UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

“DESARROLLO DE POLÍTICAS E IMPLEMENTACIÓN DE LA ADMINISTRACIÓN Y MONITOREO DE BASES DE DATOS ORACLE”

T R A B A J O E S C R I T O

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE
CAPACITACIÓN Y ACTUALIZACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:

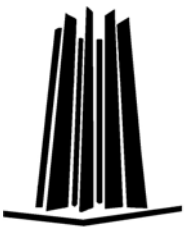
INGENIERO EN COMPUTACIÒN

P R E S E N T A :

IRVING COSME HERNÁNDEZ

ASESOR:

ING. JOSÉ AGUSTÍN VARGAS PADILLA



SAN JUAN DE ARAGÓN, ESTADO DE MÉXICO AGOSTO DEL 2006

Dedicatoria.

A Dios por haberme dejado llegar hasta donde he llegado y por todas las bendiciones que he recibido de él, junto con mi familia.

A mi mamá: Aurora Hernández Morales quien me ha apoyado toda mi vida para lograr mis objetivos; por su infinito amor, cariño, sacrificios, comprensión y ayuda para lograr esta meta y por todo lo que ha hecho por mí, para que yo siga adelante con mis todos mis planes.

A mi hermano: Ivan Eduardo Cosme Hernández por todo su apoyo incondicional que me ha brindando y por ser el ejemplo a seguir en todos los aspectos.

A mi novia: Lizbeth Leija Hernández por ser mi compañera desde el inicio de esta carrera y por estar siempre junto a mi.

A toda mi familia por ser la inspiración para lograr este objetivo y por ser el motivo de seguir adelante.

A mis amigos por el apoyo y confianza que tuvieron en mí.

Agradecimientos.

Agradezco a la Maestra Alma Ibarra Obando por la oportunidad que me dio de cursar el diplomado con el cual fue posible realizar este trabajo y por todo su apoyo que me brindo para iniciarme en el desarrollo profesional.

Agradezco a mi asesor de tesis Ingeniero José Agustín Vargas Padilla por sus consejos y apoyo durante la licenciatura y formación profesional.

Agradezco a todos los profesores de la Facultad de Estudios Profesionales Aragón de la carrera Ingeniería en Computación por su contribución en mi formación profesional.

Agradezco a la Universidad Nacional Autónoma de México por darme la oportunidad de pertenecer a la mejor institución de México y por haber hecho de mí un profesionista.

ÍNDICE

INTRODUCCIÓN

El presente trabajo describe los conceptos más importantes que fueron vistos en el diplomado “Administración de Bases de Datos”, así como ejemplos y aplicaciones útiles para esta tarea en particular.

Los tres capítulos con que cuenta, describen los temas sobre la administración de bases de datos, desde conceptos básicos hasta el uso herramientas de monitoreo, propias del manejador de base de datos Oracle.

En el primer capítulo se mencionan las definiciones primordiales que debe conocer un administrador de base de datos, así como el manejo del lenguaje estructurado de consultas, tareas del administrador y la aplicación de políticas de administración.

El segundo capítulo presenta la arquitectura del sistema manejador de base de datos Oracle y sus características.

El tercer capitulo muestra el uso de las herramientas de administración y monitoreo del manejador de bases de datos Oracle, en particular la herramienta Oracle Enterprise Manager, el cual cuenta con paquetes de monitoreo de usuarios y datos. Con este tipo de herramientas se puede aplicar la administración de las bases de datos de manera eficiente como la creación de usuarios, la gestión de espacio y el monitoreo en general de todo el sistema.

JUSTIFICACIÓN DEL PROYECTO

¿Por qué de esta investigación?

Debido a que las bases de datos son ampliamente utilizadas en las empresas, siendo en muchos casos el corazón o el núcleo de los sistemas de información y el eje de las actividades administrativas de cualquier empresa.

Mantener funcionando adecuadamente una base de datos con cientos o miles de usuarios no es una labor fácil, y en ocasiones el fallo en estos sistemas provoca que se interrumpan actividades primordiales. Es por eso que la administración y gestión de estos sistemas constituyen, un trabajo muy importante dentro de cualquier organización.

Cualquier Base de Datos actual requiere un mantenimiento, basado en una buena *Administración y Gestión* de los datos que contiene, de los procesos asociados y de los usuarios que pueden acceder.

a) Para lograr una buena administración de bases de datos se tienen que plantear las reglas o políticas de administración, esto se define para lograr un buen control del sistema y seguridad del mismo.

b) Para lograr una buena gestión de bases de datos, los administradores cuentan con herramientas que permiten monitorear la eficiencia y recursos del sistema operativo.

c) La tecnología Oracle Database es una de las bases de datos más robusta, rápida y fiable, garantiza la mayor escalabilidad y fiabilidad.

Son estas razones por las que realizo este informe sobre este tema.

OBJETIVOS

Objetivo General.

- Comprender las funciones de la Administración de Bases de Datos, así como los métodos y técnicas para asegurar una correcta administración.

Objetivo particular:

- Conocer los métodos, técnicas, funciones y normas de la administración de bases de datos.

Objetivo específicos.

- Dar a conocer el papel y las funciones de la Administración de Bases de Datos en el entorno de los Sistemas de Información en general.

- Explicar las funciones de la Administración de las bases de Datos en cada una de las etapas del ciclo de vida de una Base de Datos.

- Estudiar las herramientas con que dispone el Administrador de Bases de Datos para cumplir su función.

- Describir la Administración de Bases de Datos, referido a cuestiones como políticas, confidencialidad, seguridad e integridad, profundizando en los mecanismos y técnicas existentes para abordar con éxito estos puntos, tales como control de usuarios, control de accesos concurrentes y transacciones.

ALCANCES Y LIMITACIONES

Para lograr los objetivos planteados debemos contar con:

- La definición de reglas o políticas, ya que son una herramienta base para el administrador, provee el control de gestión y seguridad de la información.
- Las herramientas de administración permitirán una mejor visión del estado de la base de datos así como el performance de la misma.
- El monitoreo permite al administrador verificar los recursos de hardware.
- La tecnología Oracle, pueden mejorar la disponibilidad de las aplicaciones y la escalabilidad de los sistemas.

CAPÍTULO I.
INFORME GENERAL DEL DIPLOMADO
"ADMINISTRACIÓN DE BASES DE DATOS"

1.1 Conceptos Básicos

Dato: Es un conjunto de caracteres con algún significado, pueden ser alfabéticos, numéricos, o alfanuméricos.

Información: Es un conjunto ordenado de datos los cuales son manejados según la necesidad del usuario, para que un conjunto de datos pueda ser procesado eficientemente y pueda dar lugar a información, primero se debe guardar lógicamente en archivos.

Análisis de la información: Es el modelo de datos que consiste en la representación conceptual, y esta debe ser clara.

Una **Base de datos** es un conjunto de datos estructurados, es decir una colección de tablas interrelacionadas entre sí, de forma que pueden accederse a ellos automáticamente e independientemente de los programas que gestionan esos datos.

Toda base de datos está formada por uno o varios bloques de información llamados **TABLAS** que normalmente tendrán alguna característica en común.

Una Tabla o archivo de datos es un conjunto de información del mismo tipo. Cada tabla esta formada por **REGISTROS**. Un registro es la unidad elemental de la información de la tabla, cada registro esta formado por uno o más elementos llamados **CAMPOS**. Un campo es cada una de las informaciones que interesa almacenar en cada registro.

Los tres componentes principales de un sistema de base de datos son el hardware, el software **DBMS** y los datos a manejar, así como el personal encargado del manejo del sistema.

El **Sistema Manejador de Bases de Datos** (DBMS) organiza y estructura los datos de tal modo que puedan ser recuperados y manipulados por usuarios y programas de aplicación, además se encarga de manejar la creación y todos los accesos a las bases de datos. Se compone de un conjunto de programas, procedimientos y lenguajes que proporcionan a los usuarios las herramientas necesarias para operar con una base de datos. Por lo tanto, el DBMS actúa como un intermediario entre los usuarios y los datos. Debe cumplir una serie de funciones como descripción de los datos, de manera que debe permitir definir los registros, sus campos, sus relaciones de autorización, etc.

1. Cesar Pérez, "Administración y Análisis de Bases de Datos" (México, Alfaomega-Rama, 2002) 15 – 32 pp.

El DBMS debe manipular los datos permitiendo a los usuarios insertar, suprimir, modificar y consultar datos de la base de datos y por último, debe permitir usar la base de datos, dando un interfaz adecuado a cada tipo de usuario.

Características de un DBMS Manejador de Bases de Datos

Éste esta formado por una:

- Base de Datos
- Software para manipular los datos

Funciones:

- Almacena, recupera, elimina y modifica los datos
- Guarda la consistencia de los datos
- Soluciona problemas de concurrencia
- Tiene seguridad
- Creación y modificación de la base de datos

Objetivo:

- Crear un ambiente en que sea posible guardar y recuperar información en la base de datos en forma eficiente.

El manejo de datos de DBMS incluye tanto la definición como la manipulación y la seguridad de los mismos:

DDL Lenguaje de Definición de Datos
DML Lenguaje de Manipulación de Datos
DCL Lenguaje de Control de Datos

Modelos de datos

Un modelo es una representación de la realidad que contiene las características generales de algo que se va a realizar.

Un modelo de datos es un conjunto de herramientas conceptuales para describir los datos, las relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia.

1. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 15 – 32 pp.

Los modelos de datos se clasifican en tres grupos principales:

1. Modelos lógicos basados en objetos. Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. **Ejemplos:**

- Modelo entidad relación. Este modelo representa a la realidad a través de entidades**,** que son objetos que existen y que se distinguen de otros por sus características.
-
- Modelo semántico de los datos (El orientado a objetos). Se basa en objetos, los cuales contienen valores y métodos, entendidos como órdenes que actúan sobre los valores, en niveles de anidamiento.

2. Modelos lógicos basados en registros. Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los tres modelos de datos más ampliamente aceptados son:

- Modelo relacional: Los datos y las relaciones se representan mediante tablas, cada una con diferentes columnas y nombres únicos.
- Modelo de red: Los datos se representan mediante nombres de registros y las relaciones mediante conjunto de ligas.
- Modelo jerárquico: Es semejante al modelo de red, pero con una estructura de árbol.

3. Modelos físicos de datos.- Los modelos físicos describen cómo se almacenan los datos en la computadora: el formato de los registros, la estructura de los archivos y los métodos de acceso utilizados.

. Ejemplos de este tipo de modelos son:

- Modelo unificador
- Modelo memoria de cuadros.

1. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 15 – 32 pp.

1.2 Modelo de datos relacional

El modelo relacional, es el modelo lógico en el que se basan la mayoría de los SGBD comerciales en uso hoy en día.

En 1970, el modo en que se veían las bases de datos cambió por completo cuando E. F. Codd introdujo el modelo relacional. En aquellos momentos, el enfoque existente para la estructura de las bases de datos utilizaba punteros físicos (direcciones de disco) para relacionar registros de distintos archivos. Codd demostró que estas bases de datos limitaban en gran medida los tipos de operaciones que los usuarios podían realizar sobre los datos. Además, estas bases de datos eran muy vulnerables a cambios en el entorno físico. Si se añadían los controladores de un nuevo disco al sistema y los datos se movían de una localización física a otra, se requería una conversión de los archivos de datos. Estos sistemas se basaban en el modelo de red y el modelo jerárquico, los dos modelos lógicos que constituyeron la primera generación de los BDMS.

Dada la popularidad del modelo relacional, muchos sistemas se han modificado para proporcionar una interfaz de usuario relacional, con independencia del modelo lógico que soportan (de red o jerárquico).

El modelo relacional, como todo modelo de datos, tiene que ver con tres aspectos de los datos:

- Estructura de datos.
- Integridad de datos.
- Manejo de datos.

1.2.1 Estructura de datos relacional

La relación es el elemento fundamental del modelo relacional y se puede representar en forma de tabla en las cuales los renglones (tuplas) equivalen a cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la tupla como se muestra en el ejemplo de la figura 1.

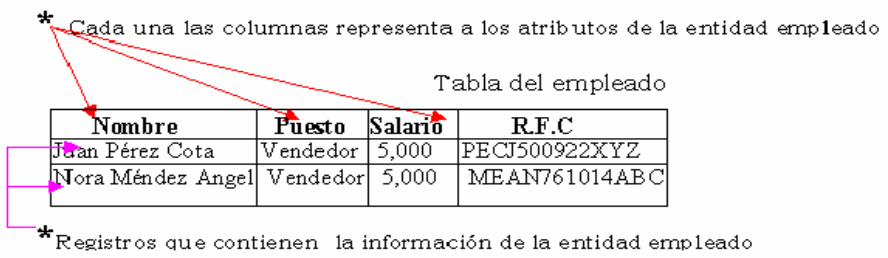


Figura 1 Tabla empleado

2. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 39-51 pp

La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las filas y las tablas son visibles en los datos. Este enfoque permite a los usuarios obtener información de la base de datos sin asistencia de sistemas profesionales de administración de información.

Un dominio es el conjunto de valores legales de uno o varios atributos. Los dominios constituyen una poderosa característica del modelo relacional. Cada atributo de una base de datos relacional se define sobre un dominio, pudiendo haber varios atributos definidos sobre el mismo dominio.

El concepto de dominio es importante porque permite que el usuario defina, en un lugar común, el significado y la fuente de los valores que los atributos pueden tomar. Esto hace que haya más información disponible para el sistema cuando éste va a ejecutar una operación relacional, de modo que las operaciones que son semánticamente incorrectas, se pueden evitar.

1.2.2 Propiedades de las relaciones

Las relaciones tienen las siguientes características:

- Cada relación tiene un nombre y éste es distinto del nombre de todas las demás.
- Los valores de los atributos son atómicos: en cada tupla, cada atributo toma un solo valor. Se dice que las relaciones están normalizadas.
- No hay dos atributos que se llamen igual.
- El orden de los atributos no importa: los atributos no están ordenados.
- Cada tupla es distinta de las demás: no hay tuplas duplicadas.
- El orden de las tuplas no importa: las tuplas no están ordenadas.
- La información en las bases de datos son representados como datos explícitos, no existen apuntadores o ligas entre las tablas.

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos.

Una superclave es un atributo o un conjunto de atributos que identifican de modo único las tuplas de una relación.

Una clave candidata es una superclave en la que ninguno de sus subconjuntos es una superclave de la relación. El atributo o conjunto de atributos de la relación es una clave candidata para si y sólo si satisface las siguientes propiedades:

2. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 39-51 pp

- Unicidad: no pueden existir dos tuplas con el mismo valor en todos los atributos que forman la llave candidata.
- Minimalidad: no existe ningún subconjunto de la llave que cumpla la regla de unicidad.

Cuando una clave candidata está formada por más de un atributo, se dice que es una clave compuesta. Una relación puede tener varias claves candidatas.

La clave primaria de una relación es aquella clave candidata que se escoge para identificar sus tuplas de modo único. Ya que una relación no tiene tuplas duplicadas, siempre hay una clave candidata y, por lo tanto, la relación siempre tiene clave primaria.

Las claves candidatas que no son escogidas como clave primaria son denominadas claves alternativas.

Una clave ajena es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (puede ser la misma). Las claves ajenas representan relaciones entre datos.

1.2.3 Integridad de datos

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina restricciones de dominios. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias.

Estas reglas son la regla de integridad de entidades y la regla de integridad referencial.

Las **reglas de integridad de entidades** establecen que ningún valor de clave primaria puede ser nulo ya que el valor de la clave primaria se usa para identificar las tuplas individuales de una relación; permitir valores nulos para la clave primaria implica que no se pueda identificar algunas tuplas. Por ejemplo, si dos o más tuplas tuvieran un valor nulo como clave primaria, no podríamos distinguirlas.

La **reglas de integridad referencial**, en cambio, se especifica entre dos relaciones, y se usa para mantener la congruencia entre las tuplas de las dos relaciones. Informalmente la restricción de integridad referencial establece que una tupla de una relación que haga referencia a otra relación debe referirse a una tupla existente en esa relación. Las condiciones de una clave ajena, especifican una restricción de integridad referencial entre los dos esquemas de relación R1 y R2.

2. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 39-51 pp

1.2.4 Manejo de datos

El sublenguaje de datos es un lenguaje de manejo de datos para el sistema relacional, el álgebra relacional y cálculo relacional, ambos lenguajes son "relacionalmente completos", esto es, cualquier relación que pueda derivarse de una o más tablas, también se puede derivar con un solo comando del sublenguaje. Por tanto, el modo de operación de entrada/Salida en un sistema relacional se puede procesar en la forma: una tabla a la vez en lugar de: un registro a la vez; en otras palabras, se puede recuperar una tabla en vez de un solo registro con la ejecución de un comando del sublenguaje de datos.

2. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 39-51 pp

1.3 Álgebra relacional.

El álgebra relacional es una colección de operaciones sobre relaciones donde cada operación toma una o más relaciones como sus operandos y produce otra relación como su resultado. Dado que el resultado de una operación del álgebra relacional es una relación, ésta a su vez puede ser sujeto de posteriores operaciones algebraicas.

En el álgebra relacional se consideran dos tipos de operadores:

- Los operadores **tradicionales** sobre conjuntos: unión, intersección, diferencia y producto cartesiano.

- Los operadores **especiales**: proyección, selección, join.

1.3.1 Operadores de Conjuntos.

Las bases de datos relacionales están basadas en el concepto matemático de relaciones entre conjuntos. Así las operaciones que se pueden efectuar entre relaciones son tanto las comunes a los conjuntos, unión, intersección, diferencia, producto cartesiano; como las específicas de las relaciones, selección, proyección, etc.

Si q, r y s son relaciones con todos los dominios iguales, esto es, con el mismo esquema, se les puede aplicar las operaciones típicas de conjuntos.

1.3.1.1 Unión (r ∪ s)

Es la relación sobre los mismos dominios que contiene las eneadas (agrupaciones) que están en r, en s o en ambas. Esto es, son todos los elementos que se encuentran en el conjunto s y en el r. Construye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones especificadas o la suma de sus elementos.

