una copia similar. Esto requerirá de la intervención del DBA ya que ninguna base de datos es capaz aún de hacer un reemplazo automático.

Si los discos dañados contienen los archivos de datos y se tienen implementadas estrategias de respaldo de transacciones, los mensajes de error son enviados a archivos de trace avisando a la base de datos alguna situación anormal quedando automáticamente inhabilitados para seguir operando adecuadamente. La estrategia de respaldo de transacciones servirá para regresar a la base de datos a operar hasta un momento antes de la falla minimizando o evitando la pérdida de información

En caso de que el archivo dañado sea el que se utiliza para almacenar el diccionario de datos será necesario que la base de datos sea dada de baja y entonces se ejecutarán los procedimientos disponibles de recuperación donde el DBA demostrará su preparación para estos casos ya que parte del objetivo adicional a restaurar los datos será disminuir al máximo el tiempo de caída de los sistemas.

Existen problemas de disco temporales por pérdida momentánea de sincronización (7), si este fuera el caso la base de datos podrá subsistir sin mayor problema y operará normalmente sin embargo los administradores deberán hacer actividades preventivas y de revisión a los discos que hayan presentado este problema para evitar una corrupción de datos que lleve a una tarea de recuperación muy laboriosa.

Para las bases de datos relacionales que he tomado como ejemplo para este trabajo (Oracle, MySQL y SQL Server) se verá de manera descriptiva algunas funcionalidades que ayudan a mitigar los problemas y fallas comunes dentro de ellas. Estas características se describirán a continuación:

3.1.2 Modo archive y modo no archive en bases de datos Oracle.

Es importante considerar los siguientes puntos para decidir el modo ARCHIVE o NO ARCHIVE en una base de datos Oracle:

En los eventos de falla de base de datos, el DBA puede recuperar todas las transacciones en la base de datos al utilizar el modo ARCHIVE. Sin embargo si se utiliza el

modo NO ARCHIVE solo puede hacerse la recuperación de la base de datos hasta el punto en que se realizó el último respaldo físico completo de los archivos de la base de datos (full-backup), o bien, mediante la combinación del full-backup y un archivo de export.

Si la base de datos esta operando en modo ARCHIVE, la base de datos completa puede abrirse y estar disponible para uso normal mientras se realizan los respaldos de los Redo Log files.

Si la operación de la base de datos será de 24 horas al día los siete días de la semana y no existe tiempo de efectuar un respaldo por medio de sistema operativo o mediante las utilerías de Oracle export/import, entonces es recomendable utilizar el modo ARCHIVE.

Si la operación de la base de datos permite algún tiempo para realizar respaldos físicos de la base de datos, entonces se podrá implementar una estrategia de respaldo en frío combinada con la utilería de Oracle export/import.

Para poder efectuar cualquier tipo de recuperación se requiere de por lo menos un respaldo físico confiable en frío (full-backup) de los archivos de la base de datos como punto inicial y primordial en este proceso.

Ventajas y desventajas del modo archive.

Las ventajas y desventajas de tener la base de datos operando en modo ARCHIVE inciden en el tiempo de recuperación que se dispone para tener la base de datos operando normalmente y el tiempo que es requerido para realizar los respaldos pertinentes de los archivos de la base de datos.

Ventajas:

- Reduce el costo por perdida de información.
- Recuperación de información hasta la última transacción salvada.
- Respaldos de los archivos de datos sin detener la operación.

Desventajas:

- Administración extra por parte del DBA para controlar los Redo Log files.
- La disponibilidad permanente de una unidad de cinta dedicada a archivar los Redo Log files, o el espacio requerido en disco para estos archivos.

*3.1.3 Bitácora de Transacciones en bases de datos SQL*Server.*

Para el caso de bases de datos SQL*Server existe una facilidad denominada Bitácora de Transacciones – “transaction log”- , el cual mantiene la secuencia de los registros dentro de esos archivos. En situaciones de falla estos archivos de transacciones se van restaurando uno a uno a través de una secuencia definida o considerando un tiempo especifico hasta donde debe aplicarse la recuperación. De no estar habilitada la funcionalidad de Bitácora de transacciones no será posible recuperar la base de datos sin perdida de información.

Este tipo de estrategias se debe reforzar con respaldos completos de la base de datos y con estrategias para ir respaldando de forma diferencial la información que va cambiando a partir del último respaldo completo que se haya hecho de la información. Cuando las bases de

datos son de misión crítica no es factible escatimar en ir mejorando las estrategias de recuperación de datos ya que el costo de mantener una base de datos caída puede ser extremadamente elevado.

Ventajas y desventajas de mantener la Bitácora de Transacciones.

De la misma manera que en Oracle, el mantener la facilidad de Bitácora de Transacciones en SQL*Server influirá directamente en el tiempo de recuperación que se dispone para tener la base de datos funcional y con la menor o nula cantidad de datos perdidos. Las ventajas son muy similares en todas las plataformas de bases de datos por lo cual se nombran con las más trascendentes:

Ventajas:

- Reduce el costo por perdida de información.
- Recuperación de información hasta la última transacción salvada.
- Respaldos de los archivos de datos sin tener que parar la operación.

Desventajas:

- Operación adicional por parte del DBA para controlar los archivos de Bitácoras de Transacciones.
- La disponibilidad permanente de una unidad de cinta dedicada a almacenar los archivos de Bitácoras de Transacciones, o el espacio requerido en disco para estos archivos.

3.1.4 Bitácora Binaria en bases de datos MySQL.

Con el paso de tiempo las bases de datos relacionales más robustas van siendo capaces de ir igualando a aquellas que han sido pioneras en algunas funcionalidades, el caso de MySQL no es la excepción y a pesar de ser una base de datos de código abierto también cuenta con la posibilidad de habilitar funcionalidades que permitan una recuperación en caso de fallas hasta llegar a un punto en el tiempo. Esto se logra a través de habilitar la opción de Bitácora Binaria.

Ventajas y desventajas de la Bitácora Binaria.

Una vez más se comprueba que cuando se habla de tecnología cada vez las diferencias son menores así que en cuanto a ventajas y desventajas de utilizar las opciones disponibles de recuperación hasta un tiempo antes de la falla no es la excepción a la regla por lo cual estamos hablando que las más significativas se mantienen.

Ventajas:

- Reduce el costo por perdida de información.
- Recuperación de información hasta la última transacción salvada.
- Respaldos de los archivos de datos sin necesidad de detener la operación.

Desventajas:

- Operación extra por parte del DBA para controlar los redo log files.

- La disponibilidad permanente de una unidad de cinta dedicada a archivar lo Redo Log files, o el espacio requerido en disco para estos archivos.

3.1.5 Estrategias de respaldo

Nuevamente a partir de esta sección y hasta terminar el capitulo hablaré en especifico de posibles estrategias de respaldo considerando a Oracle para ejemplificar de mejor manera los pasos a seguir ya que es la base de datos sobre la cual he desarrollado mi experiencia profesional. Al final del capítulo mostraré un comparativo global donde se podrá observar las opciones que presentan tanto MySQL como SQL*Server.

Antes de crear la base de datos de producción, debemos decidir cual será la política de respaldo a seguir. Para determinar esta política se puede considerar lo siguiente:

- ¿Es aceptable la pérdida de datos en caso de que se suscitará una falla? Si no es aceptable la pérdida de información en algún evento de falla, entonces la base de datos deberá operar en modo ARCHIVE. Se recomienda el uso de archivos espejo en discos diferentes para el caso de los Redo Log files y utilizar varias copias de los Control files en diferentes discos. Si es aceptable la pérdida de información entonces la base de datos puede operar en modo NO ARCHIVE.

- ¿La base de datos necesita estar disponible las 24 horas los 7 días de la semana? Si es así, no se recomienda operar en modo NO ARCHIVE debido a que se requiere efectuar respaldos físicos de la base de datos constantemente y obstruiría la operación de la base de datos ya que estos respaldos se harían cuando la base de datos pudiera darse de baja.

Estrategias de respaldo utilizando modo NO ARCHIVE.

Si la base de datos está operando en modo NO ARCHIVE, los Redo Log files no son almacenados. La única protección de la cual se dispone es el respaldo completo en frío más reciente de la base de datos. ¿Cuando efectuar los respaldos?

Crear un plan para realizar los respaldos regularmente de acuerdo a la cantidad de información que puede perderse de caso de falla y que podría ser recuperada mediante una recaptura de la misma.

- Realizar un respaldo siempre que se modifique la estructura física de la base de datos.

- Ajustar el plan a los horarios en que podría darse de baja la base de datos a respaldar.

Estrategias de respaldo utilizando el modo ARCHIVE.

Si la base de datos está operando en modo ARCHIVE, los Redo Log files son archivados en un dispositivo de almacenamiento determinado previamente quedando protegidos contra cualquier falla en la base de datos. La estrategia que podría utilizarse puede determinarse por lo siguiente:

- Cuando la base de datos es creada con el modo ARCHIVE o posteriormente es activada de la misma manera, es necesario realizar un respaldo completo de la base de datos. Este respaldo el fundamental para las recuperaciones posteriores en caso de ser

necesario. Otro punto muy importante es asegurarse de que la base de datos esté operando en modo ARCHIVE antes de realizar el respaldo completo, de otra forma, el respaldo realizado se llevará la definición de NO ARCHIVE y no podrá recuperarse la base de datos en caso de algún problema.

- Los respaldos completos no son requeridos, si la base de datos permanecerá abierta todo el tiempo.

- Respaldar datafiles para efectuar la recuperación.

- Cada vez que sea efectuado un cambio físico en la estructura de la base de datos, efectuar un respaldo del Control file utilizando el comando ALTER DATABASE con la opción de BACKUP CONTROLFILE. No usar utilerías del sistema operativo para respaldar los Control files a menos que se realice un respaldo completo de la base de datos en ese mismo momento.

- En caso que los archivos de ARCHIVE estén dirigidos a disco, estos tendrán que respaldarse periódicamente a un dispositivo de almacenamiento externo.

3.1.6 Procedimiento para respaldar en modo no archive

Los procesos de respaldo en modo NO ARCHIVE se realizan a nivel de archivos (COLD BACKUP) o con la utilería de Oracle export.

Respaldo en frío (COLD BACKUP).

Es el respaldo que se efectúa mediante la utilización de las herramientas del sistema operativo (tar, cpio, dd). Para este tipo de respaldo se requiere que la base de datos esté cerrada (SHUTDOWN).