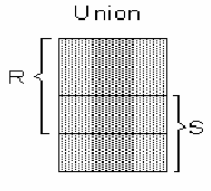


Figura 2. Union

3. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 72 – 84 pp

Ejemplo: Sean r y s relaciones con esquema {A, B, C}

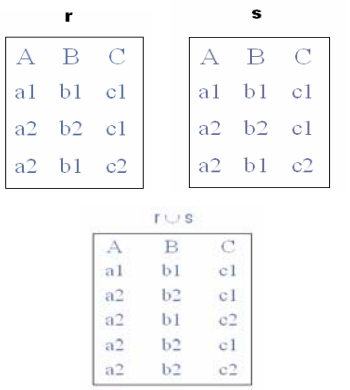


Figura 3. Ejemplo de union

1.3.1.2 Intersección (r/s ó r ∩ s)

Es la relación que contiene las eneadas que están en r y en s. Esto es, construye una relación formada por aquellas tuplas que aparezcan en las dos relaciones especificadas.

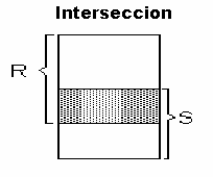


Figura 4. Intersección

Ejemplo: Sean r y s relaciones con esquema {A, B, C}

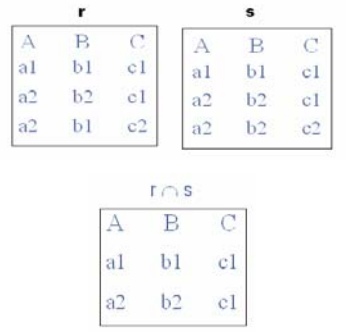


Figura 5. Ejemplo de interseccion

1.3.1.3 Diferencia (r – s)

Es la relación con las eneadas que están en r pero no en s. Corresponde a obtener los elementos del conjunto r que no se encuentran en el conjunto s.

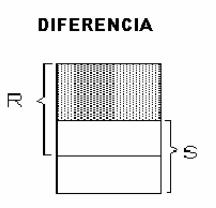


Figura 6. Diferencia.

Ejemplo: Sean r y s relaciones con esquema {A, B, C}

r

A	B	C
a1	b1	c1
a2	b2	c1
a2	b1	c2

s

A	B	C
a1	b1	c1
a2	b2	c1
a2	b2	c2

r - s

A	B	C
a2	b1	c2

s - r

A	B	C
a2	b2	c2

Figura 7. Ejemplo de Diferencia.

1.3.1.4 Producto cartesiano (r x s)

Obtiene todas las eneadas que se construyen concatenando cada eneada de r con otra de s. En este caso los dominios de r y s no tienen que ser los mismos. A partir de dos relaciones especificadas, construye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos, esto es, los pares ordenados.

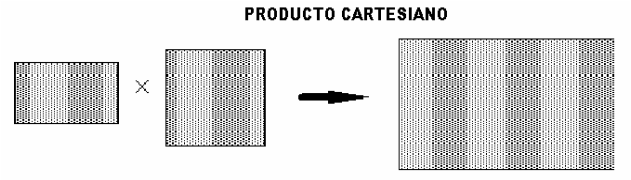


Figura 8. Producto Cartesiano.

Ejemplo: Sean r y s

r

A	B	C
a1	b1	c1
a2	b2	c1
a2	b1	c2

s

D	E	F
d1	e1	f1
d2	e2	f1
d2	e2	f2

r x s

A	B	C	D	E	F
a1	b1	c1	d1	e1	f1
a1	b1	c1	d2	e2	f1
a1	b1	c1	d2	e2	f2
a2	b2	c1	d1	e1	f1
a2	b2	c1	d2	e2	f1
a2	b2	c1	d2	e2	f2
a2	b1	c2	d1	e1	f1
a2	b1	c2	d2	e2	f1
a2	b1	c2	d2	e2	f2

Figura 9. Ejemplo de Producto Cartesiano.

1.3.2 Operadores Especiales

1.3.2.1 Proyección

Es una operación unaria. El resultado es un subconjunto de dominios, permite obtener subrelaciones de otras más grandes seleccionando algunos atributos. Se eliminan, luego, las eneadas repetidas.

Figura 10. Proyección.

Ejemplo:

Sea r una relación con esquema {A, B, C}

r

A	B	C
a1	b1	c1
a2	b2	c2
a2	b1	c2

Podríamos obtener de r:

A	C
a1	c1
a2	c2
a2	c2

Y finalmente:

A	C
a1	c1
a2	c2

Figura 11. Ejemplo de Proyección.

1.3.2.2 Selección

Produce un subconjunto de las aneadas de la relación que cumplen con una condición (simple o compuesta) sobre los valores para uno o varios de los atributos. Extrae las tuplas especificadas de una relación dada.

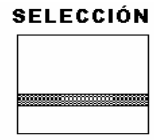


Figura 12. Seleccion.

Por ejemplo, seleccionemos los elementos de r donde existe c2:

r

A	B	C
a1	b1	c1
a2	b2	c1
a2	b1	c2

Y obtendríamos

A	B	C
a2	b2	c2

Figura 13. Ejemplo de Seleccion

1.3.2.3 Reunión o Join

Construye una relación formada por todas las eneadas que aparecen en cualquiera de las dos relaciones especificadas en que se cumple alguna condición en dominios comunes. Se obtiene concatenando una eneada de r con otra de q, de forma que cumpla con una condición en los dominios comunes. Si no hay dominios comunes, esta operación es un producto cartesiano.

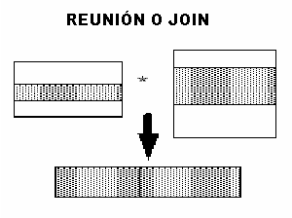


Figura 14. Join.

Ejemplo: Sean q y r dos conjuntos como siguen:

q

A	B	C
a1	b1	c1
a2	b2	c1
a2	b1	c2

r

C	D	E
c1	d1	e1
c1	d2	e2
c3	d2	e2

Obtendríamos:

A	B	C	C	D	E
a1	b1	c1	c1	d1	e1
a2	b2	c1	c1	d2	e2

Figura 15. Ejemplo de Join

1.4 Normalización.

La normalización es una técnica para diseñar la estructura lógica de los datos de un sistema de información en el modelo relacional, desarrollada por E. F. Codd en 1972. Es una estrategia de diseño de abajo a arriba: se parte de los atributos y éstos se van agrupando en relaciones (tablas) según su afinidad. Aquí no se utilizará la normalización como una técnica de diseño de bases de datos, sino como una etapa posterior a la correspondencia entre el esquema conceptual y el esquema lógico, que elimine las dependencias entre atributos no deseadas.

Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.
- Uno de los conceptos fundamentales en la normalización es el de dependencia funcional.

En el proceso de normalización se debe ir comprobando que cada relación (tabla) cumple una serie de reglas que se basan en la clave primaria y las dependencias funcionales. Cada regla que se cumple aumenta el grado de normalización. Si una regla no se cumple, la relación se debe descomponer en varias relaciones que sí la cumplan.

La normalización se lleva a cabo en una serie pasos. Cada paso corresponde a una forma normal que tiene unas propiedades. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto (más fuerte) y, por lo tanto, son menos vulnerables a las anomalías de actualización. Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización.

1.4.1 Primera forma normal (1FN)

Una relación está en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos; un grupo repetitivo será el atributo o grupo de atributos que tiene múltiples valores para cada tupla de la relación, las columnas repetidas deben eliminarse y colocarse en tablas separadas. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna.

4. Henry F. Korth, “Fundamentos de bases de datos” (México, Mc Graw Hill, 1999) 124-135 pp.

Por ejemplo, la siguiente tabla no está en 1FN:

IdPedido	IdCliente	IdVendedor	Fecha_Pedido	Fecha_Entrega	Productos
10548	Wilman	5	25/02/02	26/02/02	3 chorizos, 2 ajos, 1 cebolla
10549	Toldar	6	01/04/02	05/04/02	4 botes, 2 fresas
10550	Canair	3	05/06/02	06/06/02	1 revista, 2 zumos, 1 pan
10551	Suprem	5	07/07/02	07/07/02	2 habas, 1 pan, 2 quesos
10552	Wilman	1	08/07/02	09/07/02	1 bote, 2 panes

Figura 16. Tabla no Normalizada (1FN)

Dependencia funcional

Uno de los conceptos fundamentales en la normalización es el de *dependencia funcional*. Una dependencia funcional es una relación entre atributos de una misma relación (tabla).

Se dice que un atributo B depende funcionalmente de A (A -> B) si cada valor de A se corresponde con un único valor de B o, visto de otra manera, si dado A puedo obtener B. Un caso típico podría ser DNI -> Nombre, pues dado un DNI puedo obtener el nombre de la persona con ese DNI.

Veamos unas reglas que se pueden realizar entre atributos para poder obtener dependencias funcionales adicionalmente. Supongamos que T es una tabla relacional y X, Y, Z son subconjuntos de atributos de la tabla T.

Reflexividad: Si los valores de un subconjunto de atributos Y están incluidos dentro de un subconjunto de atributos X, se dice que X depende funcionalmente de Y (Y -> X)

Aumentación: Si un subconjunto X depende funcionalmente de otro Y, dicha dependencia se mantendrá aunque se añada otro atributo a los dos subconjuntos (X -> Y entonces X.a -> Y.a)

Transitividad: Si Y depende funcionalmente de X y Z depende funcionalmente de Y, entonces Z depende funcionalmente de X (X -> Y e Y -> Z entonces X -> Z). Por ejemplo, DNI -> NOMBRE y NOMBRE -> DIRECCIÓN, luego DNI -> DIRECCIÓN

1.4.2 Segunda forma normal (2FN)

Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.

La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN.

4. Henry F. Korth, “Fundamentos de bases de datos” (México, Mc Graw Hill, 1999) 124-135 pp.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria.

Para convertir una tabla que no esté en 2FN a 2FN se creará una tabla con la clave y todas sus dependencias funcionales totales y otra tabla con la parte de la clave que tiene dependencias con los atributos secundarios.

Por ejemplo, la siguiente tabla no está en 2FN:

NombreProducto	NombreProveedor	Categoria	TeléfonoProveedor
Diarios	Exotic Kiosk	Prensa	968582222
Revistas	Exotic Kiosk	Prensa	968582222
Habas	Tnj export	Alimentacion	975869999
Botes	Tnj export	Bebidas	975869999

Figura 17. Tabla no Normalizada (2FN)

Ya que el campo "TeléfonoProveedor" no es dependiente de la clave candidata "NombreProducto, "NombreProveedor" sino únicamente de "NombreProveedor". Se trata de no representar dos entidades distintas en una sola tabla.

En este ejemplo, reorganizaríamos los datos de la siguiente manera:

Tabla Productos:

IdProducto	NombreProduct	Categoria	IdProveedor
1	Diarios	Prensa	1
2	Revistas	Prensa	1
3	Habas	Alimentación	2
4	Botes	Alimentación	2

Figura 18. Tabla Productos.

Tabla Proveedores:

IdProveedor	NombreProveedor	TeléfonoProveedor
1	Exotic Kiosk	968582222
2	Tnj Export	975869999

Figura 19. Tabla Proveedores.

1.4.3 Tercera forma normal (3FN)

Una relación está en tercera forma normal si, y sólo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende de la clave primaria.

4. Henry F. Korth, “Fundamentos de bases de datos” (México, Mc Graw Hill, 1999) 124-135 pp.

Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

Por ejemplo, la siguiente tabla no está en 3FN:

Tabla Atletas:

Licencia	Nombre	Club	Edad	Categoria
189585	Pepe	Alcoy	12	Junior
205888	Tomas	Jaca	10	Cadete
748523	Andres	Almansa	9	Alevín

Figura 20. Tabla no Normalizada (3FN)

Ya que, dado un número de licencia, podemos obtener la edad del inscrito, y dada la edad del inscrito, podemos averiguar la categoría a la que pertenece: tenemos una dependencia funcional transitiva. Evidentemente, dado el número de licencia podemos averiguar la categoría pero lo importante aquí es que la categoría depende de un atributo que no forma parte de la clave. Para normalizar, descompondremos la tabla en las siguientes:

Tabla Atletas:

Licencia	Nombre	Club	Edad
189585	Pepe	Alcoy	12
205888	Tomas	Jaca	10
748523	Andres	Almansa	9

Figura 21. Tabla Atletas.

Tabla Categorías:

Edad	Categoria
9	Alevín
10	Cadete
12	Junior

Figura 22. Tabla Categorias

4. Henry F. Korth, “Fundamentos de bases de datos” (México, Mc Graw Hill, 1999) 124-135 pp.

1.5 Sistemas Manejadores de Bases de Datos Relaciónales (RDBMS).

1.5.1 ¿Qué es un RDBMS?

Entre la base de datos física (es decir, los datos tal y como están almacenados en la realidad) y los usuarios del sistema, existe un nivel de programas, denominado, manejador de bases de datos (MBD) o, en la mayoría de los casos, el sistema administrador de bases de datos DBMS (Data Base Management System) ya mencionado anteriormente.

Un RDBMS es el conjunto de programas que permiten la definición, manipulación y control de acceso para una o varias bases de datos.

Algunas características de los RDBMS son:

- Facilitan la integridad, seguridad y acceso de los datos.
- Los datos se almacenan como mínima redundancia.
- Las aplicaciones son independientes del almacenamiento físico de los datos.

Un DBMS debe permitir las siguientes condiciones en una base de datos:

- Los datos han de estar almacenados juntos.
- Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento.
- Los datos son compartidos por diferentes usuarios y programas de aplicación; existe un mecanismo común para la inserción, actualización, borrado y consulta de los datos.
- Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos.
- Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes DBMS.
- Conceptualmente lo que sucede en un RDBMS cuando un usuario realiza alguna petición, se presenta lo siguiente:
- El usuario solicita alguna petición a la base de datos empleando algún sublenguaje de datos determinado (SQL).
- El RDBMS interpreta esa solicitud y la analiza.
- El RDBMS inspecciona en orden el esquema externo de ese usuario, la correspondencia externa/conceptual asociada, el esquema conceptual, la correspondencia conceptual/interna y la definición de la estructura de almacenamiento.
- El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada y devuelve una respuesta al usuario.

5. Adoración de Miguel Castaño, “Diseño de bases de datos relacionales” (México, Alfaomega-Rama, 1999) 180-208 pp.

1.5.2 Esquemas de seguridad en el RDBMS:

Dentro del Sistema Manejador de Base de Datos (DBMS) podemos encontrar un acceso multicapas, como el que se muestra en la figura 23.

- El usuario final debe tener una cuenta válida dentro de la capa del servidor (DBMS). Seguridad a Nivel Servidor
- El usuario final debe ser un usuario válido dentro de la capa de la base de datos. Seguridad a Nivel de Base de Datos
- El usuario final deberá tener permiso dentro de la capa de los datos. Seguridad a Nivel de Permisos sobre Objetos y Comandos.

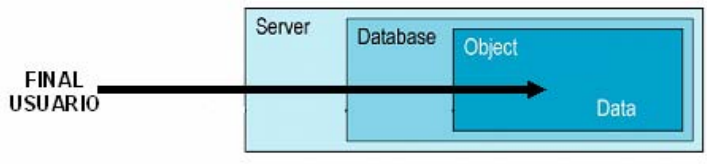


Figura 23. Esquemas de seguridad

1.5.3 Componentes de un RDBMS

El Sistema Manejador de Bases de Datos Relaciónales se compone de:

- **Un DDL o Lenguaje de Definición de Datos**: este es utilizado para crear, eliminar o modificar tablas, índices, vistas, triggers, procedimientos; es decir, nos permite definir la estructura de la base de datos mediante comandos como crear (*Create*), eliminar (*Drop*), o alterar (*Alter*).

  -**create**. Utilizado para crear nuevas bases de datos, tablas, campos, índices, vistas, defaults, reglas, procedimientos, procedimientos, triggers.

  -alter. Utilizado para modificar la estructura de una tabla para agregar campos o constrainst.

  -drop. Utilizado para eliminar bases de datos, tablas, campos, índices, vistas, defaults, reglas, procedimientos, procedimientos, triggers.

- **Un DML o Lenguaje de Manipulación de Datos**: este es utilizado para realizar la consulta y edición de la información contenida en la base de datos, esto implica: seleccionar, insertar, borrar, modificar.

5. Adoración de Miguel Castaño, “Diseño de bases de datos relacionales” (México, Alfaomega-Rama, 1999) 180-208 pp.

Los DML se distinguen por sus sublenguajes de recuperación subyacentes; se pueden distinguir dos tipos de DML, el procedural y el no procedural. La principal diferencia entre ambos es que en los lenguajes procedurales se tratan los registros individualmente, mientras que en uno no procedural se opera sobre un conjunto de registros.

Las instrucciones relacionadas con este componente son:

-select. Permite realizar consultas a la base de datos.

-insert. Empleado para agregar registros a una tabla.

-update. Utilizado para modificar los valores de los campos de una tabla.

-delete. Utilizado para modificar los valores de los campos de una tabla.

- **Un DCL o Lenguaje de Control de Datos**: este es utilizado para la definición de los privilegios de control de acceso y edición a los elementos que componen la base de datos (seguridad), es decir, permitir o revocar el acceso.

  Los permisos a nivel base de datos pueden otorgarse a usuarios para ejecutar ciertos comandos dentro de la base o para que puedan manipular objetos y los datos que puedan contener estos.

  Las instrucciones relacionadas con este componente son:

  -grant. Permite otorgar permisos a los usuarios sobre los objetos definidos en la base de datos, así como las operaciones a utilizar sobre ellos.

  -revoke. Permite revocar permisos sobre los objetos definidos en la base de datos y las operaciones sobre los mismos.

- Un **DD o Diccionario de Datos**: El contenido del diccionario puede considerarse como "datos acerca de los datos" (los cuales comúnmente reciben el nombre de metadatos), es decir, definiciones de otros objetos de la base de datos.

En particular, todos los diversos esquemas (externo, conceptual e interno), se almacenan físicamente en el diccionario, tanto en forma fuente como en forma objeto. Un diccionario amplio incluirá también las referencias cruzadas que indican, por ejemplo que partes de datos utiliza cada programa, que informes necesita cada departamento, etc. De hecho, el diccionario puede integrarse a la base de datos que describe y por tanto, incluir su propia descripción.

5. Adoración de Miguel Castaño, "Diseño de bases de datos relacionales" (México, Alfaomega-Rama, 1999) 180-208 pp.

Debe ser posible consultar el diccionario de la misma manera que cualquier otra base de datos, de modo que, por ejemplo, el DBA pueda describir con facilidad que programas tienen probabilidad de ser afectados por un cambio propuesto al sistema.

Sus principales funciones son las siguientes:

- Describe todos los elementos en el sistema.
- Los elementos se centran en los datos.
- Comunica los mismos significados para todos los elementos del sistema.
- Documenta las características del sistema.
- Localiza errores y omisiones del sistema.

Arquitectura del RDBMS:

De acuerdo a lo descrito, la arquitectura general del RDBMS queda de la siguiente forma:

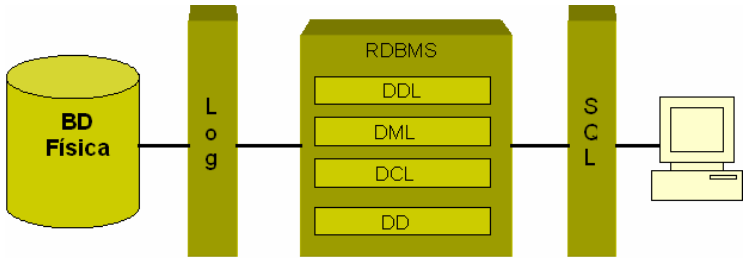


Figura 24. Arquitectura de un RDBMS

Y la arquitectura cliente servidor de un RDBMS queda así:

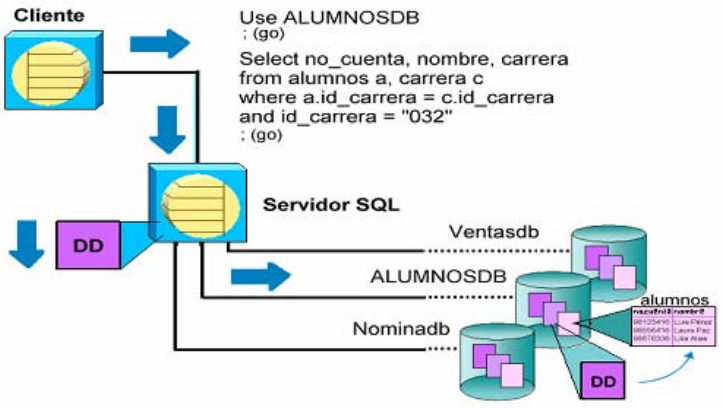


Figura 25. Arquitectura cliente/servidor de un RDBMS

5. Adoración de Miguel Castaño, “Diseño de bases de datos relacionales” (México, Alfaomega-Rama, 1999) 180-208 pp.

1.6 SQL (Structured Query Language)

SQL (Structured Query Language ó Lenguaje Estructurado de Consulta) es un lenguaje de consulta para bases de datos, siendo adoptado como estándar de la industria en 1986. Desde entonces se han realizado revisiones al estándar para incorporar nueva funcionalidad conforme la industria de las bases de datos lo va requiriendo. Una de las revisiones más importantes fue la de 1992, conocida como ANSI SQL92.

La ventaja de la adopción del ANSI SQL, es que los diversos RDBMS (Relational DataBase Management System; Sistema Manejador de Bases de Datos Relacional) tienen que acoplarse al estándar, permitiendo así una mayor compatibilidad entre ellos. Esto implica que conociendo una variante del SQL, se tienen los conocimientos necesarios para poder utilizar otros RDBMS: MS SQL Server, Oracle, Sybase, Interbase, MySQL, PostgreSQL, DB2, etc.

SQL es un lenguaje fácil de entender ya que su estructura utiliza palabras en inglés, lo que lo hace fácil de aprender y utilizar y las instrucciones se enfocan a qué buscar, dejando al RDBMS la tarea de cómo recuperar la información.

Los componentes del SQL son:

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

1.6.1 Comandos SQL

Existen tres tipos de comandos SQL:

- Los DDL que permiten crear y definir nuevas bases de datos, campos e índices.
- Los DML que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.
- Los DCL que permiten la definición de los privilegios de control de acceso y edición a los elementos que componen la base de datos.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Comandos DML	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

Comandos DDL	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comandos DCL	Descripción
GRANT.	Otorga permisos a los usuarios sobre los objetos definidos en la base de datos, así como las operaciones a utilizar sobre ellos.
REVOKE	Revoca permisos sobre los objetos definidos en la base de datos y las operaciones sobre los mismos.

1.6.2 Cláusulas.

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

1.6.3 Operadores Lógicos

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

1.6.4 Operadores de Comparación

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

Selección de datos.

Las tablas dentro de una base de datos son las estructuras que tienen almacenada la información en forma de registros. Para poder recuperar esa información almacenada, se requiere del comando SELECT de SQL.

El comando SELECT es sumamente útil, ya que a través de él es posible realizar desde una consulta simple que sólo involucra una tabla, hasta una consulta compleja donde intervienen dos o más tablas, varias condiciones, agrupaciones de datos y ordenamientos.

La estructura general de este comando es la siguiente:

SELECT {* | [DISTINCT] <*campo*>, <*campo*>...}
FROM <*nombre_tabla*>, [<*nombre_tabla*>]
[WHERE <*condición*>]
[GROUP BY <*campo*>, <*campo*>,...]
[HAVING <*condición*>]
[ORDER BY <*campo*> [ASC|DESC], <*campo*> [ASC|DESC],...]

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Lo que se puede apreciar en la estructura de la instrucción SELECT, es que nunca debe faltar ni la palabra SELECT, ni FROM. Todos los demás elementos son opcionales.

Cláusula SELECT

Esta cláusula indica que la instrucción a ejecutar es una consulta a la base de datos. SELECT nos permite indicar el nombre de los campos que queremos mostrar en la consulta. En caso de querer mostrar todos los campos de una tabla se emplea el comodín asterisco: *****.

Cuando se realiza una consulta que involucra dos o más tablas, al nombre de cada campo se le antepone el de la tabla a la que pertenece.

<*nombre_tabla*>.<*nombre_campo*>

Es posible utilizar seudónimos (alias) para cambiar el nombre de las columnas mostradas en una consulta, esto puede servir para hacer más legible los resultados mostrados, o porque así lo requiere alguna aplicación. Para colocar los seudónimos es necesario especificarlo mediante la cláusula AS, de la siguiente forma:

<*nombre_campo*> AS <*otro_nombre*>.

Ejemplos:

a) Seleccionar el nombre del empleado de la tabla empleado y el nombre de departamento de la tabla departamento.

```
SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento
```

b) Seleccionar todos los campos de la tabla empleado.

```
SELECT * FROM empleado
```

c) Seleccionar todos los campos de la tabla empleado y el nombre del departamento de la tabla departamento.

```
SELECT empleado.*, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento
```

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Cláusula FROM

La cláusula FROM sirve para indicar las tablas de las cuales se desea mostrar la información. Cuando una consulta involucra dos o más tablas, es indispensable establecer las relaciones que existen entre ellas (join) mediante una cláusula WHERE.

Ejemplos.

a) Seleccionar el nombre del empleado de la tabla empleado y el nombre de departamento de la tabla departamento.

```
SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento
```

b) Seleccionar todos los campos de la tabla empleado.

```
SELECT *
FROM empleado
```

c) Seleccionar todos los campos de la tabla empleado y el nombre del departamento de la tabla departamento.

```
SELECT empleado.*, departamento.nombre
FROM empleado, departamento
where departamento.id_departamento = empleado.id_departamento
```

Uso de alias para tablas.

Cuando una consulta involucra más de una tabla, es necesario especificar de cual tabla es el campo que se desea mostrar. Esto lo podemos ver en el ejemplo c), que involucra a las tablas empleado y departamento. Es frecuente el uso de seudónimos para las tablas, con la finalidad de simplificar la escritura de la consulta. El seudónimo se coloca inmediatamente después del nombre de la tabla. <*tabla*> [seudónimo]

Ejemplo con uso de alias

```
SELECT e.*, d.nombre
FROM empleado e, departamento d
```

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Cláusula WHERE.

La cláusula WHERE permite delimitar los registros que serán mostrados en la consulta, a través de criterios o condiciones. Es posible utilizar los operadores lógicos: OR, AND y NOT para combinar expresiones y refinar el criterio de consulta.

Para escribir condiciones de manera adecuada es muy importante recordar que:

- Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.

- Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

El valor NULL es un valor especial por lo cual se debe tener sumo cuidado cuando se desee utilizar condiciones con NULL. La única forma de comparar contra un valor NULL es utilizar el operador IS o IS NOT.

```
SELECT * FROM empleado WHERE comision = NULL;    --incorrecto
SELECT * FROM empleado WHERE comision is NULL;    --correcto
```

Las expresiones más frecuentes son las que involucran una comparación entre dos elementos.

Igualdad	=	empleado.id_departamento = departamento.id_departamento
Desigualdad	!=	nombre_cargo != 'Director'
Mayor que	>	sueldo < 15000
Menor que	<	edad > 35
Mayor o igual que	>=	sueldo >=15000
Menor o igual que	<=	edad <= 35
Similar a	LIKE	nombre_empleado like 'Al%' (% es un comodín)
Es	IS	edad IS NULL
No es	IS NOT	edad IS NOT NULL

En SQL es posible abreviar la forma de escribir ciertas condiciones:

La expresión:	Se simplifica usando:	Quedando de la siguiente manera:
sueldo >= 10000 AND sueldo	**BETWEEN**	Sueldo between 10000 and 15000

6. Cesar Pérez, "Administración y Análisis de Bases de Datos" (México, Alfaomega-Rama, 2002) 150 – 249 pp.

<=15000		
nombre_cargo = 'Gerente' OR nombre_cargo = 'Presidente' OR nombre_cargo = 'Vicepresidente' OR nombre_cargo = 'Director'	**IN**	nombre_cargo in ('Gerente', 'Presidente', 'Vicepresidente', 'Director')
nombre_cargo != 'Jefe de departamento' AND nombre_cargo != 'Vendedor'	**NOT IN**	nombre_cargo not in ('Jefe de departamento', 'Vendedor')

La comparación de similitud que se hace mediante el uso de la cláusula LIKE, requiere de incluir comodines, que sustituyan uno o varios caracteres.

%	representa 0 o más caracteres.
_	representa 1 carácter

Ejemplos. Finalmente complementando los 3 ejemplos anteriores quedarían de la siguiente manera:

a) Seleccionar el nombre del empleado de la tabla empleado y el nombre de departamento de la tabla departamento.

```
SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento
```

b) Seleccionar todos los campos de la tabla empleado. (En este caso no es necesario colocar ninguna condición adicional por lo que queda sin modificación)

```
SELECT * FROM empleado
```

c) Seleccionar todos los campos de la tabla empleado y el nombre del departamento de la tabla departamento.

```
SELECT empleado.*, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento
```

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Cláusula GROUP BY

En la cláusula GROUP BY se indica el o los campos por los cuales se desea agrupar un conjunto de registros. Comúnmente esta agrupación va acompañada con una serie de funciones que realizan ciertas operaciones sobre el valor de los campos indicados. Estas funciones son conocidas como funciones de agregación o agrupación:

Función	Acción
COUNT(*)	Regresa el número de registros encontrados
COUNT(<*campo*>)	Regresa el número de registros cuyo valor del campo especificado no es nulo
SUM(<*campo*>)	Suma los valores de la columna especificada
AVG(<*campo*>)	Promedia los valores del campo especificado
MIN(<*campo*>)	Regresa el valor mínimo del campo especificado
MAX(<*campo*>)	Regresa el valor máximo del campo especificado

Cláusula HAVING

Esta cláusula es el equivalente a la cláusula WHERE, es decir, especifica un criterio o condición, pero la diferencia radica en que se ocupa únicamente cuando se desea especificar una función de agregación en la condición.

Ejemplos:

Mostrar la clave de departamento y sueldo promedio de sus empleados, de aquellos departamentos cuyo salario promedio sea mayor que 12000.

```
SELECT id_departamento, AVG(sueldo)
FROM empleado
GROUP BY id_departamento
HAVING AVG(sueldo)>12000
```

Es incorrecto usar la cláusula WHERE junto con una función de agregación, la consulta anterior es incorrecta si se escribe:

```
SELECT id_departamento, AVG(sueldo)
FROM empleado
WHERE AVG(sueldo)>12000    --Incorrecto
GROUP BY id_departamento
```

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Cláusula ORDER BY

Esta cláusula nos permite indicar los campos por los cuales se desea ordenar la información mostrada. Es posible indicar si el orden es descendente o ascendente, de manera predeterminada es ascendente. Algunos RDBMS permiten realizar este ordenamiento especificando, en lugar del nombre del campo, la posición de este.

Ejemplos:

Mostrar todos los datos de los empleados ordenados por nombre de manera descendente.

```
SELECT *
FROM empleado
ORDER BY nombre DESC
```

Mostrar clave, nombre del empleado y salario ordenados por salario.

```
SELECT id_empleado, nombre, sueldo
FROM empleado
ORDER BY 3
```

Inserción de datos.

A través de la instrucción INSERT de SQL, se introduce la información a una tabla.

La estructura general de este comando es la siguiente:

INSERT INTO <*tabla*> [(<*nombreCampo1*>, <*nombreCampo2*>, <*nombreCampo3*>...)]
{VALUES (<*valorCampo1*>, <*valorCampo2*>, <*valorCampo3*>...) | <*Expresión select*>}

Cláusula INSERT.

Esta cláusula permite indicar que la operación a realizar es la inserción de un registro.

Cláusula INTO.

Esta cláusula permite indicar la tabla en donde se realizará dicha inserción. Únicamente se puede especificar una tabla a la vez. Después del nombre de la tabla puede o no ir el nombre de los campos donde se va insertar información, esto es opcional, pero es muy recomendable no omitirlos, por que le resta legibilidad a la instrucción.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Si no se especifica el nombre de los campos que se van a insertar, el DBMS identifica que se desea insertar información en cada uno de los campos, en el orden definido por la estructura de la tabla.

Por **ejemplo**, tomando la tabla empleado cuya estructura define 11 campos con el siguiente orden: id_empleado, nombre, apellido_paterno, apellido_materno, edad, sexo, sueldo, porcentaje_comision, fecha_contratación, id_departamento, id_cargo

INSERT INTO empleado	Es equivalente a escribir:	INSERT INTO empleado (id_empleado, nombre, apellido_paterno, apellido_materno, edad, sexo, sueldo, porcentaje_comision, fecha_contratación, id_departamento, id_cargo)

Cláusula VALUES.

Esta cláusula permite especificar los valores a insertar para cada uno de los campos involucrados en la sentencia.
Para especificar los valores a insertar de manera adecuada es muy importante recordar que:

- Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.

- Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

Ejemplo:

Insertar el registro de un nuevo empleado con clave 20, llamada Lorena Aguilar, de 19 años de edad, con un sueldo de 7,000 pesos, teniendo cargo de empleado, entrando el 26 de Abril del 2004 al departamento de recursos humanos.

INSERT INTO empleado (id_empleado id_departamento, id_cargo, nombre, edad, sexo, sueldo, fecha_contratacion) VALUES (20, 2, 1, 'Lorena Aguilar', 'M', 7000.00, '2002-04-26')

En este ejemplo se puede ver que el orden de los campos especificados se encuentran en un orden diferente al de la estructura física de la tabla (cuyo orden es: id_empleado, nombre, edad, sexo, sueldo, porcentaje_comision, fecha_contratación, id_departamento, id_cargo) y además se omite el campo de porcentaje comisión ya que este solo se emplea para los vendedores y admite valores nulos.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Eliminación de registros.

La instrucción **delete** elimina registros de la tabla indicada con la posibilidad de indicar un criterio, en caso de omitirlo, se eliminan todos los registros de la tabla.

La sintaxis es la siguiente:

DELETE FROM <*nombre_tabla*>
[WHERE <*condición*>]

Cláusula DELETE.

La cláusula DELETE permite indicar que la operación a realizar es una eliminación de registros

Cláusula FROM

La cláusula FROM permite especificar la tabla de donde se desea eliminar registros.

Cláusula WHERE

La cláusula WHERE permite delimitar el conjunto de registros a eliminar, si no se especifica esta condición, se eliminaran todos los registros que contenga la tabla especificada.

Actualización de datos.

Para modificar o actualizar los valores de los registros de una tabla se utiliza el comando **UPDATE.** Si no se especifica una condición con la cláusula WHERE, todos los registros que existan en la tabla serán actualizados.

La sintaxis es la siguiente:

UPDATE <*nombre_tabla*>
SET <*campo1*> = <*valor1*>, <*campo2*> = <*valor2*>,....
[WHERE <*condición*>]

Cláusula UPDATE.

La cláusula UPDATE es la que indica que la operación a ejecutar es una actualización. Después de la cláusula se especifica el nombre de la tabla en donde se encuentra la información que deseamos modificar. Sólo se puede especificar una tabla a la vez.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Ejemplos:

Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.

UPDATE empleado
SET sueldo = 27000, edad = 27
WHERE id_empleado = 12

Cláusula SET.

Esta cláusula permite especificar los campos que se desean modificar y su nuevo valor. La cláusula se coloca una sola vez aunque sean varios campos los que se deseen modificar.

Ejemplos:

Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.

UPDATE empleado
SET sueldo = 27000, edad = 27
WHERE id_empleado = 12

Cláusula WHERE.

Esta cláusula permite especificar un criterio para delimitar el conjunto de registros a modificar.

Ejemplos:

Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.

UPDATE empleado
set sueldo = 27000, edad = 27
WHERE id_empleado = 12

Creación de tablas

Una base de datos está compuesta por un conjunto de tablas, que corresponden a las relaciones del modelo relacional. En la terminología usada en **SQL** no se alude a las relaciones, del mismo modo que no se usa el término atributo, pero sí la palabra columna, y no se habla de tupla, sino de línea.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Cláusula CREATE

La sintaxis básica para la creación de una tabla es la siguiente:

CREATE TABLE <*nombre_tabla*>
(
<*nombre_campo1*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>],
<*nombre_campo2*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>],
<*nombre_campo3*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>],
…
<*nombre_campoN*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>]
)

Ejemplo.

```
CREATE TABLE pais
(
  id_pais numeric (3) NOT NULL,
  nombre varchar (100) NOT NULL
)
CREATE TABLE resultado
(
  id_pais1 numeric (3) NOT NULL,
  id_pais2 numeric (3) NOT NULL,
  marcador varchar (40) NOT NULL
)
```

Modificación de tablas

Dependiendo del RDBMS esta modificación de estructura de la tabla ofrece mayor o menor flexibilidad. Por ejemplo, MySQL es de las más flexibles, porque permite cambiar incluso el tipo de dato de una columna mientras que en Sybase o MS SQL Server es necesario eliminar y volver a crear la tabla con la estructura deseada.