I. Dar de baja la(s) base(s) de datos que se va(n) a respaldar.

Desde el prompt de sistema operativo dar el comando:

$. oraenv <enter>
ORACLE_SID=[test]?_

Si esta instancia es la que queremos dar de baja para respaldarla, entonces se teclea <enter>, de lo contrario, proporcionar el nombre de la instancia adecuada y presionar <enter>.

Verificar que ningún usuario este firmado en la base de datos que se va a dar de baja. Esta actividad puede realizarse a través de SQLPLUS monitoreando las sesiones a nivel base de datos; cuando ya no existan usuarios en ella, se puede proceder a dar de baja.

$ sqlplus '/ as sysdba' <enter>
SQL> shutdown <enter>

NOTA: Para dar de baja otras instancias, proporcionar el nuevo nombre cuando se efectúe el comando. oraenv desde sistema operativo.

II. Con comandos y utilerías del sistema operativo realizar el respaldo a la unidad de almacenamiento seleccionada.

Los archivos de la base de datos que deben respaldarse son:

- Datafiles
- Redo Log files
- Control files
- Archivo de parámetros init<NOMBRE_DE_LA_INSTANCIA>

Respaldo utilizando herramientas de Oracle.

Oracle cuenta con la utilería **export** y su complemento de recuperación **import**. La utilería de export de Oracle graba los datos de la base de datos en un archivo del sistema operativo con un formato especial que solo puede ser leído por la utilería import. Básicamente se utiliza como complemento de los respaldos en frío y para mover información de una base de datos a otra.

Con este tipo de respaldos la base de datos puede estar abierta y disponible para todos los usuarios, sin embargo, es recomendable que los respaldos sean efectuados sin usuarios conectados ya que al momento de realizar el proceso export y éste se encuentra leyendo una tabla importante y el usuario no ha dado commit a sus transacciones, estas no serán respaldadas en el archivo. El modo de utilización de la base de datos es de acceso restringido.

La utilería export ofrece tres modos diferentes de respaldar información:

- **Full Backup.** Exporta todos los objetos de la base de datos.
- **Usuario.** Exporta todos los objetos pertenecientes al usuario.
- **Tablas.** Exporta todos los objetos específicos de una tabla(s) que pertenecen a un usuario determinado.

También se ofrecen tres tipos de respaldos con export partiendo de un full backup.

- **Incrementales.** Exporta los objetos que hayan cambiado desde el último respaldo incremental, acumulado o completo.
- **Acumulados.** Exporta los objetos que han cambiado desde el último respaldo acumulado o completo. Es recomendable efectuar este tipo de respaldo por lo menos una vez por semana para disminuir el tamaño de los respaldos incrementales.
- **Completos.** Se exportan todos los datos de la base de datos independientemente si han o no han cambiado.

El export genera un archivo a nivel sistema operativo con la extensión .dmp que contiene toda la estructura de la base de datos y su información correspondiente. El tamaño del archivo depende del volumen de información, pero aproximadamente es entre un 10% y 15% del total de datos que tenga la base de datos.

Por ejemplo, si la tabla DBA_EXTENTS suma un total de 850MB por todos sus tablespaces asociados, entonces el archivo de export será de aproximadamente entre 85 MB y 127 MB. Este archivo puede comprimirse utilizando comandos del sistema operativo.

$ compress nombre_archivo.dmp

El porcentaje de compresión del archivo depende del comando de sistema operativo y puede reducirse a un 20% aproximadamente. Posteriormente este archivo puede respaldarse en otro medio de almacenamiento.

IMPORTANTE: No olvidarse de ejecutar el comando*. oraenv* para asegurar que estamos haciendo el respaldo de la base de datos correcta.

3.1.7 Procedimiento de respaldo en modo archive

Habilitar el modo archive en una base de datos nos da una ventaja considerable ya que a través de esta facilidad podemos hacer recuperaciones en caso de falla hasta un momento antes de la falla sin perder datos y asegurando la consistencia de información.

Base de datos en modo archive.

Para identificar en que modo se encuentra la base de datos, podemos apoyarnos de las siguientes instrucciones:

I. Conectarse a SQLPLUS

$ sqlplus '/ as sysdba'

II. Consultar el modo de operación

SQL> archive log list
SQL>exit

Con esta consulta aparecerá una lista que nos indica el modo en que esta operando la base de datos, si el respaldo es automático o manual, el destino de los archivos de ARCHIVE, la secuencia de los Log en línea y el número de secuencia que es ocupado actualmente.

Antes de realizar cualquier cambio en la base de datos, específicamente habilitar el modo ARCHIVE, es necesario efectuar un respaldo en frío de todos los archivos que componen la base de datos. Si se cuenta con una base de datos de pruebas, se recomienda realizar los cambios en ella para familiarizarse con los procedimientos de respaldo y mantenimiento antes de llevarlos a un sistema en producción. Para activar el modo archive es necesario modificar los parámetros del archivo INIT <NOMBRE DE LA INSTANCIA>.ora mencionados a continuación:

LOG_ARCHIVE_START=TRUE (indica que el modo ARCHIVE será automático).
LOG_ARCHIVE_DEST=ruta de respaldo de los archivos.
LOG_ARCHIVE_FORMAT=formato del archivo.

Pasos para activar el modo ARCHIVE.

I. Dar de baja la base de datos.

Deberá darse de baja la base de datos para poder efectuar el respaldo en frío de los archivos que componen la base de datos.

$ sqlplus '/ as sysdba'
SQL> shutdown normal
SQL> exit

II. Respaldo en frío (Cold Backup) de la base de datos.

Con comandos de sistema operativo (tar, cpio) efectuar la protección de todos los archivos que componen la base de datos.

*$ tar cvf /dev/rmt0 /u02/oradata/TEST/**

Donde:

tar	Comando del sistema operativo que se utiliza para condensar archivos de sistema operativo en un solo archivo con este formato.
cvf	Opciones para el comando tar.
/dev/rmt0	Nombre del dispositivo a donde se está enviando el respaldo.
/u02/oradata/TEST/*	Ruta de los archivos pertenecientes a la base de datos que serán respaldados.

III. Modificar los parámetros del init<NOMBRE_DE_LA_INSTANCIA>.ora.

Editar el archivo init<NOMBRE_DE_LA_INSTANCIA>.ora y modificar los parámetros LOG_ARCHIVE_START, LOG_ARCHIVE_DEST y LOG_ARCHIVE_FORMAT.

IV. Montar la base de datos sin abrirla.

$ sqldba lmode=y
SQL> connect internal
SQL> startup mount

V. Activar el modo ARCHIVE.

Una vez que la base a sido montada sin abrirse se debe activar el modo ARCHIVE.

SQL> alter database NOMBRE_DE_LA_BASE_DE_DATOS archivelog;

Para asegurarnos que la base de datos ya esta en modo ARCHIVE:

SQL> archive log list

Que especificará como habilitado el modo archive.

VI. Abrir la base de datos.

Por último, abrir la base de datos y ponerla disponible a todos los usuarios.

SQL> alter database NOMBRE_DE_LA_BASE_DE_DATOS open;

3.1.8 Respaldo con tablespaces en línea

Todos los datafiles de un tablespace individual o un datafile en específico pueden respaldarse, teniendo activado el modo ARCHIVE, mientras que el tablespace y/o los datafiles asociados al tablespace permanezcan en línea. Para efectuar este tipo de respaldos se puede proceder como sigue:

I. Identificar los datafiles asociados a los tablespaces que se van a respaldar.

Los nombres se obtienen de una consulta sobre la vista DBA_DATA_FILES.

II. Marcar el comienzo del respaldo en línea del tablespace.

$ sqlplus '/ as sysdba'
SQL> alter tablespace NOMBRE_TABLESPACE begin backup;

NOTA: Si se olvida marcar el comienzo de un respaldo de tablespaces en línea, los respaldos de datafiles no serán útiles para futuras recuperaciones.

III. Respaldar los datafiles asociados al tablespace.

En este momento podemos respaldar los datafiles que pertenezcan al tablespace en línea del cual marcamos el inicio del respaldo. Efectuar la copia de los datafiles a nivel sistema operativo mediante los comandos tar, cp, cpio o dd de sistema operativo.

IV. Marcar el final del respaldo en línea.

Después de haber copiado los datafiles, se debe indicar el fin del respaldo mediante la ocupación del comando:

SQL> alter tablespace NOMBRE_TABLESPACE end backup;

NOTA: Si por algún motivo se olvidara marcar el final del respaldo y después se procede a dar de baja la base de datos, Oracle asume que es necesario recuperar datos en el siguiente startup. La recuperación requiere de los Redo Log archivados.

Una característica importante de este proceso de respaldo de tablespaces en línea es que se puede efectuar un respaldo de varios tablespaces con la única condición de especificar el inicio y final del respaldo por cada tablespace.

3.1.9 Respaldo con tablespaces fuera de línea

Por medio de esta forma se pueden respaldar tablespaces específicos o todos los tablespaces de la base de datos (excepto el de SYSTEM) mientras estos permanezcan fuera de línea. Los demás tablespaces pueden operar normalmente y estar disponibles para todos los usuarios.

Es importante señalar que el tablespace de SYSTEM o cualquier otro que tenga segmentos de rollback activos no podrán ponerse fuera de línea. Por lo tanto el siguiente procedimiento no podrá aplicarse para todos los tablespaces.

Para efectuar un respaldo con el tablespace fuera de línea se procede de manera similar que un respaldo con tablespaces en línea.

I. Identificar los datafiles asociados.

Los nombres se obtienen de una consulta sobre la vista DBA_DATA_FILES.

II. Poner fuera de línea el tablespace a ser respaldado.

$ sqlplus '/ as sysdba'
SQL> alter tablespace NOMBRE_TABLESPACE offline;

III. Respaldar los Datafiles asociados a nivel sistema operativo.

En este momento podemos respaldar los Datafiles que pertenezcan al tablespace en línea del cual marcamos el inicio del respaldo. Efectuar la copia de los Datafiles a nivel sistema operativo mediante los comandos tar, cp, cpio o dd de sistema operativo.

IV. Poner el tablespace en línea.

$ sqlplus '/ as sysdba'
SQL> alter tablespace NOMBRE_TABLESPACE online;

Al igual que el respaldo de tablespaces fuera de línea, con esta opción es posible respaldar varios tablespaces a la vez.