La sintaxis varía de un RDBMS, debido a las diferentes características soportadas, sin embargo todos ellos utilizan la cláusula ALTER TABLE para realizar las modificaciones posibles, a una tabla.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

Cláusula ALTER

Modifica el diseño de una tabla ya existente, se pueden modificar los campos o los índices existentes. Su sintaxis es:

ALTER TABLE tabla {ADD {COLUMN tipo de campo[(tamaño)]
[CONSTRAINT índice]
CONSTRAINT índice multicampo} |
DROP {COLUMN campo I CONSTRAINT nombre del índice}}

En donde:

Tabla	Es el nombre de la tabla que se desea modificar.
campo	Es el nombre del campo que se va a añadir o eliminar.
tipo	Es el tipo de campo que se va a añadir.
tamaño	Es el tamaño del campo que se va a añadir (sólo para campos de texto).
índice	Es el nombre del índice del campo (cuando se crean campos) o el nombre del índice de la tabla que se desea eliminar.
índice multicampo	Es el nombre del índice del campo multicampo (cuando se crean campos) o el nombre del índice de la tabla que se desea eliminar.

Operación	**Descripción**
ADD COLUMN	Se utiliza para añadir un nuevo campo a la tabla, indicando el nombre, el tipo de campo y opcionalmente el tamaño (para campos de tipo texto).
ADD	Se utiliza para agregar un índice de multicampos o de un único campo.
DROP COLUMN	Se utiliza para borrar un campo. Se especifica únicamente el nombre del campo.
DROP	Se utiliza para eliminar un índice. Se especifica únicamente el nombre del índice a continuación de la palabra reservada CONSTRAINT.

Ejemplos:

ALTER TABLE Empleados ADD COLUMN Salario CURRENCY
(Agrega un campo Salario de tipo Moneda a la tabla Empleados.)

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

ALTER TABLE Pedidos ADD CONSTRAINT RelacionPedidos FOREIGN KEY (IdEmpleado)
REFERENCES Empleados (IdEmpleado)

Esta sentencia agrega un índice externo a la tabla Pedidos. El índice externo se basa en el campo IdEmpleado y se refiere al campo IdEmpleado de la tabla Empleados. En este ejemplo no es necesario indicar el campo junto al nombre de la tabla en la cláusula *REFERENCES, pues Id_Empleado es la clave principal de la tabla Empleados.*

Eliminación de tablas

Cláusula DROP

Para eliminar una tabla y los datos que contiene, así como sus índices, triggers y permisos se utiliza la siguiente instrucción:

DROP TABLE <*nombre_tabla*>

Ejemplo:

DROP TABLE empleado

1.6.5 Definición de privilegios.

Es tarea del administrador de la base de datos el encargado de asignar o revocar permisos y/o crear usuarios en la base de datos. El manejo de usuario varía considerablemente de un RDBMS a otro, siendo que en algunos es más completo, el esquema del manejo de usuario y permisos, que en otros.

Cláusula GRANT y REVOKE

Se pueden controlar los privilegios a los usuarios de creación de objetos en la base de datos, así como la manipulación de la información contenida en ella, mediante la siguiente sintaxis:

GRANT <*privilegio*> TO <*usuario*>
REVOKE <*privilegio*> FROM <*usuario*>

Donde **GRANT** se utiliza para otorgar privilegios y **REVOKE** para eliminarlos.

Por ejemplo:

Grant all to admon
Revoke create table from capturista.

6. Cesar Pérez, “Administración y Análisis de Bases de Datos” (México, Alfaomega-Rama, 2002) 150 – 249 pp.

1.7 Administración de Bases de Datos

Administrador de la Base de Datos. Es la persona encargada de definir y controlar las bases de datos corporativas, además proporciona asesoría a los desarrolladores, usuarios y ejecutivos que la requieran. Es la persona o equipo de personas profesionales responsables del control y manejo del sistema de base de datos, generalmente tienen experiencia en DBMS, diseño de bases de datos, sistemas operativos, comunicación de datos, hardware y programación.

1.7.1 Funciones del Administrador de la Base de Datos.

La característica más importante que debe poseer es un conocimiento profundo de las políticas y normas de la empresa, así como el criterio de la empresa para aplicarlas en un momento dado. La responsabilidad general del DBA es facilitar el desarrollo y el uso de la Base de Datos dentro de las guías de acción definidas por la administración de los datos.

El Administrador de Bases de Datos es responsable primordialmente de:

1.7.1.1 Administrar la estructura de la Base de Datos.

Esta responsabilidad incluye participar en el diseño inicial de la base de datos y su puesta en práctica así como controlar, y administrar sus requerimientos, ayudando a evaluar alternativas, incluyendo los DBMS a utilizar y ayudando en el diseño general de la bases de datos.

Una vez diseñada las bases de datos, es puesta en práctica utilizando productos del DBMS, procediéndose entonces a la creación de los datos. El DBA participa en el desarrollo de procedimientos y controles para asegurar la calidad y la alta integridad de la BD.

1.7.1.2 Administración de la Actividad de Datos.

El DBA no es usuario del sistema, no administra valores de datos; sino la actividad de datos; protege los datos, no los procesa. Dado que la base de datos es un recurso compartido, el DBA debe proporcionar estándares, guías de acción, procedimientos de control y la documentación necesaria para garantizar que los usuarios trabajen en forma cooperativa y complementaria al procesar datos en la bases de datos.

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

1.7.1.3 Administrar el Sistema Manejador de Base de Datos.

Existe una gran actividad al interior de un DBMS. La concurrencia de múltiples usuarios requiere la estandarización de los procesos de operación; el DBA es responsable de estas especificaciones y de asegurarse que estas lleguen a quienes concierne. Todo el ámbito de la base de datos se rige por estándares, desde la forma de como se captura la información (tipo de dato, longitud, formato), como es procesada y presentada. El nivel de estandarización alcanza hasta los aspectos más internos de la base de datos; como se accesa a un archivo, como se determinan los índices primarios y auxiliares, registros, etc.

El DBA debe procurar siempre que los estándares que serán aplicados beneficien también a los usuarios, privilegiando siempre la optimización en la operación del DBMS y el apego de las políticas de la empresa. Entre las funciones del DBA se encuentra la de revisar los estándares periódicamente para determinar su operatividad, ajustarlos, ampliarlos o cancelarlos y hacer que éstos se cumplan.

1.7.1.4 Asegurar la Confiabilidad de la Base de Datos

Se trata de realizar un sistema de bases de datos lo suficientemente robusto para que sea capaz de recuperarse frente a errores o usos inadecuados. Se deben utilizar gestores con las herramientas necesarias para la reparación de los posibles errores que las bases de datos pueden sufrir, por ejemplo tras un corte inesperado de luz.

1.7.1.5 Confirmar la Seguridad de la Base de Datos.

Coordinar las nuevas propuestas para realizar ajustes en los derechos de acceso a datos compartidos y aplicaciones específicamente propuestas, serían analizados en conjunto con los supervisores o directivos de las áreas involucradas para determinar si procede pudieran aparecer problemas cuando dos o más grupos de usuarios quedan autorizados para notificar los mismos datos. Uno de tales conflictos es el de la actualización perdida; este ocurre cuando el trabajo de un usuario queda sobrescrito sobre por el de un segundo usuario. El DBA queda responsabilizado para identificar la posible ocurrencia de dichos problemas así como de crear normas y procedimientos para su eliminación. Se obtendrán este tipo de garantías cuando el DBMS sea capaz de implementar las restricciones aplicables al acceso concurrente, y este sea utilizado adecuadamente por programadores y usuarios; para borrar lo anterior, se hace indispensable el apego a los estándares el seguimiento de instructivos y manuales y las reglas establecidas para los diversos procesamientos y procedimientos que se llevan acabo.

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

Entre las alternativas mas utilizadas por el DBA para tratar de resolver o minimizar este problema se encuentran las siguientes:

- Restringir el acceso a los procedimientos para ciertos usuarios.
- Restringir al acceso a los datos para ciertos usuarios procedimientos y/o datos.
- Evitar la coincidencia de horarios para usuarios que comparten.

Las técnicas de recuperación son otra función esencial del DBA al administrar la actividad de datos. A pesar de que el DBMS lleva a cabo una parte del proceso de recuperación, los usuarios determinan en forma critica la operatividad de esos sistemas de protección. El DBA debe anticipar fallas y definir procedimientos estándares de operación; los usuarios deben saber que hacer cuando el sistema este caído y que es lo primero que debe realizarse cuando el sistema este puesto en marcha nuevamente. El personal de operación deberá saber como iniciar el proceso de recuperación de la BD y que copias de seguridad utilizar; como programar la reejecución del tiempo perdido y de las tareas pendientes; es importante también establecer un calendario para llevar a cabo estas actividades sin afectar a otros sistemas dentro de la organización que hagan uso de los mismos recursos de computo. Destacan por su importancia en el proceso de recuperación y a su vez en la atención que prestan a otros sectores de la organización. Los dispositivos de comunicación remota, los sistemas de interconexión y otros accesorios de uso compartido.

El DBA es el responsable de la publicación y mantenimiento de la documentación en relación con la actividad de los datos, incluyendo los estándares de la BD, los derechos de recuperación y de acceso a la BD, los estándares para la recuperación de caídas y el cumplimiento de las políticas establecidas. Los productos DBMS más populares que se encuentran en el mercado proporcionan servicios de utilerías para ayudar al DBA en la administración de los datos y su actividad. Algunos sistemas registran en forma automática los nombres de los usuarios y de las aplicaciones a las que tienen acceso así como a otros objetos de la BD. Incorpora también utilerías que permitan definir en el diccionario de datos las restricciones para que determinadas aplicaciones o módulos de ellas sólo tengan acceso a segmentos específicos de la BD.

1.7.1.6 Mantener la Integridad de los Datos.

Una base de datos debe protegerse de accidentes tales como los errores en la entrada de los datos o en la programación, del uso mal intencionado de la base de datos y de los fallos del hardware o del software que corrompen los datos. La protección contra accidentes, que ocasiona inexactitudes en los datos, es parte del objetivo de garantizar la integridad de los datos. Estos accidentes incluyen los fallos durante el procesamiento de las transacciones, los

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

errores lógicos que infringen la suposición de que las transacciones preservan las restricciones de consistencia de la base de datos y las anomalías debido al acceso concurrente en la base de datos (acceso concurrente). La integridad, se encarga de asegurar que las operaciones ejecutadas por los usuarios sean correctas y mantengan la consistencia de la base de datos.

1.7.1.7 Mantener la Disponibilidad de los Datos. La posibilidad de fallos de hardware o de software requiere procedimientos de recuperación de la base de datos. Tiene que proporcionar medios para el restablecimiento de las bases de datos que se hayan corrompido por desperfectos del sistema, a un estado uniforme.

1.7.1.8 Especificación de las Restricciones de Integridad de los Datos. Las restricciones de integridad se mantienen en una estructura especial del sistema que consulta el gestor de la base de datos cada vez que se tiene lugar una actualización en el sistema. Estos son algunos métodos para asegurar la integridad de los datos:

- Privilegios:
  - Base de datos.
  - Tabla.
  - Columna
- Integridad de identidad, semántica y referencial.
- Vistas.

1.7.1.9 Administrar la Concurrencia. La administración de la concurrencia involucra como los datos son consultados y actualizados en un ambiente multiusuario. Existen dos tipos de control de la concurrencia:

- Concurrencia de Lectura: (Instrucción SELECT)
- Administrada a través de los niveles de aislamiento.
- Concurrencia de Actualización: Instrucciones INSERT, DELETE y UPDATE.

1.7.1.10 Optimización del Acceso a los Datos

Esto se logra creando:

- Índices.
- Estadísticas de actualización.
- Distribución de datos.

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

1.7.1.11 Definir el Esquema Conceptual. Es tarea del administrador de datos decidir con exactitud cual es la información que debe mantenerse en la base de datos, una vez identificado los datos a almacenar en un nivel abstracto, el dba debe crear a continuación el esquema conceptual correspondiente, empleando el DDL conceptual.

1.7.1.12 Definir el Esquema Interno. El DBA debe definir la representación de la información en la base de datos almacenada (diseño físico). Debe crear la definición de estructura de almacenamiento correspondiente (esquema interno) con el DDL interno y definir la correspondencia entre los esquemas interno y conceptual.

1.7.1.13 Vincularse con los Usuarios. El DBA debe encargarse de la comunicación con los usuarios, garantizar la disponibilidad de los datos que requieren y escribir y/o ayudar a los usuarios a escribir los esquemas externos necesarios, empleando el DDL externo aplicable.

1.7.1.14 Procedimientos de Respaldo y Recuperación. El DBA debe definir un plan de recuperación adecuado que incluya descarga o vaciado periódico de la base de datos en un medio de almacenamiento de respaldo, y procedimientos para cargar otra vez la base de datos a partir del vaciado más reciente cuando sea necesario.

1.7.1.15 Supervisar el Desempeño y Responder a cambios en los Requerimientos. El DBA debe organizar el sistema de modo que se obtenga el desempeño que sea "el mejor para la empresa", y realizar los ajustes apropiados cuando cambien los requerimientos.

1.7.1.16 Concesión de Autorización para el Acceso a los Datos. La concesión de diferentes tipos de autorización, permite al administrador de la base de datos regular que partes de la base de datos van a poder ser accedidas por varios usuarios.

1.7.1.17 Definición de esquema. Es el esquema original de la base de datos se crea escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.

1.7.1.18 Definición de la estructura de almacenamiento del método de acceso. Estructuras de almacenamiento y de acceso adecuados se crean escribiendo un conjunto de definiciones que son traducidas por el compilador del lenguaje de almacenamiento y definición de datos.

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

1.8 POLÍTICAS.

Las políticas son guías para orientar la acción; son lineamientos generales a observar en la toma de decisiones, sobre algún problema que se repite una y otra vez dentro de una organización. En este sentido, las políticas son criterios generales de ejecución que auxilian al logro de los objetivos y facilitan la implementación de las estrategias.

1.8.1 Clasificación de las políticas

- **Estratégicas o generales**.

Se formulan al nivel de alta gerencia y su función es establecer y emitir lineamientos que guíen a la empresa como una unidad integrada. Ejemplo: "Los empleados que laboran en la empresa tendrán la posibilidad de ascender de puesto, de acuerdo con su eficiencia y antigüedad".

- **Tácticas o departamentales.**

Son lineamientos específicos que se refieren a cada departamento. Ejemplo: "El departamento de producción determinara los turnos de trabajo conforme a sus necesidades, siguiendo las disposiciones legales".

- **Operativas o específicas.**

Se aplican principalmente en las decisiones que tienen que ejecutarse en cada una de las unidades de las que consta un departamento. Ejemplo: "Sección de tornos; de ocurrir una falla en el equipo, es conveniente reportarla inmediatamente al supervisor en turno o en su caso, al departamento de mantenimiento".

Las políticas, no interesando su nivel, deben estar interrelacionadas y contribuir a lograr las aspiraciones de la empresa; asimismo, su redacción debe ser clara, accesible y de contenido realista, de tal forma que su interpretación sea uniforme.

1.8.2 Importancia de las políticas

- Facilitan la delegación de autoridad.

- Motivan y estimulan al personal al dejar a su libre arbitrio ciertas decisiones.

- Evitan pérdidas de tiempo a los superiores, al minimizar las consultas innecesarias que pueden hacer sus subordinados.

- Otorgan un margen de libertad para tomar decisiones en determinadas actividades.

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

- Contribuyen a lograr los objetivos de la empresa.

- Proporcionan uniformidad y estabilidad en las decisiones.

- Indican al personal como debe actuar en sus operaciones.

- Facilitan la inducción del nuevo personal.

1.8.3 Lineamientos para la formulación de políticas

a) Establecerlos por escrito y darles validez.

b) Redactarse claramente y con precisión.

c) Darse a conocer a todos los niveles donde se va a interpretar y aplicar.

d) Coordinarse con las demás políticas.

e) Revisarse periódicamente.

f) Ser razonables y aplicables en la práctica.

g) Estar acordes con los objetivos de la empresa.

h) Ser flexibles.

Ejemplo:

Políticas de Administración del Espacio en los Ambientes de Trabajo con Oracle

En este apartado se presentarán las políticas para la administración del espacio de las bases de datos.

Petición del espacio. Las peticiones de espacio para las bases de datos o de copia de bases de datos (ya sean totales o parciales), junto con las fechas en que dichas operaciones se requieren, deberán hacerse por los responsables de los diferentes modelos indicándolo así en las formas de "Planeación de Proyectos" que el Departamento de Ingeniería de Información proporciona. Estas formas sirven para que DII lleve un mejor control de la calendarización de actividades importantes. Se debe de especificar la(s) base(s) de datos para la cual se hace el requerimiento.

Este tipo de requerimiento debe hacerse de acuerdo con las siguientes reglas:

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

Dos revisiones por semestre. A inicios de cada semestre (febrero, agosto) se deberá de recibir los requerimientos de espacio para las máquinas administrativas.

A mediados de cada semestre (mediados de octubre, mediados de abril) se deberá revisar con los departamentos de desarrollo estos requerimientos en caso de que necesiten ajustarse.

El Departamento de Ingeniería de Información revisará estos requerimientos con el Departamento de Tecnología Computacional y de Información para que se realicen las compras de disco que sean necesarias.

Especificación de las necesidades. DII revisará el detalle de cada una de las peticiones de espacio con el responsable de cada modelo. El responsable de cada modelo deberá entregar a DII la siguiente información:

Estimado inicial. Para cada tabla se deberá de entregar el estimado inicial del número de registros que va a contener, junto con la fecha en que esta carga se va a realizar.

Estimados de crecimiento. Para cada tabla se deberá entregar el número estimado de registros que se espera vaya creciendo la misma, especificando la periodicidad de este crecimiento.

Planeación del espacio. DII, tomando en cuenta los datos anteriores, revisará con DTCI la disponibilidad de espacio actual en los equipos y la llegada de los nuevos discos. El DBA deberá planear con base a la información proporcionada por los responsables de modelo, los tamaños de los *tablespaces* y revisar con el administrador del equipo el alojamiento de los *tablespaces* en los *file systems.*

Aviso de conflictos. DII deberá revisar el calendario de operaciones con bases de datos para descubrir posibles problemas al ejecutar las operaciones de crecimiento o copia de bases de datos en el orden que lo necesitan los diferentes responsables de modelo.