3.1.10 Respaldo de los archivos de control

Es una tarea primordial hacer respaldo de los archivos de control siempre que se efectúen cambios en la operación de la base de datos, por ejemplo en el caso de cambiar a modo ARCHIVE, o después de hacer cambios a la estructura de la base de datos, como agregar un Datafile, crear un tablespace nuevo, y otras actividades más.

Para respaldar los archivos de control se utiliza la instrucción:

SQL> alter database backup controlfile to NOMBRE.ctl reuse;

Donde:

NOMBRE.ctl	Nombre del archivo de control a respaldar.
reuse	Permite que el nuevo archivo de control sobre escriba en el archivo de control existente.

Cuando se tiene activado el modo ARCHIVE, todas las transacciones son grabadas en archivos con extensión .arc, depositados en la mayoría de los casos en disco, por lo tanto, estos archivos están sujetos a perderse por alguna falla en el disco. Es recomendable respaldar a cinta estos archivos diariamente para garantizar la recuperación de información en caso de ser necesario.

3.2 Recuperación

En todos los sistemas de base de datos, la posibilidad de una falla siempre está presente. Cuando una falla se suscita, la base de datos debe ser recuperada con la mayor rapidez y eficiencia amortiguando de la mejor manera el impacto sobre los usuarios del sistema.

El tiempo y los datos a recuperar dependerán en gran medida del buen control y ejecución de los respaldos. La información que puede ser recuperable se basa en el tipo de respaldo efectuado y en el modo de recuperación a efectuar. Por tal motivo, esta tarea es crucial para el desempeño del sistema y sin dudarlo puede llegar a ser la actividad más crítica en la cual el DBA debe capacitarse adecuadamente.

Otro punto que influye en la acción a tomar para recuperar información de la base de datos es la estrategia seleccionada para respaldar la información.

3.2.1 Procedimiento para recuperar en modo no archive

Cuando la operación de la base de datos se encuentre en modo NO ARCHIVE, la única forma de recuperar nuestra información es a través del último respaldo en frío que se tenga o el último respaldo de export.

Recuperación con respaldo en frío.

I. **Tener o crear la misma estructura de File System en sistema operativo.**

   La estructura a nivel sistema operativo se debe hacer con las herramientas que el mismo sistema operativo nos proporciona, es requisito importante que la estructura sea idéntica ya que de lo contrario el respaldo no funcionará para la recuperación.

II. **Bajar el respaldo de los archivos de la base de datos del dispositivo de almacenamiento donde se encuentren y colocarlos en los File System correspondientes.**

   El respaldo debe haber sido generado exitosamente y debe ser consistente, es decir no debe haber archivos que no hayan sido respaldados al mismo tiempo o con la base de datos disponible a todos los usuarios. Los respaldos pueden estar almacenados en cualquier dispositivo de almacenamiento.

III. **Abrir la base de datos en forma normal.**

   Una vez que ha bajado el respaldo, lo que resta es aplicar los siguientes comandos para poner disponible la base de datos.

   $ sqlplus '/ as sysdba'
   SQL> startup

Recuperación desde un archivo Export.

I. Crear la base de datos.

```
$ . oraenv
$ ORACLE_SID=[base]? test <enter>
$ sqlplus '/ as sysdba'
SQL> startup nomount pfile=/u01/oracle/product/7.3.3/dbs/initTEST.ora
SQL> create database test
2>      maxinstances 8
3>      maxlogfiles 32
4>      maxdatafiles 100
5>      character set "US7ASCII"
6>      datafile '/u02/oradata/test/system01.dbf' size 30M
7>      logfile   '/u02/oradata/test/log01.dbf' size 1M,
8>                '/u02/oradata/test/log02.dbf' size 1M,
9>                '/u02/oradata/test/log03.dbf' size 1M;
```

II. Bajar el archivo de formato dmp en caso de que se encuentre en una cinta u otro dispositivo externo en formato cpio o tar.

$ cpio -icvBdu < /dev/rmt0	*Baja el archivo de Export almacenado en cinta con formato cpio.*
$ uncompress resp.dmp.Z	*Descomprime el archivo en caso de que hubiera sido almacenado a cinta ya comprimido para reducir su espacio.*

En caso de utilizar formato tar en lugar de cpio se utiliza la instrucción siguiente:

```
$ tar xvf /dev/rmt0
```

III. Importar los datos.

Se puede generar un archivo ascii con los parámetros necesarios para el import o se puede teclear el comando directamente desde sistema operativo.

El archivo de parámetros sería algo similar a este:

$ vi impfull.dat	*Edita el archivo impfull.dat.*
userid=system/manager	*Nombre de usuario y password.*
file=resp.dmp	*Nombre del archivo a importar.*
buffer=100000	*Tamaño del buffer a utilizar.*
full=y	*Especifica que es un Import de toda la base de datos.*

y la ejecución desde sistema operativo sería:

$ imp parfile=impfull.dat <enter>	*Parfile especifica el archivo de parámetros, en caso de teclear directamente el comando, entonces se teclear todos los parámetros separados cada uno por un espacio en blanco.*

3.2.2 Procedimiento para recuperar en modo ARCHIVE

Es el método de recuperación más confiable que existe en el RDBMS de Oracle debido a que se pueden efectuar recuperaciones hasta un momento antes de sucedida la falla, y solo se perderá la información sin commit, es decir aquellas transacciones que no fueron completadas y almacenadas en la base de datos.

Recuperación completa con la base de datos cerrada.

I. Corregir el problema que provocó la falla en la base de datos.

Si los archivos están definitivamente dañados, recuperar los archivos desde el respaldo más reciente con que se cuente, no restaurar los archivos que no estén dañados ni ningún redo log file para ahorrar tiempo en el proceso. Si no existiera copia de los archivos, se podrá crear un tablespace vacío y entonces recuperar toda la información aplicando los redo log archivados que sean necesarios.

II. Entrar a SQLPLUS con el usuario sys.

$ sqlplus '/ as sysdba'

III. Montar la base de datos sin abrirla.

SQL> startup mount

IV. Si alguno de los datafiles es ubicado en un disco que no era su ruta original, utilizar el siguiente comando para actualizar los control files conforme se va ejecutando el comando. Realizar esta actividad por cada datafile que se encuentre en una posición diferente.

SQL> alter tablespace nombre_tablespace datafile '/u02/oradata/test/demo1.dbf'
to '/u05/oradata/test/demo1.dbf';

V. Todos los datafiles que serán recuperados deben estar en línea durante el proceso de recuperación completa. Se puede utilizar la siguiente instrucción para asegurarnos de que tengan ese estado.

SQL> alter database datafile 'nombre_Datafile.dbf' online;

VI. Es posible la recuperación de un solo datafile o de todos a la vez mediante la utilización de la sentencia siguiente:

SQL> alter database recover automatic; Para recuperar todos los datafiles.
SQL> alter database recover datafile
2> 'nombre_datafile1', 'nombre_datafile2'; Para recuperar datafiles específicos.

VII. Por último, Oracle comienza a aplicar los redo log necesarios para reconstruir los datos. Una vez concluida la aplicación de los redo log, aparecerá en pantalla el mensaje media recovery complete. Teclear el comando correspondiente para abrir la base de datos a operación normal.

SQL> alter database open;

Recuperación completa con la base de datos abierta.

Cuando uno o más datafiles se encuentran dañados, pero la base de datos puede seguir en operación, entonces puede aplicarse una recuperación con la base de datos abierta; es importante hacer notar que no se puede realizar una recuperación de los datafiles de SYSTEM con la base de datos abierta. Para realizar la recuperación de este datafile es necesaria la recuperación con la base de datos cerrada.

La serie de pasos a seguir para lograr la recuperación con la base de datos abierta son:

I. **Corregir el problema que haya ocasionado la falla.**

   Las fallas pueden ser diversas y de diferentes causas por lo cual es importante tomar en cuenta que no es un método muy usual para recuperar una base de datos.

II. **Levantar la instancia y abrir la base de datos.**

   En ese momento, Oracle detecta los datafiles dañados y lo notifica a través de un mensaje en pantalla poniendo fuera de línea los tablespaces asociados a los datafiles dañados. Para asegurarnos que el tablespace esta fuera de línea teclear:

   $ sqlplus '/ as sysdba'
   SQL> startup
   SQL> alter tablespace nombre_tablespace offline;

III. **Si los archivos se dañaron permanentemente, utilizar los archivos de respaldo más recientes y reemplazarlos. Si no se tuvieran los respaldos, crear un nuevo tablespace con el mismo nombre y la misma ruta que el dañado.**

   Al igual que el primer paso es un método muy riesgoso ya que complica la situación el tener usuarios accediendo a la base de datos y demorará considerablemente la recuperación.

IV. **Si uno o más datafiles fueron reubicados en otros discos, es necesario indicarlo mediante el comando:**

   alter tablespace nombre_tablespace datafile '/u02/oradata/test/demo1.dbf' to '/u05/oradata/test/demo1.dbf' para actualizar de esta manera los control files.

V. **Para recuperar los datafiles asociados al tablespace dañado existen dos opciones que pueden ser consideradas; la primera es utilizar el comando:**

   alter database recover tablespace 'nombre_tablespace';

   o bien, si solo se daño un datafile, entonces utilizar el comando:

   alter database recover datafile 'nombre_datafile.dbf';.

Recuperación parcial.

La recuperación parcial puede ser de tres tipos diferentes: Por cancelación de usuario, Basado en el tiempo o Basado en cambios. Básicamente, este tipo de recuperación parcial se utiliza para recuperar errores de los usuarios como por ejemplo la recuperación de una tabla borrada accidentalmente.

Los puntos a que a continuación se mencionan son los requeridos para efectuar la recuperación parcial de información con la decisión previa del tipo de evento a tomar en cuenta:

I. Verificar que la base de datos esté abierta y disponible para todos los usuarios.

A nivel sistema operativo UNIX, ejecutar el siguiente comando:

$ ps -ef |grep nombre_base_de_datos

Si la base de datos está abierta, aparecerá una lista con los procesos de background levantados por la instancia, los cuales tiene un número de proceso UNIX asignado, los procesos obligatorios que deben observarse en la lista son: PMON, SMON, LGWR y DBWR. En caso de que la base no se encuentre abierta, levantar la base de datos desde SQLPLUS.

$. oraenv
$ sqlplus '/ as sysdba'
SQL> startup

II. Colocar los redo log files requeridos para la recuperación en la ruta especificada por el archivo de parámetros init<NOMBRE_DE_LA_INSTANCIA>.ora en caso de que se encuentren almacenados en algún dispositivo externo de almacenamiento.

III. Determinar el tipo de recuperación a efectuar.

Si la opción es recuperación basada en cancelación de usuario teclear en SQLPLUS la siguiente instrucción:

SQL> alter database recover database until cancel;

De esta forma, cuando sea aplicado el último redo log que contiene la información que se desea recuperar, entonces, desde el prompt se tendrá que teclear la palabra CANCEL para finalizar el proceso.

En caso de aplicar recuperación basada en el tiempo, entonces teclear desde SQLPLUS el siguiente comando especificando el tiempo hasta donde se desee efectuar la recuperación.

SQL> alter database recover database until 'YYYY-MM-DD:24:MI:SS';
SQL> alter database recover database until time '1997-04-15:16:30:40';

Para efectuar la recuperación basada en cambios, teclear:

SQL> alter database recover database until change 20;

El número 20 indica la secuencia del último archivo de redo log a ser aplicado.

IV. Para finalizar este proceso de recuperación, verificar que los cambios se hayan efectuado correctamente y finalmente se da de alta la base de datos nuevamente utilizando este comando:

SQL> alter database open resetlogs / noresetlogs;

En caso contrario, aplicar el respaldo de seguridad para regresar al estado previo a la recuperación y volver a ejecutar nuevamente el proceso, con previa verificación de la situación por la cual los cambios no fueron efectuados correctamente.

Comparativo Oracle, MySQL y SQL*Server para respaldos y recuperación de bases de datos.

Partiendo del hecho que estos tres núcleos de bases de datos cuentan con esquemas robustos para hacer respaldos y sus correspondientes recuperaciones en caso de fallas, se propone una manera muy simple pero efectiva de realizar los respaldos por categorías, donde se hará básicamente un respaldo de Sistema Operativo, de Base de Datos y de aplicaciones, sugiriendo un esquema como este:

Sistema Operativo:

- Respaldo de file system o sistemas de archivos. Se deberá clasificar el respaldo con archivos que se consideren útiles para la recuperación de un equipo o servidor. Por ejemplo: Archivos para reinicio del sistema operativo, archivos de Kernel (librerías y binarios adicionales), scripts y procesos tanto administrativos como para la operación.

Considerando los sistemas operativos, tenemos diversos comandos nativos que nos pueden servir para realizar respaldos sin considerar utilerías costosas, sin embargo los esquemas institucionales si deben considerar esquemas más robustos.

UNIX	Linux	Windows
Ufsdump	ufsdump	zip
Tar	tar	copy
Gzip	gzip	xcopy
Compress	compress	-
Copy	copy	-
Dd	dd	-

Bases de Datos:

Aquí tenemos más diversidad en la forma en que son llamados los archivos de bases de datos, así que pondremos de acuerdo a su núcleo los archivos que se consideran importantes.

- **Oracle:** Archivos de datos o datafiles, de transacciones o archive logs, archivos de redo logs, archivos de control, archivos de parámetros y de configuración de accesos remotos como tnsnames.ora, listener.ora, archivos de respaldos por base de datos como exports y RMAN, y finalmente scripts de procesos, depuración y monitoreo.

- **MySQL:** Archivos de datos, archivos de configuración (MySQL.ini), archivos dump, scripts de procesos, depuración y monitoreo.

- **SQL*Server:** Archivos de datos, archivos de transacciones (transaction logs), scripts de procesos, depuración y monitoreo.

A continuación se muestra una tabla comparativa donde se podrá ver los tipos de respaldo soportados por cada RDBMS:

Oracle	MySQL	SQL Server
Hot/Cold backup	Hot/Cold backup	Hot/Cold backup
RMAN	MySQL Administrator	Microsoft SQL Server Management Studio
export full export incremental export de esquemas export de objetos	mysqldump full mysqldump incremental mysqldump de objetos	Transaction log backup backup de esquemas backup de objetos backup diferencial

Aplicaciones:

Para el caso de las aplicaciones, es importante considerar que los propietarios y fabricantes de las mismas deben proveer la manera más eficiente de respaldar ya que puede bastar con un simple respaldo por comandos de sistema operativo o incluso puede ser necesario respaldar a través de utilerías debido a que los archivos puede estar siendo utilizados por recursos que no permitan hacer un respaldo simple consistente. Así mismo deben considerarse los propósitos de esas aplicaciones como pueden ser servidores Web, servidores de aplicaciones, códigos y programas y utilerías.

4 La afinación de la base de datos.

4.1 Afinación de Base de Datos.

Esta es una actividad primaria para el DBA experimentado pero debido a que consume mucho tiempo y es un proceso constante, le proporciona un valor agregado al perfil de DBA; es una tarea que involucra un conocimiento más profundo de la arquitectura del RDBMS, procesos a nivel base de datos, procesos a nivel sistema operativo, y de las herramientas que sirven como apoyo a la afinación.

Un aspecto muy importante que debe considerarse es la disponibilidad de la base de datos para darla de baja cada vez que se afina algún proceso o parámetro de la base de datos, esto es debido a que esta actividad no comprende una serie de pasos a seguir sino que es una prueba constante en el desempeño hasta considerar que la base de datos a quedado bien afinada.

El RDBMS de Oracle es altamente mejorable, es decir, que puede afinarse considerablemente el desempeño ajustando las aplicaciones que corren en la base de datos, la base de datos misma e incluso en sistema operativo. A todos estos ajustes se les conoce como afinación. La afinación se va efectuando según el monitoreo de la base de datos en horas en que se presenta la mayor carga de trabajo.

Para la afinación se cuenta con cuatro métodos importantes:

- Afinación de las sentencias de SQL y las aplicaciones.
- Afinación de la memoria.
- Afinación de Entradas/Salidas.
- Afinación de contención.

4.1.1 Afinación de Sentencias SQL y las Aplicaciones.

Antes de afinar la base de datos como tal, es necesario considerar la afinación de las aplicaciones por medio del ajuste de las sentencias de SQL favoreciendo la rapidez en la ejecución de los procesos efectuados por la aplicación. Para lograr un mejor desempeño en las aplicaciones pueden tomarse en cuenta las siguientes características:

- El diseño de datos.
- El optimizador en el caso del RDBMS de Oracle.
- Los índices.

- El manejador de candados a nivel registro.
- El PL/SQL.
- El generador de secuencias.
- Los clusters.
- El procesamiento de arreglos.

4.1.2 Afinación de la Memoria.

La asignación apropiada de los recursos de memoria para las estructuras de memoria de Oracle puede tener un gran impacto en el desempeño. En este método se puede decidir la cantidad de memoria que debe asignarse a las siguientes estructuras:

- Áreas de contexto.
- El caché del diccionario de datos.
- El buffer caché.
- Cuando se ajustan de manera adecuada estos parámetros podemos obtener los siguientes beneficios:
- Aumento en el desempeño
- Reducción del parseo de las sentencias de SQL.
- Reducción de la paginación y uso de área de swap.

4.1.3 Afinación de Lecturas/Escrituras.

Las lecturas/escrituras de disco tienden a reducir el desempeño de muchas aplicaciones, sin embargo, el RDBMS de Oracle esta diseñado para que su desempeño no se vea limitado por las entradas/salidas. La afinación de estos procesos involucra:

- La distribución de las entradas/salidas para evitar la contención de disco.
- El almacenamiento de datos en bloques para un mejor acceso.
- Crear extents lo suficientemente grandes para los datos.

4.1.4 Afinación de Contención.

Los procesos concurrentes de múltiples usuarios de Oracle pueden crear contención por los recursos de Oracle y puede causar que los procesos tengan que esperar hasta que el recurso que solicitan este disponible. En este caso debe reducirse la contención de:

- Los segmentos de rollback.
- Los latches de los redo log buffer.
- Modificación de registros.
- Transaccionalidad en tablas e índices.

Al terminar con el ajuste en la contención es importante volver a evaluar el desempeño de la base de datos y decidir si es necesario volver a afinar.

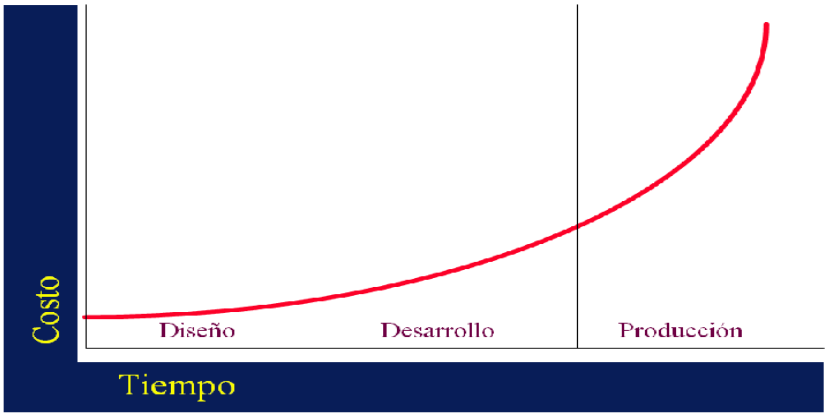
4.1.5 ¿Cuándo se Debe Afinar?

La mayoría de la gente cree que el proceso de afinación inicia cuando los usuarios se quejan del pésimo tiempo de respuesta de la base de datos y las aplicaciones, en ese instante es muy tarde para aplicar algunas de las estrategias más efectivas de afinación. Solo queda considerar en caso de no ser posible un rediseño de la aplicación la afinación de memoria y de los procesos de lectura/escritura. Oracle proporciona muchas características que pueden aumentar el desempeño cuando son utilizadas propiamente en un buen diseño del sistema.

El diseño de la aplicación necesita cubrir las expectativas de desempeño de la aplicación en la fase de diseño. Durante la fase de diseño y desarrollo, el diseñador debe considerar que características de un RDBMS pueden beneficiar al sistema.

El buen diseño de un sistema puede eliminar el costo en el tiempo de vida de la aplicación. En la figura # 1 se muestra el costo involucrado según la fase en la que se efectúa una afinación.

Figura # 1. Costo de la afinación durante el tiempo de vida de una aplicación.



CYRAN, Michelle. Oracle 8i Designing and Tuning for Performance Release 2 (8.1.6) Diciembre 1999. On-Line Documentation CD-ROM Release 2 (8.1.6) for Microsoft Windows. Part # A84050-01.