DII deberá dar a conocer estos problemas y posibles soluciones a la brevedad posible después de haber recibido los requerimientos de espacio o copia. De igual manera, DII deberá de estar en contacto con DTCI para poder avisar con tiempo cualquier problema que exista con la obtención de disco que se pueda llegar a necesitar para completar las operaciones.

Monitoreo del uso del espacio. El DBA deberá de realizar un plan de monitoreo del uso del espacio y revisarlo con los responsables de los modelos para estar seguros de que los

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

requerimientos de espacio fueron estimados debidamente. Este monitoreo debe de ser periódico y debe de tenerse al tanto a DTCI de cualquier desviación de las estimaciones que previamente se habían hecho. DII deberá además monitorear el espacio para planear actividades de defragmentación y reorganización, las cuales deberán de planearse y anunciarse a los desarrolladores con al menos 1 semana de anticipación.

Tiempos de entrega de los discos. Es importante tener en cuenta este punto al momento de hacerse la planeación y/o revisión del espacio requerido en las bases de datos, ya sea para crecimiento u otras operaciones. Los discos que se compran al proveedor que se maneja tardan en entregarse de 6 a 8 semanas después de haberse colocado el pedido. Si los discos se piden a un tercero, el tiempo de entrega de los mismos es de 4 semanas después de haber sido colocada la orden.

7 Administración de Bases de Datos, http://www.redcientifica.com/oracle/ (Acceso Noviembre, 2005).

CAPÍTULO II

TECNOLOGÍA ORACLE.

2.1 Introducción

La tecnología Oracle se encuentra prácticamente en todas las industrias, por sus características de compatibilidad, portabilidad, conectividad y capacidad. Así, los cinco principios que manejan en Oracle son: globalización, simplificación, estandarización, automatización e innovación.

Oracle proveen grandes beneficios a sus usuarios, protegiendo inversiones presentes y futuras en tecnología informática.

El éxito alcanzado por Oracle de México desde su instalación, es el resultado directo del liderazgo tecnológico de los productos Oracle, de la amplia adopción de los mismos por parte de los clientes y de la dedicación y profesionalismo de los integrantes de la subsidiaria.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, rápida, integrable, disponible en todas las plataformas (Unix, Linux y Windows) destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Multiplataforma.

Oracle database es la primera base de datos diseñada específicamente para Internet. Hoy en día la información es la herramienta más valiosa con la que cuentan los negocios modernos para ser competitivos dentro del mercado.

Aquellas organizaciones que son capaces de utilizar esa información de forma eficiente son las que reciben los mayores beneficios y que pueden sobrevivir en el difícil ambiente económico actual. El que una organización utilice de manera efectiva esa información depende de la infraestructura con la que cuentan. La database de Oracle permite a las organizaciones guardar, manejar, integrar y utilizar información del negocio de una manera eficiente.

Esta base de datos ha sido desarrollada desde hace mas de 20 años desde las relational database (SQL) hasta la era del Internet. Ha sido diseñada para proveer la solución más completa y de menor costo para cualquier requerimiento de información de negocio y actualmente es la única herramienta capaz de:

9. Oracle, http://www.oracle.com/global/es/tecnologia/index.html (Acceso Enero, 2006).

- Tener información crítica disponible en cualquier momento.
- Dar protección y seguridad a información privilegiada.
- Integrar información de diversas fuentes, incluyendo bases de datos externas y archivos de sistemas.
- Consolidar y administrar todo el contenido de Internet.
- Reducir el tiempo que le toma a un negocio tomar mejores decisiones analizando la información más rápido.
- Permitir a la organización desarrollar e implementar “business solutions” rápidamente.

9. Oracle, http://www.oracle.com/global/es/tecnologia/index.html (Acceso Enero, 2006).

2.2 Componentes principales de la arquitectura de Oracle

La arquitectura general de Oracle consiste de varios procesos corriendo en la máquina donde reside la instancia, más los espacios de memoria dedicados a ejecutar procesos específicos o al almacenaje de información de cada proceso y la base de datos física propiamente tal, con sus archivos de control, de datos y de transacciones.

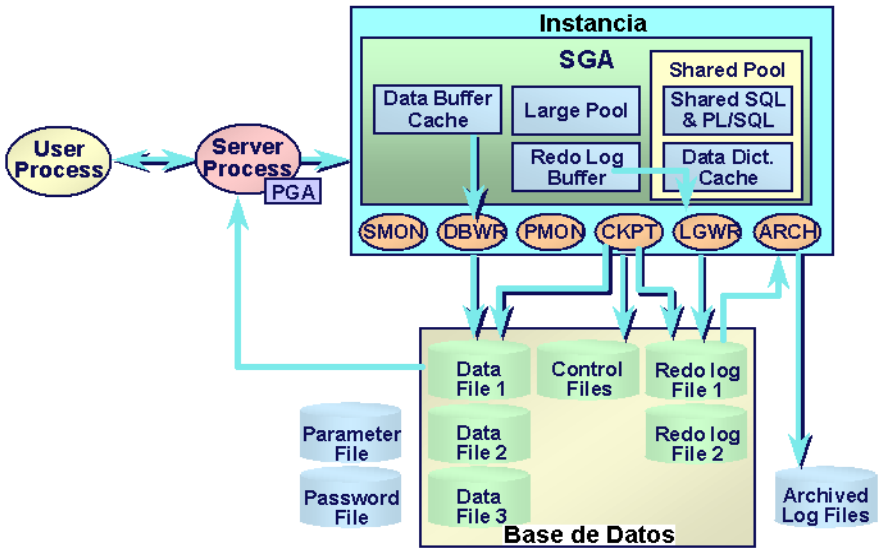


Figura 26. Arquitectura general del RDBMS Oracle

2.2.1 La instancia de Oracle

Una instancia de Oracle está conformada por varios procesos y espacios de memoria compartida que son necesarios para acceder a la información contenida en la base de datos.

La instancia está conformada por procesos del usuario, procesos que se ejecutan en el background de Oracle y los espacios de memoria que comparten estos procesos.

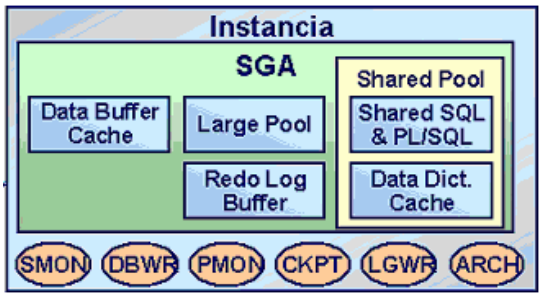


Figura 27. Arquitectura de la Instancia de Oracle

Procesos de la Instancia

Según lo que se advierte en la figura, los procesos que se implementan en una instancia de Oracle y su función principal son los siguientes:

- DBWR (*database writer*): Es el responsable de la escritura en disco de toda la información almacenada en los *buffers* de bloques que no se han actualizado.
- LGWR (*log writer*): Es el responsable de escribir información desde el *buffer de log* hacia el *archivo redo log*.
- CKPT (*checkpoint*): Es el responsable de advertir al proceso DBWR de efectuar un proceso de actualización en el disco de los datos mantenidos en memoria, incluyendo los *datafiles* y *control files* (para registrar el *checkpoint*). Este proceso es opcional, si no está presente, es el proceso LGWR quien asume la responsabilidad de la tarea.
- PMON (*process monitor*): Su misión es monitorizar los procesos del servidor y tomar acciones correctivas cuando alguno de ellos se interrumpe en forma abrupta, limpiando la caché y liberando los posibles recursos que pudieran estar asignados en ese momento. También es responsable por el restablecimiento de aquel proceso que se ha interrumpido bruscamente.
- SMON (*system monitor*): Levanta una instancia cuando se le da la instrucción de partida (al comienzo del trabajo, encontrándose previamente en *shutdown*). Enseguida limpia los segmentos temporales y recupera las transacciones que pudieran haberse interrumpido debido a una falla del sistema. Además disminuye la fragmentación del sistema agrupando aquellas extensiones libres que existen dentro de la base de datos.
- ARCH (*archiver*): La función de este proceso es la de *respaldar* la información almacenada en los archivos *redo log* cuando éstos se llenan. Este proceso está siempre activo cuando se ha establecido el modo ARCHIVELOG. Si el sistema no está operando en este modo se hace más difícil recuperar el sistema sin problemas luego de una falla general.

10. Arquitectura Oracle, http://www.infor.uva.es/oracledba/capitulo1.html (Acceso Febrero, 2006).

2.2.2 El área global del sistema (SGA)

El SGA es un área de memoria compartida que se utiliza para almacenar información de control y de datos de la instancia. Se crea cuando la instancia es levantada y se borra cuando ésta se deja de usar (cuando se hace *shutdown*). La información que se almacena en esta área consiste de los siguientes elementos, cada uno de ellos con un tamaño fijo.

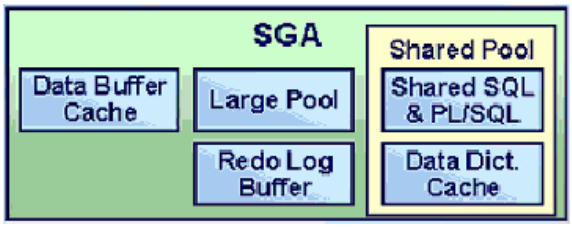


Figura 28. Arquitectura del Área Global del Sistema

- El buffer de caché (database buffer cache) Almacena los bloques de datos utilizados recientemente (se hayan o no confirmado sus cambios en el disco). Al utilizarse este buffer se reducen las operaciones de entrada y salida y por esto se mejora el rendimiento.
- El buffer de *redo log*: Guarda los cambios efectuados en la base de datos. Estos *buffers* escriben en el archivo físico de redo log tan rápido como se pueda sin perder eficiencia. Este último archivo se utiliza para recuperar la base de datos ante eventuales fallas del sistema.
- El área *shared pool*: Esta sola área almacena estructuras de memoria compartida, tales como las áreas de código SQL compartido e información interna del diccionario. Una cantidad insuficiente de espacio asignado a esta área podría redundar en problemas de rendimiento. En resumen, contiene las áreas del *caché de biblioteca* y del caché del *diccionario de datos*.
- El caché de biblioteca se utiliza para almacenar código SQL compartido. Aquí se manejan los árboles de *parsing* y el plan de ejecución de las *queries*. Si varias aplicaciones utilizan la misma sentencia SQL, esta área compartida garantiza el acceso por parte de cualquiera de ellas en cualquier instante.
- El caché del diccionario de datos está conformado por un grupo de tablas y vistas que se identifican la base de datos. La información que se almacena aquí guarda relación con la estructura lógica y física de la base de datos. El diccionario de datos contiene información tal como los privilegios de los usuarios, restricciones de integridad definidas para algunas tablas, nombres y tipos de datos de todas las columnas y otra información acerca del espacio asignado y utilizado por los objetos de un esquema.

10. Arquitectura Oracle, http://www.infor.uva.es/oracledba/capitulo1.html (Acceso Febrero, 2006).

2.2.3 El área global de programas (PGA)

Esta área de memoria contiene datos e información de control para los procesos que se ejecutan en el servidor de Oracle (relacionados con la base de datos, por supuesto). El tamaño y contenido de la PGA depende de las opciones del servidor que se hayan instalado.

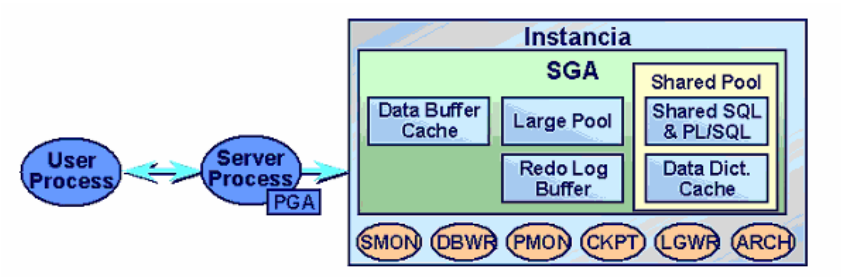


Figura 29. Arquitectura del Área Global de programas

2.2.4 La base de datos

La base de datos de Oracle se compone de la siguiente manera:

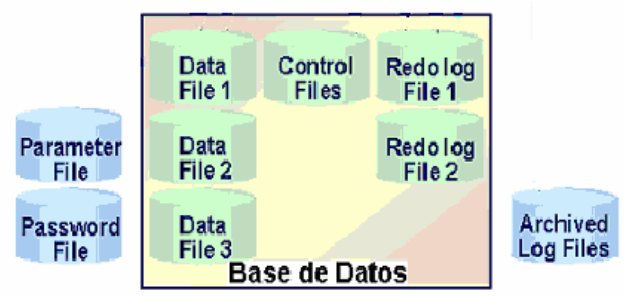


Figura 30. Estructura de la Base de Datos

- Control File Contiene información para mantener y controlar la integridad de la base de datos.
- Data Files Son los archivos donde se almacenan los datos de las aplicaciones.
- Redo Log Files Almacena los cambios hechos en la base de datos con propósito de recuperarlos en caso de falla.
- Archivo de Parámetros. Contiene parámetros y valores que definen las características de la instancia y la base de datos, por ejemplo contiene parámetros que dimensionan el SGA.
- Archivo de Password. Se utiliza para validar al usuario que puede bajar y subir la instancia de Oracle.
- Archivos Archived Log Files. Los Archived Log Files son copias fuera de línea de los archivos Redo Log Files que son necesarios para el proceso de Recovery en caso de falla del medio de almacenamiento.

La base de datos de Oracle tiene una capa lógica y otra física. La capa física consiste de archivos que residen en el disco y los componentes de la capa lógica son estructuras que mapean los datos hacia estos componentes físicos.

2.2.4.1 La Capa Física

Ya se dijo que consiste de archivos físicos que se encuentran en los discos. Estos pueden ser de tres tipos diferentes:

- Uno o más *datafiles*
- Los *datafiles* almacenan toda la información ingresada en una base de datos. Se pueden tener sólo uno o cientos de ellos. Muchos objetos (tablas, índices) pueden compartir varios *datafiles*. El número máximo de *datafiles* que pueden ser configurados está limitado por el parámetro de sistema MAXDATAFILES.
- Dos o más archivos *redo log* (de *deshacer*)
- Los archivos del tipo *redo log* almacenan información que se utiliza para la recuperación de una base de datos en caso de falla. Estos archivos almacenan la historia de cambios efectuados sobre la base de datos y son particularmente útiles cuando se necesita corroborar si los cambios que la base de datos ya ha confirmado se han efectuado realmente en los *datafiles*.
- Uno o más *control files*
- Estos archivos contienen información que se utiliza cuando se levanta una instancia, tal como la información de dónde se encuentran ubicados los *datafiles* y los archivos *redo log*. Estos archivos de control deben encontrarse siempre protegidos.

2.2.4.2 La Capa Lógica

La capa lógica de una base de datos consta de los siguientes elementos:

- Uno o más *tablespaces*
- El esquema de la base de datos (*schema*), el cual consiste de objetos como tablas, clusters, índices, vistas, procedimientos almacenados, triggers, secuencias y otros.

Los Tablespaces y los Datafiles

Como se mencionó, una base de datos se encuentra dividida en una o más piezas lógicas llamadas tablespaces, que son utilizados para separar la información en grupos y así simplificar la administración de los datos. Los tablespaces pueden ocupar uno o más *datafiles*. Si se decide que utilice varios *datafiles*, el administrador del sistema puede gestionar que éstos queden localizados en discos diferentes, lo que aumentará el rendimiento del sistema, principalmente por la mejora en la distribución de la carga de entrada / salida.

10. Arquitectura Oracle, http://www.infor.uva.es/oracledba/capitulo1.html (Acceso Febrero, 2006).

En la figura 31 se aprecia la diferencia entre estos tres conceptos. Una base de datos de ejemplo contiene tres *tablespaces* lógicos (parte superior de la figura) que utiliza para almacenar información del sistema, de los datos del usuario y de los índices de las tablas. Asimismo, existen los espacios físicos (*datafiles*) que guardan esta información en los diferentes discos disponibles y que se señalan en la parte inferior del dibujo.

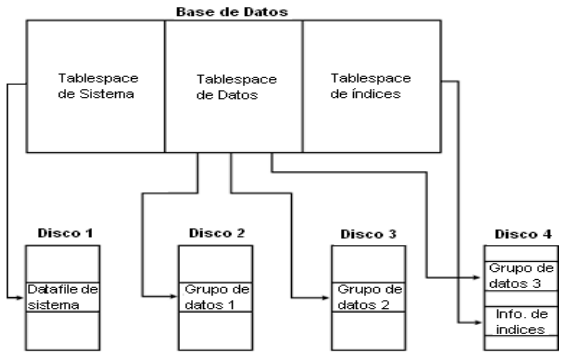


Figura 31. Relación entre la base de datos, los *tablespaces* y los *datafiles*

Segmentos, Extensiones y Bloques

Dentro de los *tablespaces* y *datafiles*, el espacio utilizado para almacenar datos es controlado por el uso de ciertas estructuras; éstas son las siguientes:

- *Bloques:* Un bloque es la unidad de almacenamiento más pequeña en una base de datos Oracle. Contiene una pequeña porción de información (*header*) referente al bloque en sí y el resto a los datos que guarda. Generalmente, un bloque de datos ocupará aprox. 2 KB de espacio físico en el disco (asignación típica).
- *Extensiones:* Es un grupo de bloques de datos. Se establecen en un tamaño fijo y crecen a medida que van almacenando más datos. También se pueden redimensionar para aprovechar mejor el espacio de almacenamiento.
- *Segmentos:* Es un grupo de extensiones utilizados para almacenar un tipo particular de datos. Existen 4 tipos de segmentos: datos, índices, rollback y temporales.

  - Segmentos de datos: Cada segmento de datos almacena los datos correspondientes a una tabla
  - Segmentos de índice: Cada segmento de índice mantiene los datos para un índice definido dentro de la base de datos.
  - Segmento de Rollback: un segmento de Rollback permite almacenar las acciones de una transacción que deben ser deshechas bajo ciertas circunstancias.
  - Segmentos Temporales: Los segmentos temporales se crean cuando se requiere de un espacio temporal para procesar una instrucción de SQL, y son destruidos una vez que haya culminado el procesamiento de la instrucción.