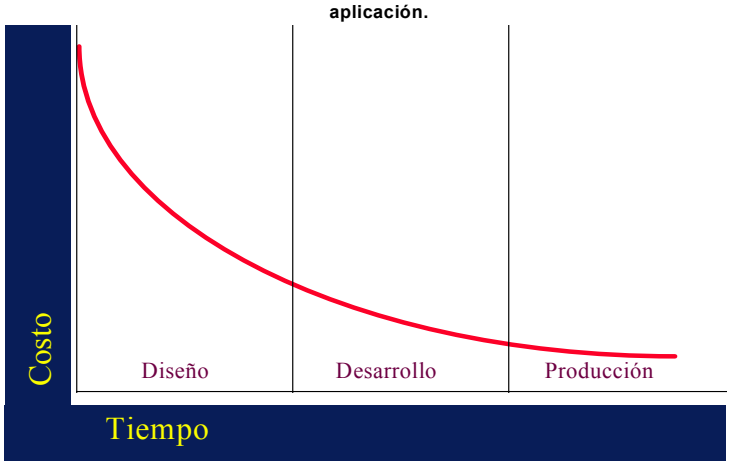
4.1.6 Afinación al Diseñar o Desarrollar un Sistema.

Un sistema bien diseñado puede prevenir problemas de desempeño después de poner en marcha una aplicación. Los desarrolladores y diseñadores de aplicaciones deben entender el mecanismo de procesamiento de las consultas de Oracle para poder escribir correctamente una sentencia de SQL. Al diseñar un sistema, se pueden considerar los siguientes puntos para obtener el mejor desempeño posible:

- Eliminar el tráfico en la red en aplicaciones cliente/servidor.
- Utilizar las aplicaciones apropiadas para el sistema a desarrollar (Procesamiento en paralelo, bases de datos distribuidas, particiones, etc.)
- Utilizar las opciones de candados predeterminados a menos que la aplicación tenga necesidades específicas.
- Registrar los módulos de la aplicación en la base de datos para poder tener pistas del desempeño a partir de módulos base.
- Elegir el mejor tamaño de bloque para la base de datos.
- Distribuir los datos de manera tal que aquellos utilizados por un nodo estén almacenados localmente en ese mismo nodo.

En la figura # 2 se muestra el beneficio involucrado de una afinación en una fase temprana.

Figura # 2. Beneficio de la afinación durante el tiempo de vida de una aplicación.



CYRAN, Michelle. Oracle 8i Designing and Tuning for Performance Release 2 (8.1.6) Diciembre 1999. On-Line Documentation CD-ROM Release 2 (8.1.6) for Microsoft Windows. Part # A84050-01.

4.1.7 Afinando un Sistema en Producción.

A través de los siguientes aspectos se describe una forma rápida y fácil para la identificación de los "cuellos de botella" y así corregir este problema. Este método requiere de la comprensión de la arquitectura y características de los RDBMS que se utilicen, estar familiarizado con los conceptos del RDBMS antes de intentar afinar un sistema. Por medio de estos puntos se puede afinar adecuadamente un sistema:

- Afinar el hardware y software a nivel sistema operativo. Antes de afinar adecuadamente a un RDBMS, primero debemos asegurarnos que el sistema operativo esta funcionando con un desempeño apropiado, este tipo de trabajo debe efectuarse conjuntamente con el administrador del sistema para asegurar que se le asignen a las bases de datos los recursos de sistema operativo apropiados.

- Identificar los "cuellos de botella" consultando continuamente las tablas de sesiones. Usualmente las bases de datos presentan los eventos que provocan la espera de una sesión y es posible identificar los "cuellos de botella" de manera rápida.

- Determinar y corregir el problema. Cada problema de desempeño tiene una causa y solución únicas y quizás sea necesario consultar otras vistas dinámicas para encontrar la causa de estos y darles solución. Los problemas de desempeño tienden a manifestarse según la siguiente categoría:

    - CPU

    - Memoria

    - Entradas/Salidas

    - Contención por latches (8) u otras estructuras.

Una de las facilidades con las que ahora cuentan las bases de datos para determinar el funcionamiento es el monitoreo por medio de herramientas gráficas las cuales son de gran ayuda para poder identificar de manera más amigable la información obtenida a través de las vistas dinámicas de comportamientos tanto el tiempo real como en históricos que sirvan de análisis a los administradores.

4.1.8 Diseño Adecuado de una Aplicación.

Existen diversos tipos de aplicaciones que pueden construirse con los motores de bases de datos existentes.

Las aplicaciones **Online Transaction Processing** (OLTP) tienen un alto índice de procesos, inserciones y actualizaciones, estas aplicaciones se caracterizan por tener un crecimiento constante en el volumen de datos que son consultados por cientos de usuarios concurrentes.

En este tipo de aplicaciones se pueden afinar las siguientes estructuras:

- Segmentos de rollback

- índices, clusters y hashing
- Transacciones discretas
- Tamaño del bloque de datos
- Asignación dinámica de espacio para tablas y segmentos de rollback
- Monitores de transacciones y el multi-threaded server
- El shared pool
- Sentencias SQL
- Constraints de integridad
- Arquitectura cliente-servidor
- Procedimientos, packages y funciones

Otro tipo de aplicaciones existentes son las de **soporte de decisiones** como las de minería de datos usualmente consumen una gran cantidad de recursos tanto de memoria, procesamiento y de accesos a datos en disco y en caché. En ellas se ejecutan consultas sobre una gran cantidad de datos almacenados donde los históricos juegan un papel importante. Las personas encargadas a la toma de decisiones utilizan estas aplicaciones para determinar que estrategia debe seguir la organización basada en la información disponible. Al diseñar este tipo de aplicaciones se debe considerar la afinación de los siguientes puntos:

- Índices, clusters y hashing
- Tamaño de bloque de datos
- La opción de parallel query
- El optimizador
- El uso de hints (9) en las consultas.
- Funciones de PL/SQL en las sentencias SQL.

Las aplicaciones **científicas o estadísticas** ejecutan frecuentemente cálculos complejos sobre una gran cantidad de datos. Con frecuencia estos tipos de datos representan tipos de datos complicados como la latitud o la presión atmosférica. Por ejemplo, una aplicación meteorológica puede constar de diversos cálculos y estadísticas para predecir los futuros patrones de clima. El objetivo clave de estas aplicaciones puede resumirse en velocidad y exactitud y muchas veces las arquitecturas de hardware son primordiales para este tipo de aplicativos. La afinación consiste en consumir adecuadamente los recursos del sistema para lo que necesitan considerar los siguientes puntos:

- Las funciones de PL/SQL en las sentencias SQL.
- La opción de parallel query
- El optimizador

- El uso de hints en las consultas.
- Aprovechar los ciclos de reloj de los procesadores.

4.1.9 Configuraciones de bases de datos.

Las configuraciones que **Oracle** ofrece pueden estar disponibles dependiendo del tipo de software y hardware con el que cuenta un sistema y su elección depende en gran medida del tipo de aplicaciones que estarán soportadas en ellas. A continuación se presentan las configuraciones básicas:

Base de datos distribuida: Esta configuración involucra datos distribuidos sobre múltiples servidores. Varias máquinas pequeñas pueden ser una solución más barata y flexible que una máquina grande. Esta opción permite que los datos estén localizados físicamente en varios sitios, y cada sitio puede acceder de manera transparente todos los datos. Con esta configuración se tienen las siguientes ventajas: Concurrencia, restauración, disponibilidad y precisión.

Cuando se opta por una base de datos distribuida, la localización de los datos es de vital importancia debiendo asegurarnos de que el acceso a los clientes sea rápido para los datos utilizados con mayor frecuencia disminuyendo así las operaciones remotas.

Oracle Parallel Server: Esta opción se encuentra disponible en sistemas en cluster o en sistemas de procesamiento masivo en paralelo que soportan discos compartidos. Un sistema en parallel server permite que varias máquinas convivan con su respectiva instancia las cuales comparten la misma base de datos. Cuando se configura un sistema de este tipo la clave principal es poder distribuir los recursos del sistema para los datos entre los diferentes nodos.

Procesamiento por Parallel Query: En esta configuración varios procesos trabajan simultáneamente para procesar una sola sentencia de SQL. Para lograr particionar una sentencia SQL es necesario contar con dos o más procesadores que puedan repartirse los procesos. El parallel query puede aumentar considerablemente el desempeño en aplicaciones de soporte de decisiones o en ambientes de VLDB.

Sistemas Híbridos: En aplicaciones de propósitos múltiples como en las de marketing en donde se ocupa OLTP y soporte de decisiones se utilizan configuraciones híbridas, buscando con ello una mejor solución de desempeño.

Hablando de **MySQL**, este maneja varios motores y tienen diferentes usos, por ejemplo:

MyISAM: Este es el motor por default que utiliza MySQL, utilizado generalmente por la mayoría de los sitios Web o de Data Warehouse. Es soportado por todas las versiones de MySQL.

InnoDB: Utilizado por su uso transaccional, puede manejar adecuadamente transacciones mediante commit y rollback y tiene capacidades para recuperación en caso de falla; mantiene consistencia de lectura similar a Oracle y soporte alta concurrencia y buen desempeño. Soporta integridad referencia.

Falcon: Diseñado para grandes volúmenes de información con gran desempeño y soporta fácilmente alta concurrencia y múltiples transacciones.

Maria: Diseñado con el mismo motor que el **MyISAM** sin embargo se le integra la funcionalidad de recuperación en caso de fallas.

Memory (HEAP): Esta diseñado para almacenar todos los datos el la RAM para tener un gran desempeño y acceso a la información muy rápidos.

Existen otras tantos motores para diferentes propósitos como son: **Merge**, **Archive**, **Federated**, **CSV**, **Blackhole** y **Examples**. Sin embargo las más comunes y de mayor uso han sido ya listadas con una breve explicación de su uso.

La tabla siguiente muestra una comparación de los motores principales basados en su uso y funcionalidad:

Característica	MyISAM	Memory	InnoDB	Archive	Falcon
Límite de almacenamiento	256TB	RAM	64TB	-	512 ZB
Transacciones	No	No	Si	No	Si
Nivel de candados	Tabla	Tabla	Registro	Registro	Registro
MVCC	No	No	Si	No	Si
Soporte a tipo de datos Geoespacial (geospatial)	Si	No	Si	Si	No
Soporte de indexado geoespacial (Geospatial)	Si	No	No	No	No
Índices B-tree	Si	Si	Si	No	Si
Índices tipo Hash	No	Si	No	No	No
Índices de búsqueda Full-text	Si	No	No	No	No
Índices en Cluster	No	No	Si	No	No
Caché de datos	No	-	Si	No	Si
Caché de índices	Si	-	Si	No	Si
Compresión de datos	Si**[a]**	No	Si**[b]**	Si	Si
Encriptación de datos**[c]**	Si	Si	Si	Si	Si
Soporte de Cluster de bases de datos	No	No	No	No	No
Soporte de replicación**[d]**	Si	Si	Si	Si	Si
Soporte de integridad referencial (llave foránea)	No	No	Si	No	No
Respaldo y recuperación en el tiempo**[e]**	Si	Si	Si	Si	Si
Soporte de Consultas a caché	Si	Si	Si	Si	Si
Actualización de estadístcias en el dicconario de datos	Si	Si	Si	Si	Si

[a] La compresión de datos en tablas MyISAM es soportado solo cuando se utiliza el formato de registro compactado (Compressed Row). Estas tablas son de solo lectura.