10. Arquitectura Oracle, http://www.infor.uva.es/oracledba/capitulo1.html (Acceso Febrero, 2006).

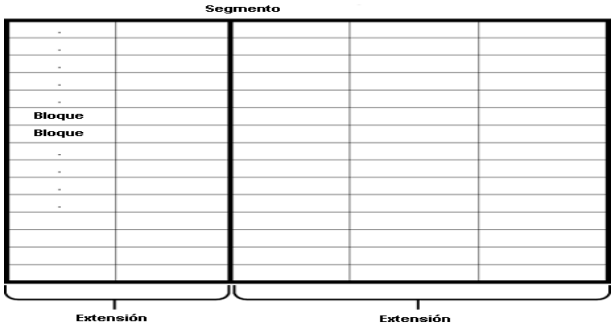


Figura 32. Relación entre bloques, extensiones y segmentos

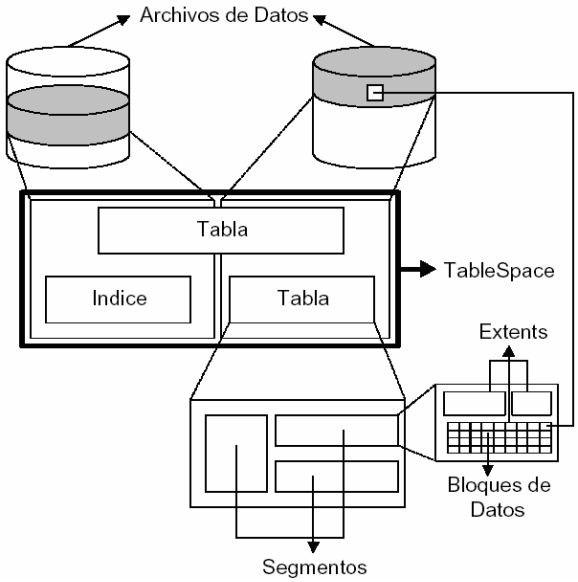


Figura 33. Estructura de datos manejadas por el RDBMS ORACLE

Los tablespaces pueden estar constituidos de varios Data Files. Al utilizar más de un Data File por tablespace puede distribuirse los datos sobre varios discos y balancear la carga de E/S, mejorando así el rendimiento del sistema.

Como parte del proceso de crear la base de datos, ORACLE automáticamente crear un tablespace llamado SYSTEM. Aunque sólo una base de datos pequeña puede ser almacenada en SYSTEM, es recomendable que se cree uno (o varios) tablespace(s) para los datos del usuario. El tablespace SYSTEM almacena los datos del diccionario.

El esquema de la base de datos

Un esquema es una colección de objetos lógicos, utilizados para organizar de manera más comprensible la información y conocidos como objetos del esquema.

10. Arquitectura Oracle, http://www.infor.uva.es/oracledba/capitulo1.html (Acceso Febrero, 2006).

CAPÍTULO III.

ADMINISTRACIÓN Y MONITOREO DE BASES DE DATOS ORACLE.

Como ya se menciono en el capitulo anterior Oracle es una Manejador de Base de Datos Relacional muy poderoso, el cual cuenta con herramientas muy utiles para facilitar la administracion y el monitoreo de una o varias bases de datos, estas herramientas son:

3.1 SQL PLUS

¿Qué es SQL*Plus?

SQL*Plus es una poderosa herramienta de Oracle que permite la interacción completa del usuario de la base de datos, para que este efectúe todas las operaciones cotidianas sobre ella, como son:

- Consultas a la base de datos.
- Operaciones de actualización sobre los datos.
- Definición de las estructuras para almacenamiento de los datos.

SQL*Plus implementa por completo el estándar ANSI de SQL para el DDL (Lenguaje de definición de datos) y el DML (Lenguaje de manipulación de datos). Además añade algunas características poderosas para las consultas y permite la elaboración de reportes de mediana complejidad.

SQL*Plus permite al usuario comunicarse con el servidor, para procesar comandos SQL y PL/SQL, tiene la flexibilidad de poder realizar inicio y parada (shutdown) de la base de datos.

Utilizando la herramienta Oracle SQL*PLUS

Inicio de Sesión con SQL*Plus

En la ventana inicial de conexión (figura 34) debemos ingresar el usuario y su contraseña.

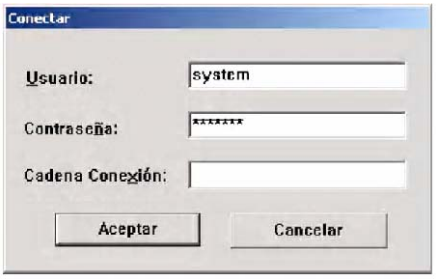


Figura 34. Conexion a la base de datos con SQLPLUS

11. Cesar Pérez, “Administración y Análisis de Bases de Datos Oracle 9i” (México, Alfaomega-Rama, 2002) 150 -249 pp.

Después de haber hecho la conexión con la base de datos, muestra la pantalla de bienvenida de SQL*Plus con la información del manejador:

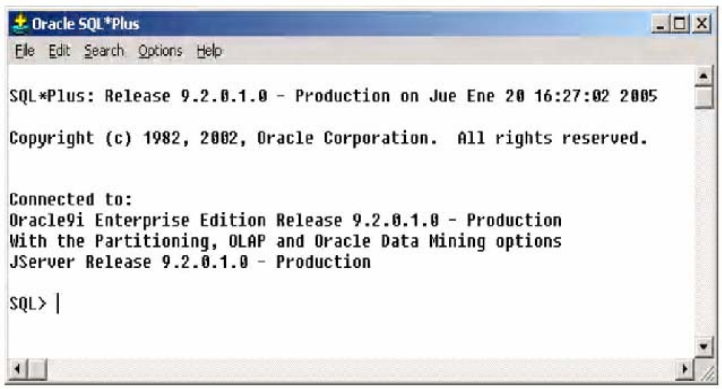


Figura 35. Ventana de bienvenida SQLPLUS

La base de datos de oracle incluye esquemas de ejemplo como SCOTT el cual es el nombre del esquema Oracle asociado al nombre de usuario Scott.

En estos momentos estamos listos para trabajar, por ejemplo si queremos conectarnos como scott, el comando es el siguiente:

```
SQL> connect scott/tiger
Conectado.

SQL>
```

Figura 36. Conexion al esquema SCOTT

Esquema de Ejemplo SCOTT

Este esquema esta formado por las tablas EMP, DEPT, BONUS y SALGRADE y están relacionadas de acuerdo a como se muestra en la Figura 37.

11. Cesar Pérez, “Administración y Análisis de Bases de Datos Oracle 9i” (México, Alfaomega-Rama, 2002) 150 -249 pp.

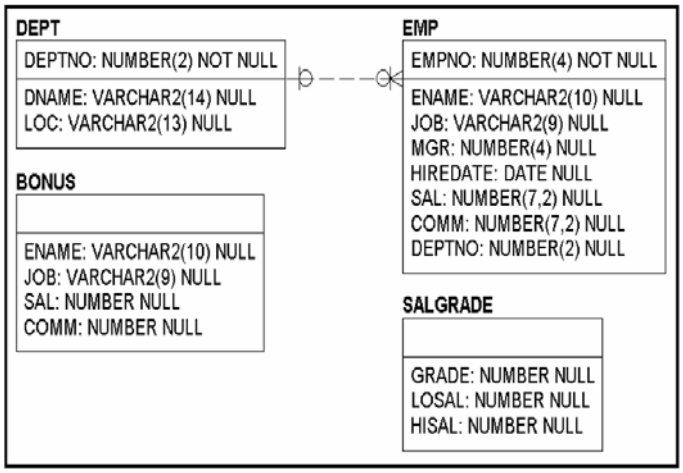


Figura 37. Esquema de ejemplo SCOTT

De igual forma que el esquema SCOTT, se encuentra el esquema HR el cual representa el área de Recursos Humanos y esta formado por las tablas JOBS, EMPLOYEES, DEPARTMENTS, JOB_HISTORY, REGIONS, LOCATIONS y COUNTRIES como se muestra en la Figura 38.

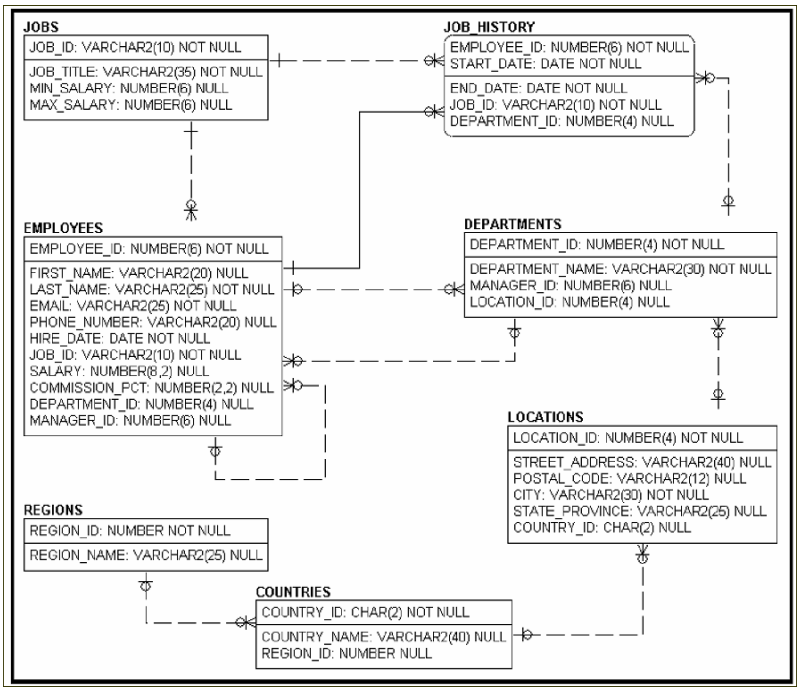


Figura 38. Esquema de Ejemplo HR

Ya que nos hemos conectado con cualquiera de estos dos usuarios podemos ingresar la siguiente consulta para poder ver con que usuario estamos conectados.

```
SQL> select user from dual;

USER
------------------------------
HR
```

Figura 39. Usuario conectado.

Como ya se ha mencionado con SQL*Plus podemos iniciar y parar una instancia de base de datos, crear y ejecutar consultas, modificar datos y escribir reportes personalizados.

A continuación se muestran algunos ejemplos de sentencias SQL que podemos realizar con esta herramienta.

- Consultar una tabla

```
SQL> connect scott/tiger
Connected.

SQL> describe emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

Figura 40. Estructura de una tabla

- Seleccionando los datos

```
SQL> conn hr/hr
Connected.

SQL> select * from jobs;

JOB_ID     JOB_TITLE                          MIN_SALARY MAX_SALARY
---------- ---------------------------------- ---------- ----------
AD_PRES    President                               20000      40000
AD_VP      Administration Vice President           15000      30000
AD_ASST    Administration Assistant                 3000       6000
FI_MGR     Finance Manager                          8200      16000
FI_ACCOUNT Accountant                               4200       9000
... ... ... ...
... ... ... ...
IT_PROG    Programmer                               4000      10000
MK_MAN     Marketing Manager                        9000      15000
MK_REP     Marketing Representative                 4000       9000
HR_REP     Human Resources Representative           4000       9000
PR_REP     Public Relations Representative          4500      10500

19 rows selected.
```

Figura 41. Cláusula Select.

- Tabla DUAL

```
SQL> select sysdate, user from dual;

SYSDATE  USER
-------- ------------------------------
22/01/05 HR

SQL> select 'Yo soy ' || user || ' Hoy es ' || sysdate
  2  from dual;

'YOSOY'||USER||'HOYES'||SYSDATE
--------------------------------------------------
Yo soy HR Hoy es 22/01/05
```

Figura 42. Uso de la tabla DUAL

- Funciones de fecha

```
SQL> alter session set nls_date_format='DD-Mon-YYYY HH24:MI:SS';

Session altered.

SQL> select current_date from dual;

CURRENT_DATE
--------------------
26-Ene-2005 15:16:15
```

Figura 43. Fecha del sistema.

- Funciones de conversión

```
SQL> select to_char(sysdate, 'Day, Month YYYY','NLS_DATE_LANGUAGE=English')
  2  from dual;

TO_CHAR(SYSDATE,'DAY,MONT
-------------------------
Wednesday, January   2005


SQL> select to_char(sysdate, 'Day, Month YYYY','NLS_DATE_LANGUAGE=Spanish')
  2  from dual;

TO_CHAR(SYSDATE,'DAY,MONTH
--------------------------
Miércoles, Enero      2005


SQL> Select to_char(sysdate, 'Day, DD "de" MONTH "de" YYYY')
  2  from dual;

TO_CHAR(SYSDATE,'DAY,DD"DE"MONTH"DE
-----------------------------------
Miércoles, 26 de ENERO      de 2005
```

Figura 44. Funciones de conversión

- Insertando registros

```
SQL> connect hr/hr
Connected.

SQL> insert into
  2  departments(department_id, department_name, manager_id, location_id)
  3  values(300, 'Departamento 300', 100, 1800);

1 row created.

SQL> commit;
Commit complete.

SQL> insert into employees (employee_id,
  2      first_name, last_name,
  3      email, phone_number,
  4      hire_date, job_id, salary,
  5      commission_pct, manager_id,
  6      department_id)
  7  values(250,
  8      'Gustavo', 'Coronel',
  9      'gcoronel@miempresa.com', '511.481.1070',
 10      sysdate, 'FI_MGR', 14000,
 11      NULL, 102, 100);

1 row created.

SQL> commit;
Commit complete.
```

Figura 45. Cláusula Insert.

- Modificando datos

```
SQL> update employees
  2  set salary = salary * 1.10;

109 rows updated.

SQL> Commit;
Commit complete.
```

Figura 46. Cláusula update.

- Eliminando registros

```
SQL> delete from copia_emp
  2  where department_id = 50;

44 rows deleted.


SQL> commit;
Commit complete.


SQL> delete from copia_emp
  2  where department_id = 50;

44 rows deleted.


SQL> commit;
Commit complete.

SQL> delete from copia_emp
  2  where department_id = 50;

44 rows deleted.


SQL> commit;
Commit complete.
```

Figura 47. Cláusula delete.

Creación de un esquema de base de datos.

- Modelo lógico

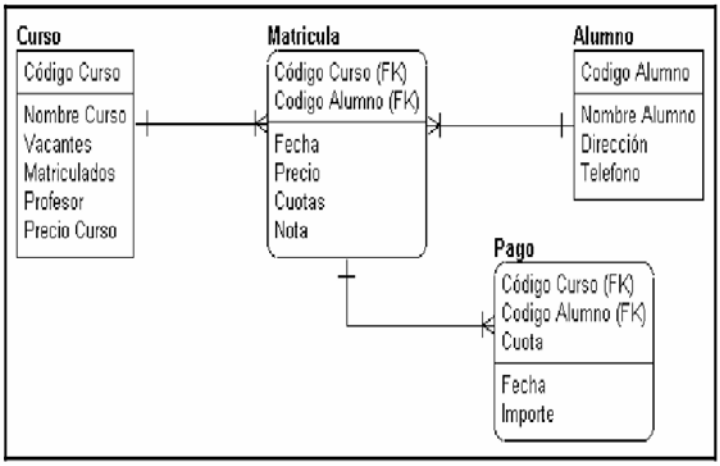


Figura 48. Modelo Lógico del esquema de la base de datos.

- Modelo físico

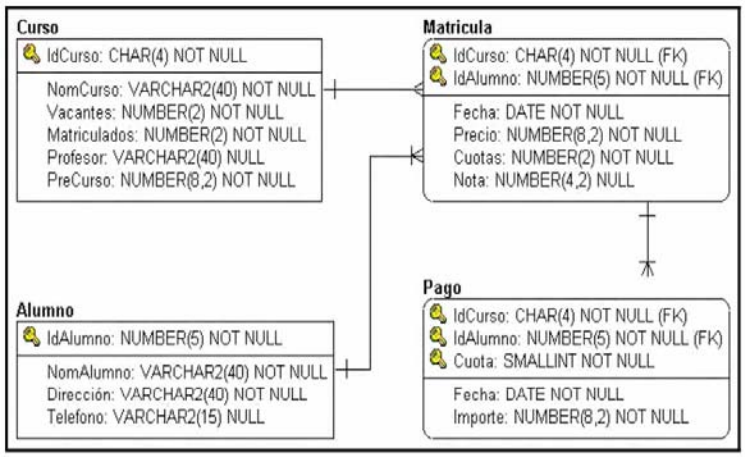


Figura 49. Modelo Físico del esquema de la base de datos

- Para la creación de este esquema primero debemos crear un usuario de la siguiente manera:

```
SQL> conn / as sysdba
Connected.


SQL> create user egcc
  2  identified by admin;

User created.
```

Figura 50. Cláusula Create user.

- Asignarle privilegios de connect y resource para que se pueda conectar y crear tablas.

```
SQL> grant connect, resource to egcc;

Grant succeeded.
```

Figura 51. Asignación de privilegios

- Creación de tablas

Tabla curso

```
SQL> connect egcc/admin
Connected.