[b] La compresión de datos en las tablas de InnoDB solo es posible utilizando el plugin correspondiente.

[c], [d], [e] Implementado en el servidor a través de funciones de encripción en lugar de incluirlos en el motor de la base de datos.

En **SQL*Server** las configuraciones de bases de datos también vienen preestablecidas dependiendo del tipo de base de datos a utilizar, es decir, si se usará una base de datos altamente transaccional, si se requerirá una base de datos para minería de datos o para otros propósitos. Las configuraciones bases vienen previamente configuradas sin dar un margen más dinámico de configuraciones adicionales como lo hacen al día de hoy Oracle y MySQL.

4.1.10 El Optimizador en Oracle.

La optimización es el proceso de elegir la forma más eficiente para ejecutar una sentencia SQL, se considera como un paso muy importante en el procesamiento de las sentencias de DML (Data Manipulation Lenguage -select, insert, update o delete-). El optimizador considera un número de factores para considerar la mejor opción para ejecutar una sentencia. Sin embargo un diseñador puede elegir un método más apropiado por el cual puede ejecutarse una sentencia debido a que el conoce más a detalle los datos de la aplicación.

Para ejecutar una sentencia DML, Oracle ejecuta internamente una serie de pasos conocidos como plan de ejecución, el cuál puede ser examinado a través del comando Explain Plan. Este comando causa que el optimizador elija el plan de ejecución y entonces inserta los datos descritos por el plan dentro de una tabla de la base de datos.

El optimizador se basa en dos aproximaciones para elegir el plan de ejecución:

Basado en reglas: El optimizador elige el plan de ejecución basado en la ruta de acceso disponible según la prioridad mostrada en la siguiente tabla:

Prioridad:	Ruta de acceso:
1	ROWID
2	Single row por Cluster join
3	Llave de Hash cluster con índice único o llave primaria
4	Índice único o llave primaria
5	Cluster join
6	Llave de Hash cluster
7	Cluster key indexado
8	Llave compuesta
9	Columna única indexada
10	Búsqueda de rango limitado en una columna indexada
11	Búsqueda de rango no limitado en una columna indexada
12	Join de ordenamiento combinado
13	MAX o MIN de columna indexada
14	ORDER BY en una columna indexada
15	Full table scan

- **Basado en costos:** El optimizador considera las rutas de acceso disponibles y los factores basados en las estadísticas obtenidas de los objetos del diccionario de datos (tablas, clusters o índices) determinando así el plan de ejecución más eficiente. Las estadísticas se generan por medio del comando ANALYZE. Con el uso de estas estadísticas el optimizador estima la cantidad de procesos de Entrada/Salida, tiempo de CPU, y la cantidad de memoria que es requerida para ejecutar una sentencia SQL.

El optimizador de Oracle es una característica interesante que no contienen los otros motores de bases de datos relacionales y que permiten al DBA el ajustar los planes de ejecución de las sentencias de SQL basado en la experiencia del propio administrador y del desarrollador que esta construyendo los accesos a los datos.

Los otros motores como SQL*Server y MySQL tiene ya establecidos internamente el método de búsqueda de datos y no dejan al desarrollador ni al administrador elegir basado en su experiencia en mejor camino de acceso a la información.

4.1.11 La Opción de Parallel Query.

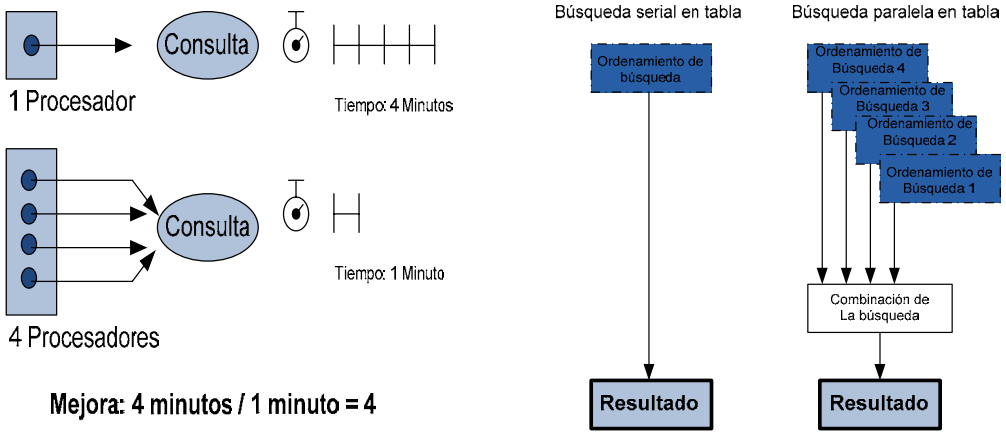
Sin la opción de parallel query el procesamiento de una sentencia de SQL es efectuado por un solo proceso de servidor. Utilizando Parallel Query, varios procesos trabajan en forma simultánea para efectuar una sentencia de SQL. Esta capacidad es conocida como procesamiento en Parallel Query. El manejador de base de datos puede procesar la sentencia más rápidamente que si un solo proceso de servidor la procesara.

El Parallel Query ayuda a los sistemas a escalar el desempeño cuando se agregan recursos de hardware. Si los CPU's y los controladores de disco están altamente cargados, se requiere disminuir la carga del sistema antes de utilizar la opción de Parallel Query. El Oracle Server utiliza la opción de Parallel Query en la mayoría de las siguientes sentencias:

- Sentencias Select
- Subqueries con sentencias Update, Insert y Delete
- Sentencias Create Table...As Select
- Sentencias de Create Index
- Arquitectura del Parallel Query y Paralelismo.

El Parallel Query permite que ciertas operaciones puedan ser ejecutadas en paralelo por varios procesos de servidor, El elemento para manejar los procesos múltiples es conocido como coordinador de consultas y es el encargado de dividir la ejecución de la sentencia en varios servidores de consultas y coordina el resultado desde todos los servidores y regresar el resultado al usuario.

El coordinador de consultas puede dividir las funciones de ejecución en piezas paralelas y entonces integrar el resultado parcial que producen los servidores de consulta. El número de servidores de consulta asignados a una sola operación es conocido como grado de paralelismo para una consulta de SQL, gráficamente mostrado en la siguiente figura:



LANDEROS, Darío. Tesina de Administración de las bases de datos relacionales y mejoramiento de su desempeño. México: UNAM FES Acatlán, 2010.

Afinación de la opción de Parallel Query.

Existen cuatro factores importantes a considerar en la afinación del Parallel Query:

1. **Identificar los sistemas apropiados para usar esta opción.** El Parallel Query puede ayudar a la mayoría de los sistemas siguientes: Sistemas de multiprocesamiento simétrico (SMP) o de procesamiento masivo en paralelo, sistemas con un alto porcentaje de procesos de Entrada/Salida (muchos datafiles distribuidos en varios discos), sistemas con un bajo porcentaje de uso de CPU (aproximado a 30%), y sistemas con memoria suficiente para asignar espacio a las sentencias de SQL. Si en determinado momento un sistema no cumpliera con alguna de estas características entonces se recomienda considerar el uso de Parallel Query debido a que una mala elección puede perjudicar el desempeño en vez de mejorarlo.

2. **La relación entre las Entradas/Salidas y Parallel Query.** El Parallel Query trabaja mejor en sistemas que tienen sus archivos distribuidos en varios discos, esto permite que los servidores de consulta aumenten el acceso concurrente a los datos. El tamaño de un archivo distribuido en varios discos debe ser un múltiplo mayor al tamaño de bloque de sistema operativo.

3. **El grado de paralelismo que debe utilizarse.** Es muy importante determinar el número de servidores de consulta que puede soportar el sistema a afinar y entonces dividir los servidores de consulta entre el número estimado de consultas concurrentes. El número máximo de servidores de consulta que un sistema puede soportar depende de la combinación de los siguientes factores: El número y capacidad de CPU's en el sistema, el número límite de procesos manejados por el sistema, el número de controladores de disco en donde los archivos que contienen las tablas están distribuidos, la cantidad de consultas en el sistema, la localización de los datos (memoria o disco), y el tipo de operaciones en una sentencia SQL.

4. **La cantidad de servidores de consultas que se van a utilizar.** Los diccionarios de datos de los RDBMS contienen tablas internas donde alojan estadísticas para determinar el número apropiado de servidores de consulta, las estadísticas que se utilizan para este caso son en especifico los tiempos que se mantiene ocupado un proceso en paralelo, los tiempos ociosos y la cantidad de procesos que se levantan para atender una sesión en paralelo.

Para poder dar mantenimiento a procesos en paralelo es importante consultar las tablas internas de manera periódica de tal forma que se puedan identificar posibles problemas de desempeño dentro del sistema.

Afinación de sentencias SQL.

Si estamos escribiendo sentencias SQL para una nueva aplicación, necesitamos tomar en cuenta los siguientes factores como una buena práctica inicial:

1. **Crear índices que puedan utilizarse en las sentencias.** Un índice puede ayudar a aumentar el desempeño de una consulta que selecciona una pequeña porción de renglones de una tabla, como guía general debe considerarse la utilización de un índice si el resultado de la consulta esta entre el 2% y el 4% del total de renglones de la tabla para considerar como eficiente su utilización. Las columnas que son buenas candidatas para formar un índice son aquellas que se utilizan

frecuentemente en la cláusula WHERE, aquellas utilizadas para realizar un JOIN con otra tabla o aquellas que tengan una buena selectividad (la selectividad de un índice es el porcentaje / cantidad de renglones que tienen el mismo valor para una columna indexada).

2. **Crear clusters para optimizar los JOIN's de una sentencia.** Las tablas candidatas a formar parte de un cluster son aquellas que son consultadas frecuentemente por las aplicaciones.