SQL> CREATE TABLE Curso (
  2         IdCurso              CHAR(4) NOT NULL,
  3         NomCurso             VARCHAR2(40) NOT NULL,
  4         Vacantes             NUMBER(2) NOT NULL,
  5         Matriculados         NUMBER(2) NOT NULL,
  6         Profesor             VARCHAR2(40) NULL,
  7         PreCurso             NUMBER(8,2) NOT NULL
  8  );

Table created.
```

Figura 52. Cláusula Create Table.

- Crear llaves primarias

```
SQL> Alter Table Curso
  2      Add Constraint PK_Curso
  3      Primary Key ( IdCurso );

Table altered.
```

Figura 53. Asignación de llave primaria a la tabla curso.

- Crear Llaves foráneas

```
SQL> Alter table Matricula
  2      Add Constraint FK_Matricula_Curso
  3      Foreign Key ( IdCurso )
  4      References Curso;

Table altered.
```

Figura 54. Asignación de llave foránea a la tabla Matricula.

Estos son ejemplos de administración de bases de datos con el uso del Lenguaje Estructurado de Consultas (SQL) mediante la herramienta SQLPLUS de Oracle.

Este tipo de administración también se puede llevar acabo con la herramienta gráfica Enterprise Manager Console de Oracle como se muestra en el siguiente punto.

11. Cesar Pérez, “Administración y Análisis de Bases de Datos Oracle 9i” (México, Alfaomega-Rama, 2002) 150 -249 pp.

3.2 Enterprise Manager Console

El DBMS Oracle denomina servicios a las bases de datos. Un DBMS Oracle puede tener definidas múltiples bases de datos y, para cada una de ellas, todos los objetos necesarios, como:

- Usuarios. Usuarios del DBMS definidos en el propio DBMS que presentan una serie de privilegios concedidos por el usuario del sistema (**system**) o por otro usuario con suficientes privilegios.
- Espacios de tablas. Proporcionan una forma de organizar lógicamente las tablas. Hay que conceder permiso explícito para que los usuarios puedan acceder a ellas (con **CREATE USER** o **ALTER USER**, al indicar la cuota permitida sobre los espacios de tablas, bien sea el predeterminado (**DEFAULT**) u otro).

Los usuarios que crean objetos (tablas, vistas, índices) son los propietarios de éstos y pueden conceder acceso a otros usuarios.

3.2.1 A**dministración con Enterprise Manager Console**

Oracle Enterprise Manager es una consola para la administración del servidor. Mediante una interfaz gráfica se pueden administrar el esquema de las bases de datos, la seguridad, las sesiones, el almacenamiento y otros aspectos como los almacenes de datos y las bases de datos XML. Esta consola se ejecuta desde el acceso directo Enterprise Manager Console.

La consola se articula en torno a una jerarquía de objetos del servidor. En el nodo Bases de datos se encuentran todos los servicios que ofrece uno o más servidores, que se han definido previamente con la herramienta Net Configuration Assitant o manualmente en los ficheros .ora. Cada servicio o base de datos contiene, entre otros:

- El esquema que lo define. Este esquema está organizado por usuarios y, por cada usuario, los objetos que puede manejar (tablas, vistas, índices, etc.).
- Información de seguridad. Usuarios, roles y perfiles.

Con esta herramienta se pueden crear, borrar y modificar los mismos objetos y sus parámetros que con los lenguajes DDL y DCL. Las acciones de administración puntuales, como alterar una tabla, conceder permisos, se realizan de forma muy cómoda con esta herramienta.

12. Oracle, “Oracle Enterprise Manager”
(Oracle, 2001) 27 - 100 pp.

3.2.2 Autentificación y gestión de usuarios

Creación de Usuarios

Para crear un usuario se debe seleccionar, en primer lugar, la opción correspondiente en la aplicación, de la siguiente manera:

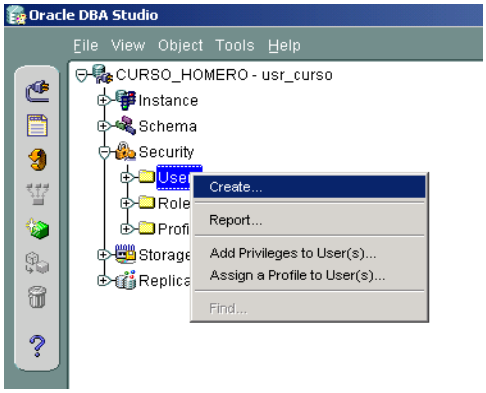


Figura 55. Menú flotante de creación de usuarios

Haciendo clic con el botón derecho una vez que estamos sobre la opción "Users" del administrador de seguridad (figura anterior), aparece la interfaz de creación de usuarios, que tiene las siguientes características y que se rellena con los datos que se muestran:

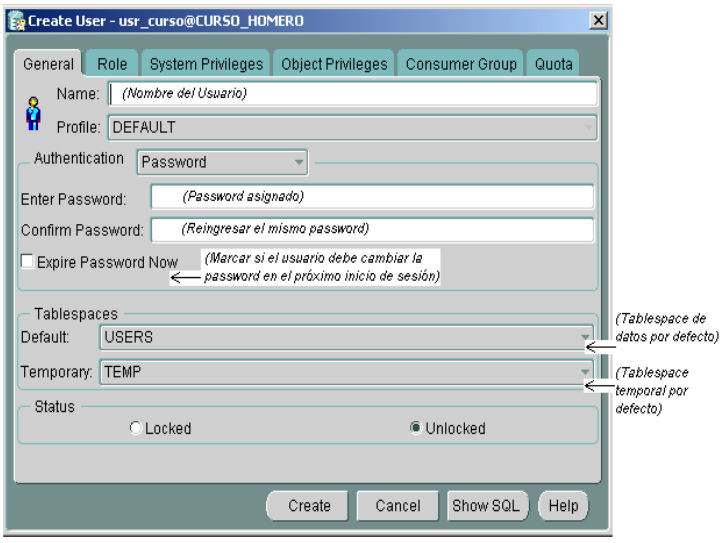


Figura 56. Ventana principal de creación de usuarios

Además, como se observa en la figura anterior, existen otras fichas que permiten asociar al usuario algún rol, privilegios sobre objetos comunes o del sistema y cuotas de espacio, entre otros.

Otras opciones que es necesario configurar la primera vez se refieren a brindar la capacidad al usuario de poder conectarse a una base de datos e iniciar una sesión por primera vez, dándosele también la oportunidad de crear objetos en su espacio o esquema, asignando un volumen máximo a ese espacio, para cada usuario que se crea.

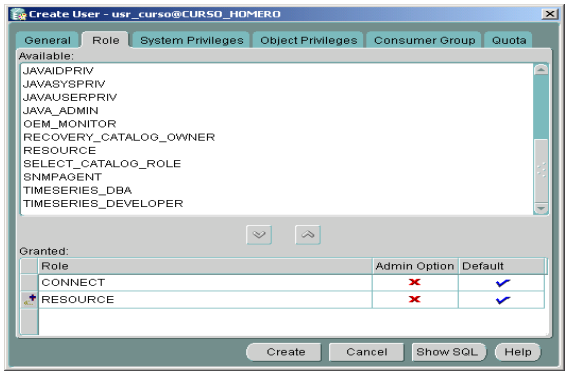


Figura 57. Roles concedidos al usuario

Rol Connect: Permite al usuario iniciar una sesión en la base de datos.

Rol Resource: Permite crear objetos, entre otros.

El símbolo  a la izquierda del rol resource significa que esa es una línea que se está agregando en la lista. En efecto, sólo se asigna por defecto el rol connect y nosotros debemos agregar el segundo cada vez para permitir al usuario crear objetos en su esquema.

Enseguida, para definir la cuota de espacio, tenemos que abrir la última pestaña de la ventana de creación de usuarios y empezar a asignar, tablespace por tablespace, el espacio definido para este usuario en particular. De esta forma podemos establecer las cuotas de cada usuario en cada uno de los espacios definidos en el sistema.

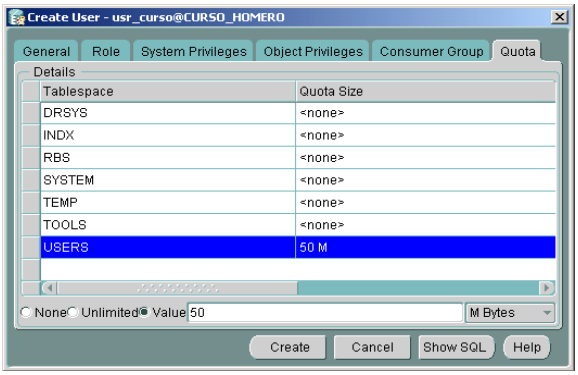


Figura 58. Cuota del usuario por tablespace

3.2.3 Gestión de espacio

Creación de Tablespaces

Para efectos prácticos vamos a suponer que los usuarios creados en este ejercicio deben estar asignados a un espacio de tablas diferente a los ya existentes. Por lo tanto, no nos sirve que tengan el tablespace users asignado por defecto.

Para crear un nuevo tablespace y asignarlo a los usuarios creados, se debe proceder de la siguiente manera:

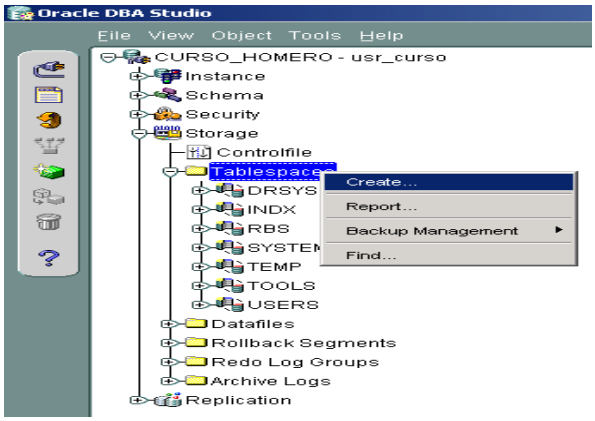


Figura 59. Como crear un tablespace

La interfaz principal de creación de los tablespaces aparece cuando seleccionamos la opción mostrada en la figura anterior y es la siguiente:

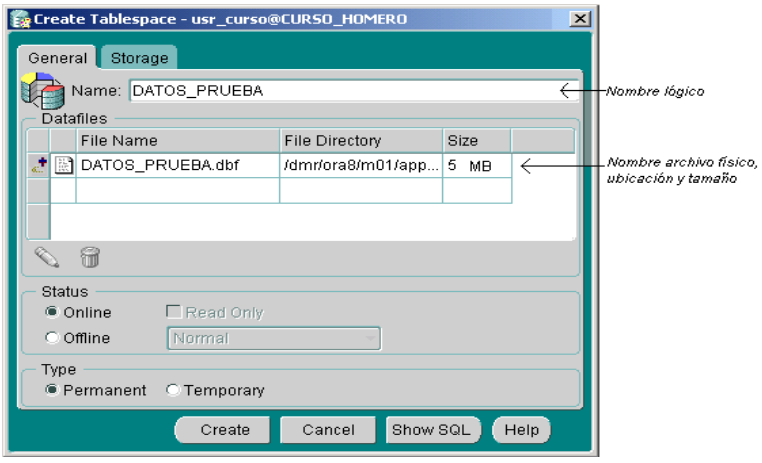


Figura 60. Detalle de la creación de un tablespace

En la figura anterior se observa la relación entre tablespace y datafile, éste último corresponde al archivo físico de extensión DBF que se muestra en la línea de detalle.

Para el ejemplo, el tablespace creado se llama datos_prueba.

Ahora, para asignar el espacio de tablas recién creado a nuestros usuarios, basta con editar sus características (botón derecho sobre el nombre del usuario) y asignar el nuevo tablespace a cada uno de ellos.

Posteriormente, con el fin de poder otorgarle al usuario la posibilidad de crear tablas en su esquema, no debemos olvidarnos de asignar una cuota de espacio a cada usuario dentro del tablespace.

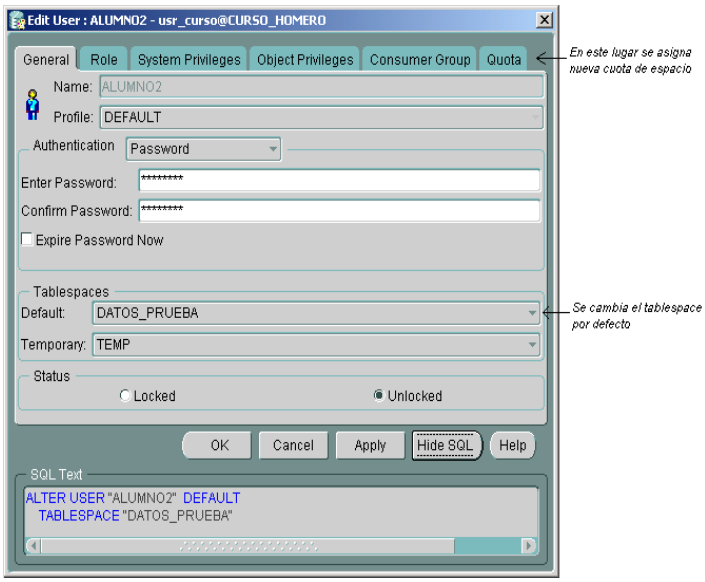


Figura 61. Modificación de datos de un usuario

Creación de Tablas

Enseguida, con el fin de conocer en la práctica cómo trabaja Oracle las extensiones de las tablas y aprender a monitorearlas para evitar que crezcan demasiado y puedan llegar a causar detenciones de la base de datos.

Un error muy común es que una tabla haya alcanzado el máximo posible de sus extensiones (valor indicado al crearla) y que por lo tanto no pueda seguir creciendo en tamaño, por lo que cada vez que se intente insertar datos o actualizarla incrementando su tamaño, aparezca un error de Oracle que nos lo impida.

Ejemplo:

Al crear una tabla, los parámetros que identifican los tamaños y cantidad de extensiones posibles para una tabla son los siguientes:

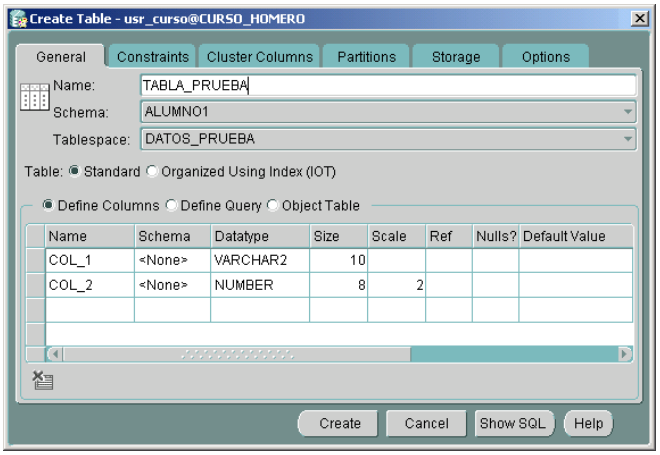


Figura 62. Creación de una tabla (paso 1)

Las opciones que le permiten a Oracle efectuar el manejo del espacio de almacenamiento se ingresan (considerando la figura anterior), en la pestaña "Storage":

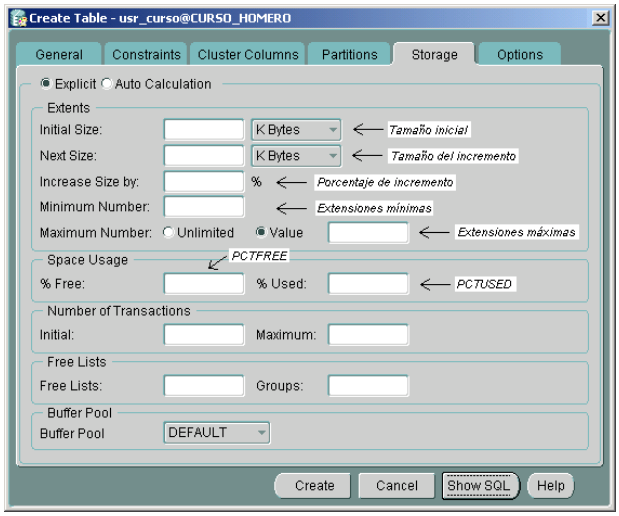


Figura 63. Creación de una tabla (paso 2)

Finalmente, introduciendo ciertos valores que deberán establecerse en rigor luego de un exhaustivo análisis del objeto que se está creando (porcentaje de volatilidad, crecimiento esperado, restricciones de tamaño en los discos, etc.), un dba

12. Oracle, “Oracle Enterprise Manager” (Oracle, 2001) 27 - 100 pp.

podría establecer que para esta tabla podrían aplicar ciertos valores, que para este ejemplo, se muestran a continuación, junto con la sentencia SQL que podría haberse escrito en lugar de utilizar la forma gráfica:

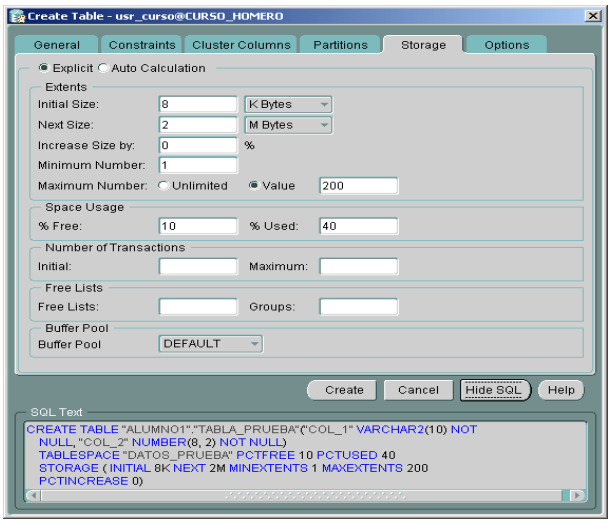


Figura 64. Sintaxis de creación de la tabla del ejemplo

3.3 Monitoreo

Monitoreo de Bases de Datos:

Para ayudar en la identificación y resolución de problemas, los DBA's deben ser capaces de revisar rápidamente las características más críticas en cada área de una base de datos que afecte al performance. Para esto debe realizar el monitoreo diario por horas de las bases de datos para chequear rendimientos y espacios requeridos para el correcto funcionamiento.

3.3.1 Oracle Performance Manager

Oracle Performance Manager es una herramienta de Interfaz Grafica de Usuario (GUI) para el monitero del performance que permite observar indicadores del performance de bases de datos en tiempo real el cual facilita el diagnóstico de problemas.

Los administradores del sistema y de las bases de datos pueden monitorear las estadísticas de performance incluyendo las bases de datos, servidores web, aplicaciones y el sistema como tal.

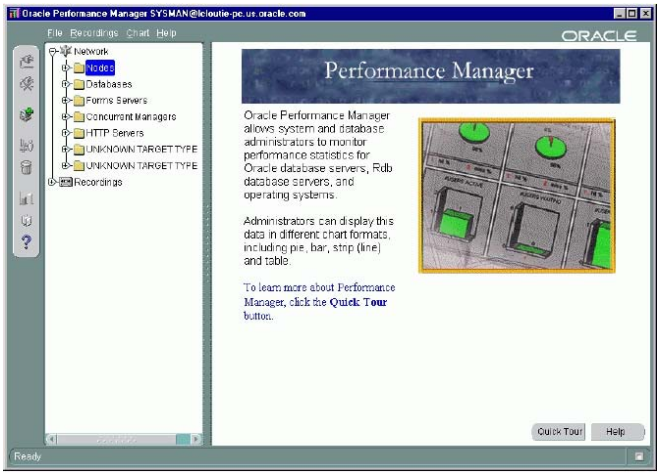


Figura 65. Consola Performance Manager

Una característica del Performance Manager es que el performance de los datos pueden ser grabados durante varias horas y después mostrarlos gráficamente para ayudar a identificar puntos en la actividad que negativamente puede impactar en el sistema. La siguiente figura muestra algunas áreas de tuning que se pueden monitorear con *Performance Manager*.

13. Oracle, "Database Tuning with the Oracle Tuning Pack" (Oracle, 2001) 232 - 295 pp.

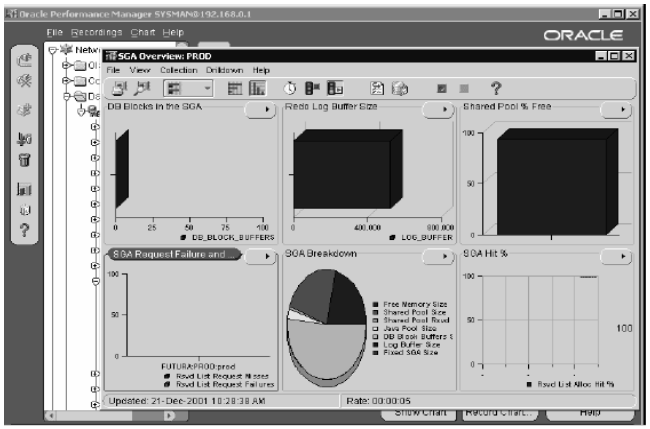


Figura 66. Graficas de Monitoreo con Performance Manager

3.3 Performance Overview

Performance Overview muestra una simple vista del performance de la base de datos, las estadísticas de funcionamiento para la memoria, el uso de CPU, la entrada/salida física y los procesos SQL.

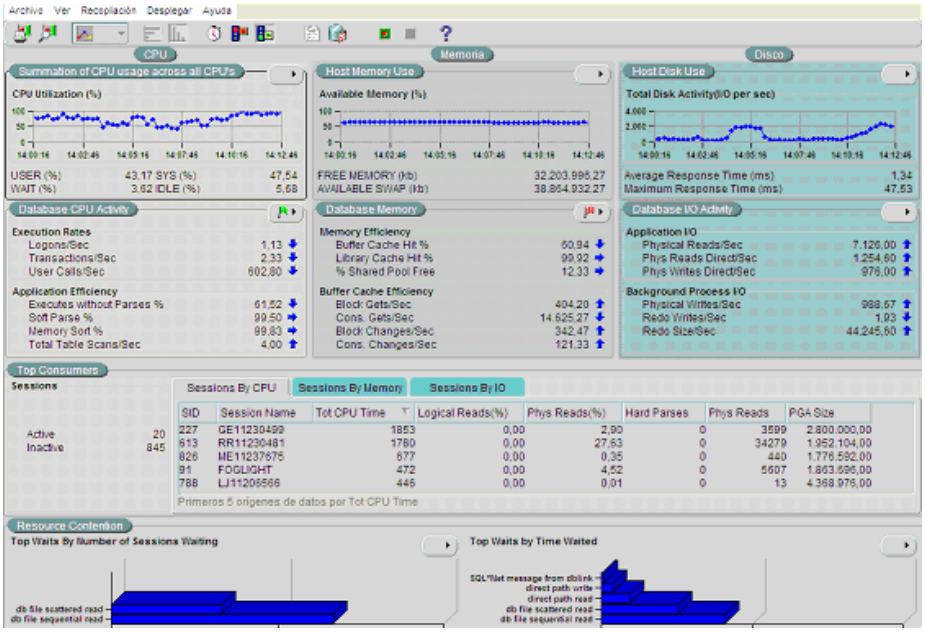


Figura 67. Consola Performance Overview

Estas graficas están organizadas en categorías de recurso: CPU, memoria y actividad de entrada - salida. Consumidores de recurso principales como top database sessions y resource contention.

13. Oracle, “Database Tuning with the Oracle Tuning Pack” (Oracle, 2001) 232 - 295 pp.

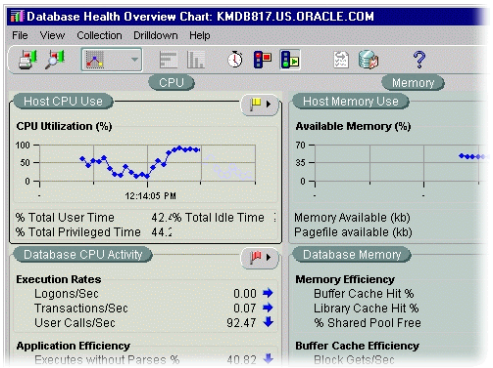


Figura 68. Grafica de uso de CPU con Performance Overview

El Performance Overview proporciona una metodología para poner la información de problemas de funcionamiento. Cada grafica contiene descripciones métricas de la acción de los usuarios las cuales ayudan a detectar la fuente de un problema.

3.3. TopSessions

TopSessions es una herramienta gráfica para identificar en tiempo real que usuarios consumen la mayor parte de recursos del servidor. Los recursos que pueden ser supervisados son el CPU, la memoria, lecturas a disco. y el volumen de transacción.

La figura muestra una muestra una pantalla de TopSessions.

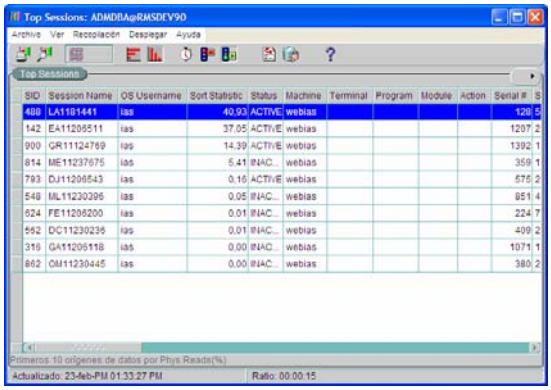


Figura 69. Pantalla TopSessions

Podemos ver los detalles de una actividad, de una sesión seleccionada, incluyendo el uso de recursos y el SQL que esta ejecutado. Una sesión puede ser investigada por la creación de *explain plans*, y la ejecución de SQL tuning directamente de la grafica de Sesión Details.

13. Oracle, “Database Tuning with the Oracle Tuning Pack” (Oracle, 2001) 232 - 295 pp.

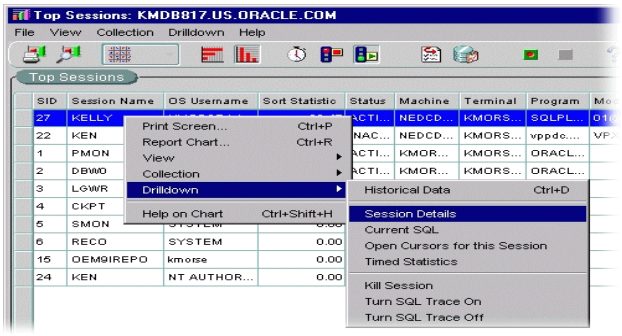


Figura 70. Detalles de sesión.

Esta grafica presenta una vista de los detalles de las sesiones de la base de datos e incluye las siguientes secciones:

- Session Identification Information
- Current State
- Session CPU Activity
- Session Memory Use
- Session I/O Activity
- Current SQL
- Top Wait Events

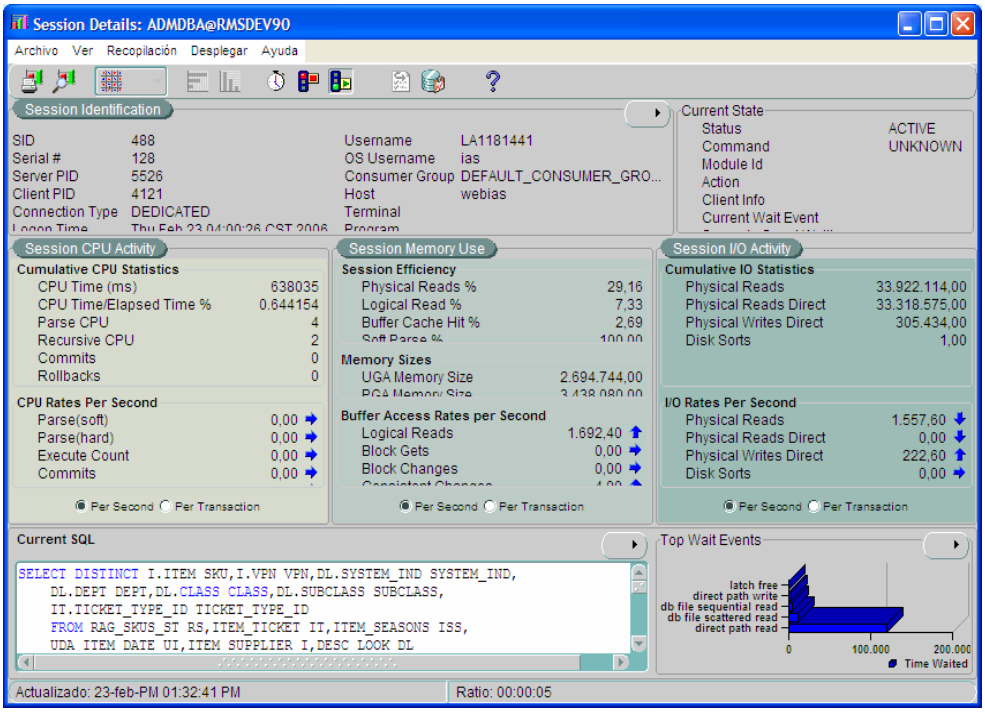


Figura 71. Pantalla General de Detalles de Sesión.

3.3.4 Top SQL

13. Oracle, “Database Tuning with the Oracle Tuning Pack” (Oracle, 2001) 232 - 295 pp.

Top SQL esta diseñado para ayudar a identificar fácilmente que declaraciones SQL consumen la mayor parte de recursos de sistema. La figura muestra una ejemplo de esta ventana.

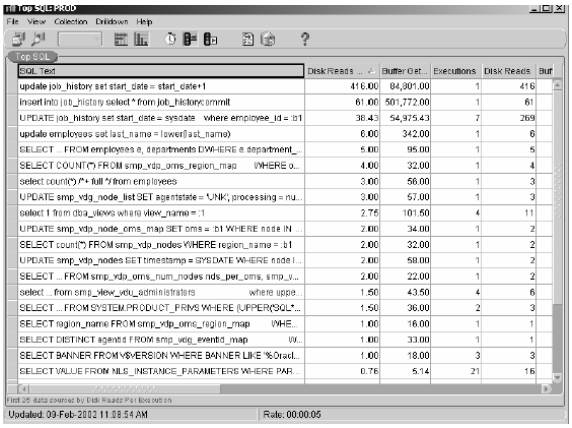


Figura 72. Pantalla Top SQL

Esta figura muestra los comandos SQL que consumen la mayor parte de los recursos del sistema. Top SQL también permite desplegar la información detallada del plan de ejecución de declaraciónes y las estadísticas sobre declaraciones de entrada - salida y el uso de CPU.

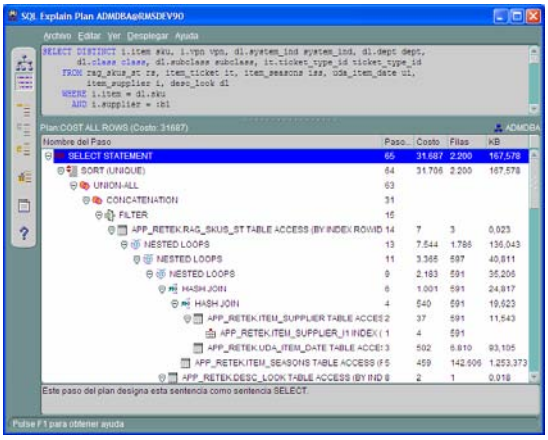


Figura 73. Sentencia de mayor consumo de recursos del sistema

Estas son algunas herramientas de Oracle, que nos ayudan a monitorear a los usuarios y su consumo de recursos del sistema, para detectar problemas y poderlos resolver de manera rápida y que el funcionamiento de la base de datos sea eficiente.

13. Oracle, “Database Tuning with the Oracle Tuning Pack” (Oracle, 2001) 232 - 295 pp.

Conclusiones.

La Universidad Nacional Autónoma de México, a través de la Dirección General de Servicios de Cómputo Académico, imparte el Diplomado de Administración de Base de Datos, con el objetivo de formar profesionales que cuenten con los elementos necesarios para realizar una adecuada y productiva administración de Sistemas Manejadores de Bases de Datos Relaciónales (RDBMS).

La gran diversidad, volumen e importancia tanto económica como estratégica, que tiene la información en cualquier organización hoy en día, ya sea ésta una institución pública o privada, implica el uso de sistemas manejadores de bases de datos para garantizar su seguridad, consistencia, integridad accesibilidad, entre otros factores.

Esto ha creado la inherente necesidad de contar con profesionales de las Tecnologías de la Información, que tengan los conocimientos y habilidades requeridas para administrar en forma eficaz y eficiente, tan valioso recurso.

La administración de bases de datos es un campo que siempre se presta a la inventiva, la imaginación y la mejora constante, reduce la duplicidad de datos y los integra de manera que puedan ser accesados por múltiples programas y usuarios.

Dado que las bases de datos son un recurso compartido, la administración debe proporcionar estándares, guías de acción, procedimientos de control y la documentación necesaria para garantizar que los usuarios trabajen en forma cooperativa y complementaria al procesar datos en la bases de datos es decir establecer las políticas de administración.

El principal objetivo de este trabajo, ha sido presentar una alternativa de administración mediante el uso de herramientas y en combinación con otro tipo de tecnologías lograr un acoplamiento que de cómo resultado un sistema funcional y eficiente para cualquier organización.

El diplomado de Base de Datos es una buena opción para introducirse al mundo de la administración de esta área, por que se exponen diferentes temas como: programación, sistemas operativos, seguridad, y otros referentes a las bases de datos, esto otorga una visión mas amplia de lo que involucra al área de sistemas en general, ya que puedes aportar opiniones, ayuda o soluciones a problemas que no competen solo a tu área de trabajo.

Además conocerás cuales son las diferencias entre los distintos manejadores de bases de datos que existen actualmente en el mercado y sobre que tipo de sistema operativo adaptarlo de acuerdo a los requerimientos de cada empresa, por que esto depende de el capital con que cuenta, los recursos físicos (hardware), el numero de bases, la cantidad de información que desea almacenar, las aplicaciones que utilizan y el sistema operativo que manejan.

De acuerdo a estos requerimientos, tú podrás determinar que manejador implementaras en una empresa si se te pide hacer este análisis o si ya cuentan con alguno en particular dar tu opinión sobre sí es óptimo o sí requieren actualizar su sistema por otro de mayor rendimiento.

Actualmente muchas empresas utilizan el manejador de bases de datos Oracle, por que es un sistema seguro, confiable, optimo, fácil de administrar, con herramientas graficas para la gestión de las bases de datos, monitoreo de datos, monitoreo de usuarios y monitoreo de CPU, además existe mucha información en libros e Internet con la que puedes aprender de forma individual desde tu casa, ya que puedes bajar de manera gratuita manuales de instalación, configuración, administración, monitoreo y muchos otros más; también puedes descargar el software del manejador en el sitio de Oracle sin ningún costo, ya sea para un sistema operativo Windows, Linux o Solaris, esto con fines prácticos. En comparación de otros manejadores como Sybase, Informix MySQL, Postgresql, que no ofrecen estas características o no cumplen con las necesidades de cada empresa por la cantidad de información que manejan o la seguridad de los datos con que cuentan o por que su uso esta limitado solo para algunos sistemas operativos o para alguna aplicación en particular. Por este motivo, decidí emplear en este trabajo la administración y el monitoreo de bases de datos con el RDBMS Oracle.

En mi opinión, como ya comente este curso solo introduce a la administración de las bases de datos, pero no te especializa como administrador en algún manejador en especial, debido a que el temario es muy ambicioso, ya que tratan de enseñar de todo un poco, tanto de bases de datos como programación y otros temas relacionados con bases de datos pero que para mi punto de vista eso no es útil, por que no aprendes un tema bien. A mi consideración creo que tienen que estructurar bien el temario y enfocarse de lleno al titulo de este diplomado “Administración de Bases de Datos”.

Bibliografía.

- BASES DE DATOS
  Desde Chen hasta Codd con ORACLE
  LUQUE, Irene; GÓMEZ-NIETO, Miguel; LÓPEZ, Enrique y CERRUELA, Gonzalo
  Coedición: Alfaomega-Rama

- DISEÑO DE BASES DE DATOS RELACIONALES
  De MIGUEL, Adoración; PIATTINI, Mario y MARCOS, Esperanza
  Coedición: Alfaomega-Rama

- FUNDAMENTOS DE BASES DE DATOS
  Abraham Silberschatz
  Henry F. Korth
  S. Sudarhan
  Mc Graw Hill

- FUNDAMENTOS Y MODELOS DE BASES DE DATOS
  De Miguel, Adoración y Piattini, Mario
  2ª Edición.
  Coedición: Alfaomega-Rama

- ORACLE ENTERPRISE MANAGER
  Getting Started with the Oracle Diagnostics Pack
  Release 9.0.1
  ORACLE

- ORACLE ENTERPRISE MANAGER
  Database Tuning with the Oracle Tuning Pack
  Release 9.0.1
  ORACLE

- ADMINISTRACIÓN Y ANÁLISIS DE BASES DE DATOS ORACLE 9i
  PÉREZ, C.
  Coedición: Alfaomega-Rama

Referencias Web

Bases de datos
http://www.programacion.com/bbdd/

Introducción a bases de datos
http://apuntes.rincondelvago.com/introduccion-a-bases-de-datos.html

Arquitectura Oracle
http://www.infor.uva.es/~jvegas/cursos/bd/oracledba/capitulo1.html

Administración y Optimización de Bases de Datos Oracle
http://www.redcientifica.com/oracle/

Documentación Oracle Enterprise Manager
http://www.oracle.com/technology/documentation/oem_arch_2x.html

Oracle Diagnostics Pack

http://www.alise.lv/ALISE/mrktinfo.nsf/diagnostics_pack_twp.pdf

Conceptos Oracle9*i* Database
http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96524/c11schem.htm

Administrador Oracle Database
http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96521/toc.htm