3. **Elegir un método de optimización según el motor adecuado de base de datos.** Para Oracle debemos elegir el optimizador basado en costos en aplicaciones nuevas, el optimizador basado en costos generalmente elige un plan de ejecución más conveniente que el optimizador basado en reglas. Para MySQL es importante considerar los motores y propósitos que necesitamos cubrir con mayor importancia. Si hablamos de SQL Server la mayor relevancia es el propósito de la base de datos ya que de eso depende las configuraciones previamente establecidas.

4. **Utilizar hints cuando sea conveniente para las sentencias.** Un diseñador de aplicación puede decidir el uso de algún hint que beneficiará en determinado momento la búsqueda de datos. Un hint es la forma de obligar al RDBMS a tomar una determinada forma de acceso como son: accesos de búsqueda completa en tabla, búsqueda por índices, uso estadísticas, uso de paralelismo y/o caché o accesos a memoria.

5. **Considerar sintaxis alternativas.** Debido a que SQL es un lenguaje flexible puede ser que dos sentencias produzcan el mismo resultado, pero sin embargo una de ellas será ejecutada de manera más óptima en el manejador de base de datos. Por medio de las herramientas gráficas de análisis de consultas disponibles se puede revisar el plan de ejecución y la forma en que se está obteniendo la información.

Afinación de sentencias SQL de una aplicación.

- La afinación de sentencias SQL existentes en una aplicación ya diseñada es algo más que escribir una nueva sentencia ya que el construir un nuevo índice por ejemplo puede afectar el acceso a otras sentencias ya construidas. Por ello es importante considerar que antes de desarrollar una nueva sentencia el desarrollador debe conocer a detalle el entorno que afecta a la aplicación. Para comenzar con la afinación de un SQL el desarrollador debe de estar familiarizado con la aplicación, sus sentencias y sus datos. En caso de no haber diseñado o desarrollado la aplicación es muy importante ponerse en contacto con aquellas personas que lo hicieron para contestar la información pertinente para afinar las sentencias, esta puede ser:

  - Que tipo de sentencias SQL utiliza la aplicación.
  - Que datos se procesan en la aplicación.
  - Cuales son las características y distribución de los datos.
  - Que operaciones ejecuta la aplicación sobre los datos.

Para obtener un parámetro que pudiera considerarse como buen desempeño, es necesario consultar con los usuarios el tiempo de respuesta de la aplicación preguntándoles

que parte de la misma consideran que aún no tiene buen desempeño, para entonces atacar a estas sentencias SQL aisladamente si fuera posible.

Las herramientas de diagnóstico apoyarán al desarrollador a medir el desempeño de las sentencias y de la aplicación misma, una facilidad que ofrecen las herramientas es la obtención de trazados de acceso a la información ofreciendo estadísticas de cada sentencia SQL procesada por el RDBMS dando un panorama con datos como:

- El número de veces que a cada sentencia SQL se le aplica la fase de parseo, de ejecución y finalmente la presentación de la información al usuario.
- El tiempo necesario para procesar cada sentencia.
- El uso de memoria y disco asociado a la sentencia.
- El número de renglones que procesa cada sentencia.

4.1.12 Afinación de la Instancia.

La afinación de la instancia comprende tres partes importantes que pueden evaluarse, estas son:

- Asignación de memoria.
- Entradas/Salidas.
- Contención.

Afinación de la asignación de memoria. Los RDBMS utilizan dos lugares para almacenar información: En memoria o en disco. El acceso a memoria es mucho más rápido que el acceso a disco por lo que es deseable que el mayor número de accesos se hagan a memoria. Sin embargo los recursos de memoria del sistema pueden ser limitados por lo que la afinación de asignación de memoria involucrará los recursos disponibles para las estructuras de memoria.

Los recursos que un manejador necesita dependen de la aplicación, por lo que se recomienda afinar las estructuras de memoria después de haber afinado la aplicación y las sentencias SQL. El hecho de afinar la asignación de memoria antes de afinar la aplicación y/o las sentencias SQL puede hacer que sea necesario reconsiderar el tamaño de algunas estructuras de memoria. Otro factor importante es el considerar afinar primero la asignación de memoria antes de afinar las Entradas/Salidas.

El proceso de afinar la asignación de memoria requiere de considerar cuatro estructuras principales:

1. **Afinación del sistema operativo.** Tiene tres objetivos básicos: Reducir la paginación, adecuar las estructuras de memoria del RDBMS dentro de la memoria principal del equipo, y asignar suficiente memoria para los usuarios. Estos objetivos aplican en general a la mayoría de los sistemas operativos, sin embargo, los detalles de afinación dependen estrictamente del tipo de sistema operativo que se este utilizando.

   Para reducir el área de **paginación** debemos considerar que las áreas de memoria real, memoria virtual, almacenamiento extendido y disco estén calculadas

adecuadamente. El acto de mover la información de una localización de almacenamiento a otra se le conoce como paginación. Debido a que el constante movimiento de la información en estructuras de almacenamiento requiere de cierto tiempo, el exceso de paginación puede reducir el desempeño de los sistemas operativos.

Cuando se monitorea la conducta del sistema operativo y encontramos que hay exceso de paginación, entonces debemos aumentar el total de memoria del sistema o disminuir la cantidad de memoria que se tiene asignada a ciertos procesos no críticos.

El segundo aspecto importante es la afinación de las estructuras de memoria del manejador de base de datos cuyo propósito es almacenar la mayor cantidad de datos para un acceso más rápido, siempre debe estar comprendido dentro de la memoria principal del equipo. Si el propio RDBMS provoca paginación a disco, los datos dentro de ella no son accesibles rápidamente.

El otro punto a considerar es la asignación de memoria a cada usuario. En algunos sistemas operativos podemos tener el control sobre la cantidad de memoria física asignada a cada usuario; dependiendo del sistema operativo los recursos que pueden asignarse son: Acceso al ejecutable de los RDBMS, las estructuras de memoria, las herramientas de aplicación y el acceso a determinados datos de la aplicación.

Según el sistema operativo, el software de cada manejador puede instalarse con una imagen del ejecutable de RDBMS que se comparten por muchos usuarios, logrando que se reduzca la cantidad de memoria requerida por usuario.

4.1.13 Afinación de Lecturas/Escrituras en Disco.

La afinación de Entradas/Salidas puede ayudar al desempeño si un disco que contiene archivos de la base de datos está operando a toda su capacidad. Sin embargo, no puede mejorarse si el CPU opera a su máxima capacidad. Para poder afinar este punto es necesario haber afinado previamente la asignación de memoria, y las sentencias de SQL.

Un factor muy importante a considerar es la disminución de la contención en disco, esto es cuando varios procesos tratan de ingresar el disco en forma simultánea. La mayoría de los discos tienen un límite en cuanto al número de accesos y la cantidad de datos que pueden transferir por segundo, cuando estos límites son alcanzados, los procesos restantes deben esperar a que algún recurso sea liberado para poder utilizarlo.

La actividad de un disco se refleja en las estadísticas de procesos de entrada/salida de archivos de cada manejador y en las estadísticas de sistema operativo. La recolección de estadísticas de sistema operativo es específica de la plataforma y de la versión de sistema operativo del cual se trate por lo que es necesario mantenerse informado y actualizado con respecto al sistema operativo que ocupamos para poder determinar la asignación de recursos y los límites con los que cuenta el sistema.

Las estadísticas más importantes cuando hablamos de problemas de contención en disco son:

Lecturas Físicas. Este valor es el número de lecturas a cada archivo de la base de datos que se encuentra en disco, cabe resaltar que los accesos cuentan a nivel tabla y a nivel índice.

Escrituras Físicas. Refleja el número de escrituras para cada archivo de la base de datos en disco y cuenta de la misma forma tanto para tablas como para índices.

Para poder determinar un buen perfil de acceso a los diferentes discos se recomienda consultar periódicamente las estadísticas antes de iniciar la afinación. El total de entradas/salidas a disco está determinado por la suma de Lecturas Físicas y Escrituras Físicas para todos los archivos de la base de datos manejados por el RDBMS en cada disco que compone el sistema de almacenamiento.

Una vez monitoreado periódicamente el acceso a disco pueden determinarse los valores para cada disco así como también el rango de ocurrencia de entradas/salidas por cada disco dividiendo el total de entradas/salidas por el intervalo de tiempo sobre el cual fueron reunidas las estadísticas.

Después de haber analizado las estadísticas y las ventajas que ofrece la tecnología de almacenamiento que utiliza el sistema podemos iniciar a afinar los problemas de contención siguiendo los siguientes patrones:

- Separar los archivos de datos y los de bitácoras o registros en discos diferentes
- Distribuir los datos de las tablas de mayor uso en discos diferentes
- Separar tablas e índices en varios discos
- Reducir las entradas/salidas no relacionadas al RDBMS, es decir, manejar aplicaciones en otros discos así como archivos de sistema operativo evitando que estos compitan unos con otros en los accesos a disco.

4.1.14 Afinación de Espacio de Bloques de Datos.

Una forma de afinar la asignación de espacio en los bloques de datos es controlando el almacenamiento de los datos y ver la forma en que son almacenados lo más eficientemente posible basándonos en el conocimiento de la aplicación.

Los problemas principales que podemos enfrentar en este caso son la migración y encadenamiento de renglones. Cuando una sentencia de UPDATE aumenta la cantidad de datos en un renglón de forma tal que el renglón completo no cabe en su respectivo bloque de datos, entonces éste se mueve hacia un nuevo bloque provocando con ello una migración de renglones o registros. Si el renglón es demasiado grande para almacenarse en un bloque de datos disponible entonces los manejadores separan el renglón en varias piezas y almacena cada una de ellas en bloques separados ocasionando con ello un encadenamiento de renglón. Los renglones también pueden encadenarse a través de una sentencia INSERT.

El manejo de espacio dinámico como la migración y el encadenamiento disminuyen el desempeño de la base de datos. Las actualizaciones que ocasionan migración y encadenamiento se ejecutan más pobremente al igual que las consultas que seleccionan los renglones migrados o encadenados requieren de más procesos de entrada/salida.

Para reducir la migración o encadenamiento de registros es importante aprovechar los beneficios y facilidades ofrecidas por cada manejador, así mismo es importante dar mantenimiento frecuente a las tablas e índices con mayor crecimiento.

Es recomendable también elaborar rutinas de verificación de crecimientos en cada segmento de las bases de datos haciendo labores de administración que permitan mantener el control de cómo irán creciendo los objetos sin dejarlos disponer del espacio de manera libre. Ciertos manejadores como Oracle y MySQL tienen herramientas que permiten

modificar el crecimiento de los bloques, hacer una nueva distribución de espacios en los objetos, compactar espacios vacíos y moviendo información a espacios contiguos. SQL Server utiliza frecuentemente las utilerías que el mismo windows ofrece.

A nivel global, derivado del tema que se ha hablado en este capítulo, se presenta la siguiente tabla donde vemos una comparación de cada manejador que he utilizado en este trabajo.

Atributo	Oracle	MySQL	SQL*Server
Actualización de estadísticas	Si	Si	Si
Analyze de tablas	Si	Si	Si
Índices B-tree	Si	Si	Si
Índices Bitmap	Si	No	No
Índices Hash	Si	No	No
Compactación de espacios	Si	Si	Si
Diferentes puntos de montaje para archivos	Si	Si	No
Hints	Si	Si	Si
Manejo de caché	Si	Si	Si

Nuevamente se observa que los manejadores actuales nos ofrecen características suficientes para realizar una adecuada administración y afinación tanto de bases de datos, aplicaciones y sentencias SQL.

Conclusiones.

Para concluir he de señalar que el trabajo de un administrador de base de datos es crucial para tener disponible la información en el momento en que sea requerida, cuya importancia radica en ser el activo de más valor para una empresa.

Durante este trabajo se han mencionado las actividades primordiales y esenciales que un DBA debe procurar en sus actividades diarias pero basta con decir que la información no es nada si no es utilizada para los propósitos de generar utilidad en una empresa. Es por ello que basta solo decir que el no tener la información a tiempo implica grandes pérdidas para un negocio. Entonces a todo esto, ¿Como poder llegar a ser competitivo en un mundo globalizado?, pues simplemente si partimos de que la información es poder y quién la posee y sabe utilizarla es poderoso en buen sentido de la palabra siempre y cuando se use para buenos fines, el poder llegar a un esquema de compartir esa información hace invencible a quien se puede dar el lujo de proporcionarla.

Claro está que la administración entra en un contexto muy amplio del quehacer diario de una empresa, pero, ¿Que haría un directivo que depende de la información fresca y en el momento para tomar una decisión de millones de dólares?, y ¿Que pasaría si además de esto la información no fuera confiable?

Es evidente que quién no tiene información no puede hacer una buena toma de decisiones, entonces es ahí en donde justamente un DBA juega un papel importante, desde el momento en que está administrando la información que es generada en un negocio sea del tamaño que sea y que además de eso debe tener esquemas de respaldo para que la información pueda ser recuperada en casos de alguna falla o en caso de alguna catástrofe, y en adición a estas dos actividades, esta otra no menos importante que es la de procurar tener la información con métodos de acceso que aseguren que las aplicaciones que hace uso de ella sea ágiles y de un tiempo de respuesta adecuado para quién necesite de esa información.

Hoy en día se manejan tipos de datos muy complejos y que a su vez requieren de arquitecturas de alta disponibilidad, de aplicaciones que puedan dar un acceso seguro casi desde cualquier parte del mundo y de sistemas de bases de datos que aseguren una integridad de la información, así como de sistemas tolerantes a fallas y de misión crítica que puedan ofrecer la información las 24 horas de un día toda la semana durante todo el año. No es una tarea fácil ya que de ello depende también que las labores de mantenimientos preventivos deban planearse y asistir con habilidad y destreza en caso de mantenimientos correctivos.

Al final del día el hardware puede fallar ya que no tiene palabra de honor pero la real situación es que la pérdida de información es un punto que un administrador de base de datos no puede darse el lujo de tener.

En situaciones reales hay decisiones que se basan en la importancia de recuperar el servicio lo más rápido posible, poniendo en tema de juicio incluso la pérdida de información donde es imperante saber las necesidades del negocio. Si de ello depende que un sistema crítico deba salir a producción sin importar por el momento la información entonces vale la pena correr el riesgo de poner disponible los servicios en producción y después preocuparse por ver como restaurar esa información en una segunda fase.

En la realidad el que alguien esté en una situación crítica de restauración de información implica un gran control en ese momento que es de alto contenido de estrés y de nerviosismo; sin embargo hasta en el temple y carácter se identifican los buenos DBA's, además de quienes saben tomar las decisiones adecuadas en problemas en donde el riesgo de pérdida de información es un peligro inminente.

El tamaño de las corporaciones ha obligado a que existan grupos de administradores de bases de datos especializados en cada área en específico como por ejemplo:

- Control de accesos y seguridad.
- Afinación de desempeño.
- Respaldo y recuperación.

¿Quien puede decir y determinar cual de estas tres responsabilidades tiene más prioridad? Cada una en su momento es crucial en el manejo de la información, en caso de ser vulnerables a intrusiones, la información tiene un riesgo latente de pasar a manos que no se desearía que la tuvieran, tal vez en manos de la competencia en donde estaría más que dispuesta a pagar cantidades exorbitantes por saber los planes de lanzamiento de algún producto innovador en el mercado; o por ejemplo si la información se tiene pero no esta disponible en medio de una operación financiera de compra-venta de alguna nueva empresa; y ni que decir si como ya se comentó la información llegara a perderse por una mala planeación de respaldos de la misma.

Sin lugar a dudas y sin temor a equivocarme yo me atrevería a aseverar que un buen sistema de respaldo y recuperación de información es un seguro de vida que bien vale la pena tener disponible en el momento que se solicite ya que los otros dos puntos bien o mal tienen alguna solución de carácter urgente pero manejable.

Los sistemas de hoy en día tienen cada vez una mayor complejidad y los datos se van haciendo más robustos por lo cual debemos considerar que estos sean escalables mientras tanto estos vayan creciendo en concurrencia, volumen o transacciones.

Considerar esquemas de replicación puede ser un gran acierto ya que nadie puede saber cuando se tendrán contingencias que ameriten el uso de esquemas que beneficien la continuidad del negocio. Hablar de temas como diseños de estrategias de continuidad de negocio y/o planes de recuperación en caso de desastres, es algo que todo mundo en un área de sistemas tiene en mente; sin embargo, son tareas nada fáciles de llevar a cabo debido a los grandes costos que puede acarrear el tener sitios de computo dedicados en exclusiva a entrar en operación cuando suceda una contingencia de grandes magnitudes, pero ¿No vale la pena invertir en planes que garanticen que una empresa pueda mantener su operación mínima necesaria?.

Estos temas mencionados en el párrafo anterior dan mucho de que hablar y requieren de un amplio conocimiento de gente experta para realmente establecer un plan y llevarlo a la puesta en marcha.

Este trabajo tiene la intención de destacar labores importantes en la administración de base de datos; sin embargo, solo es una pequeña porción y contribución a un espectro mucho mayor de opciones que cada vez van siendo enriquecidas por los avances tecnológicos en el manejo de datos que cada fabricante utiliza para hacer cada vez más robusto y confiable el manejo de información.

Glosario.

(1) Edgar F. Codd. En las décadas de los sesenta y los setenta trabajó en sus teorías sobre modelado de datos, publicando su trabajo "Un modelo relacional de datos para grandes bancos de datos compartidos" ("A Relational Model of Data for Large Shared Data Banks") en 1970. Para su descontento, IBM no se apresuró a explotar sus sugerencias hasta que no empezaron a ser puestas en práctica por rivales comerciales. También es el autor del término de OLDAP y redacto las 12 leyes del procesamiento analítico informático.

(2) Metadatos. Una descripción simple del concepto podría ser que son datos que representan otros datos, en el caso del diccionario de datos y los catálogos son precisamente datos que se utilizan para controlar más datos.

(3) Constraint. Es una restricción que se utilizan para evitar ambigüedades en la definición de datos.

(4) Tupla. Secuencia ordenada de datos, las tuplas se utilizan para describir objetos matemáticos que tienen estructura y que pueden ser descompuestos en cierto número de componentes.

(5) Multiplexada. Se refiere a colocar de manera combinada los archivos de redo log de las bases de datos.

(6) Espejo (Mirroring). Se refiere a tener dos discos conectados, donde uno es el primario y el otro es el secundario que mantiene una copia idéntica del primario.

(7) Sincronización. Es el proceso interno que mantienen los discos a través de sus controladores y buffers de lectura y escritura para mantener la consistencia de la información, en muchas situaciones puede perderse la sincronización por problemas de estos buffers.

(8) Latches. El término hace referencia a candados internos de Oracle, principalmente por procesos de memoria.

(9) Hints. Se definen como las pistas que se le dan a un manejador de base de datos para que elija un plan de ejecución adecuado. Los hints se incorporan en sentencias DML.

Bibliografía.

1. **Oracle 9i PL/SQL programming.**
   Scott Urman
   Editorial Mc Graw Hill / Osborne
   2002

2. **OCA Oracle 9i Associate DBA Certification Exam Guide.**
   Jason Couchman
   Editorial Mc Graw Hill / Osborne
   2002

3. **Oracle 7, Oracle 8, Oracle 8i and Oracle 9i Documentation CD.**
   Oracle Technical Guides and Generic Documentation.
   Editorial Oracle Corporation
   2000, 2001, 2002, 2003, 2004, 2005

4. **Microsoft SQL Server 2000 System Administration (MCSE).**
   Training Kit
   Editorial Microsoft press
   2001

5. **Norma Internacional ISO 690-2**
   Organización Internacional de Normalización
   http://www.iso.org/iso/home.htm
   Primera Edición 1997

6. **Metalink Web Site.**
   http://metalink.oracle.com
   Oracle Customer support Web Site.

7. **Software Engineering Group Web Site.**
   http://www.swen.uwaterloo.ca
   University of Waterloo.

8. **Oracle Corporation Web Site.**
   http://www.oracle.com
   Oracle Homepage.

9. **Oracle Frequently Asked Questions Web Site.**
   http://www.orafaq.com

10. **MySQL Developers Website**
    http://dev.mysql.com/

11. **MySQL Website**
    http://www.mysql.com/

12. **GNU Project Website**

http://www.gnu.org/

13. **Opensource Initiative Website**
    http://www.opensource.org/

14. **Microsoft Corporation Website**
    http://technet.microsoft.com

15. **Techrepublic Website**
    http://techrepublic.com.com/

16. **Wikipedia Website**
    http://wikipedia.com/

17. **Technology Website**
    http://www.lifeaftercoffee.com