00321

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

52

FACULTAD DE CIENCIAS

ADMINISTRACION DE BASES DE DATOS: PROBLEMAS, CAUSAS Y SOLUCIONES

T E S I S

QUE PARA OBTENER EL TITULO DE

ACTUARIO

PRESENTA:

EDGAR RICARDO LOPEZ GALVAN

FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS:

DRA. AMPARO LOREZ GAONA

VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

DRA. MARÍA DE LOURDES ESTEVA PERALTA
Jefa de la División de Estudios Profesionales de la Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

Administración de Bases de Datos: problemas, causas y soluciones

realizado por Edgar Ricardo López Galván

con número de cuenta 8927061-3 , quién cubrió los créditos de la carrera de Actuaría

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	Dra. Amparo López Gaona
Propietario	Lic. Egar Arturo García Cárdenas
Propietario	Act. Hortensia Cano Granados
Suplente	Mat. Salvador López Mendoza
Suplente	M. en I.O. Maria de Luz Gasca Soto

Consejo Departamental de Matemáticas

M. en C. Jose Antonio Flores Díaz
Coordinador de la Carrera de Actuaría

AGRADECIMIENTOS

A DIOS

Por darme la oportunidad de vivir estos momentos

A MIS PADRES

Sabiendo del sacrifico de tiempo por tu parte papá, para que no me faltara lo indispensable y poder llegar hasta este momento, a ti mamá por tus noches de desvelo, por cuidarme y educarme, a ambos, gracias por su amor, y quiero que sepan que este logro también es suyo.

A LA UNAM

Por haberme formado profesional y espiritualmente, con una identidad única, que es la de sentirme orgulloso de ser su hijo.

A MI ESPOSA

Por tu amor y apoyo, que fueron, son y serán importantes para seguir hacia adelante.

A MIS HERMANOS

Alejandro, Itzel, Oscar y Herson, por contribuir con su cariño y amistad, a cada momento, esperando que esta perdure y trascienda en nuestras generaciones.

A LA DRA. AMPARO LÓPEZ GAONA

Con gran estimación, por su ayuda y su gran calidad humana.

A DON LINO

Por ese gran ejemplo de vida, y por todas esas historias contadas, que son historia de mis raíces y de mi país.

A MIS TIOS Y PRIMOS

En especial a mis tíos Domingo y Tomás, por ser un ejemplo para mi, y a todos los demás por su cariño y ayuda que me brindaron, gracias.

A CADA UNO DE MIS MAESTROS

Por que cada uno de ellos aportó en mi formación como estudiante y como persona.

A MIS AMIGOS

Saúl, Lalo y Mariana por sus años de amistad, a Janet por tu amistad y sinceridad, y ese afecto tan especial que te tengo.

"A TI, POR HABER DEJADO HUELLA EN MI CAMINO"

P. C

TABLA DE CONTENIDO

9 - D

INTRODUCCIÓN

Actualmente, y aún con el avance tecnológico, muchos son los aspectos que pueden llevar al fracaso o afectar el desarrollo de proyectos de software, o ciclo de vida del mismo, y muchos de ellos pueden estar relacionados con los datos, bases de datos o administración de bases de datos. Durante el periodo de 1997 al 2001 me desempeñe como administrador de bases de datos; durante este tiempo tuve la oportunidad de participar en varios proyectos de software, algunos de los cuales no tuvieron el éxito esperado.

Situándonos en torno a la administración de las bases de datos, algunos factores que llevaron al fracaso dichos proyectos fueron:

> Falta de experiencia o conocimientos en la administración de bases de datos, selección del sistema manejador de base de datos, exceso en carga de trabajo y falta de personal del grupo de administración de bases de datos, falta de estándar en el nombrado de objetos de la base de datos, falta de un marco de trabajo común como CMM (del inglés Capability Maturity Model), falta del uso de estándares como UML, convenciones para la codificación, etcétera, falta de políticas y procedimientos, así como la documentación de los mismos, estructuras organizacionales inadecuadas, etcétera.

La presente tesis tiene como objetivo, proponer sugerencias y soluciones para algunos de los problemas en torno a la administración de bases de datos, no pretende ser una "receta de cocina" para llevar una buena administración de bases de datos o crear un buen entorno para ello, pero sí evidenciar algunos elementos que a mi juicio son factores que pueden ayudar a llevar una buena administración de bases de datos y a crear un buen ambiente de dicha administración.

Para dicho propósito dividimos el trabajo de la siguiente manera:

Dentro del primer capítulo se proporciona el marco teórico necesario de las bases de datos, los sistemas manejadores de las bases de datos y el entorno que los rodea. Se hace un mayor énfasis sobre el modelo relacional propuesto por E.F.Codd, debido a que aún en la actualidad es el más usado, además se mencionan los objetivos, la arquitectura y el entorno de un sistema manejador de bases de datos, resaltando al administrador de la base de datos como el usuario identificable con la responsabilidad central de los datos.

El segundo capítulo se presenta con la finalidad de dar a conocer las principales funciones y tareas de la administración de las bases de datos, desglosándolas de la siguiente manera:

Diseño.- donde mencionamos las principales prácticas o técnicas de modelado, el modelo entidad-relación y la normalización, también hablamos un poco del proceso de transformación del modelo lógico al físico.

Integridad.- se explican los dos tipos de integridad de las bases de datos: integridad de la estructura e integridad semántica.

Seguridad.- mencionamos los distintos niveles de seguridad y las maneras de implementarlo mediante las características que podemos encontrar en la mayoría de los sistemas manejadores de bases de datos.

Disponibilidad.- se explica su importancia como el punto central de la administración de las bases de datos y su relación con el desempeño.

Respaldo y recuperación.- mencionamos los distintos tipos de respaldo y de recuperación, así como los aspectos que el administrador de bases de datos debe tomar en consideración para ello.

Desempeño.- mencionamos los elementos principales en la gestión del desempeño y los tipos de afinación de las bases de datos.

El tercer capítulo se describen situaciones y ejemplos que evidencian algunos problemas y causas en torno a la administración de las bases de datos, al mismo tiempo proponemos sugerencias para evitarlos y prevenirlos.

Por último, el cuarto capítulo es un ejemplo verídico de un proyecto que muestra los problemas que se mencionan en el tercer capítulo y otros problemas además de estos.

Capítulo 1

TEORÍA DE LAS BASES DE DATOS Y ENTORNO DEL DBMS

Este capítulo pretende dar el marco teórico necesario de las bases de datos, los sistemas manejadores de bases de datos y el entorno que los rodea. Se hace un mayor desarrollo sobre el modelo relacional propuesto por E.F. Codd en 1970, debido a que éste es aún en la actualidad el más usado, y a partir del cual se han desarrollado otras aplicaciones o tipos de bases de datos como son: objeto-relacional, dataware house, temporales, espaciales, multimedia, etcétera., también se menciona brevemente los modelos jerárquico y red.

1.1 ¿Qué es una base de datos?

La Real Academia Española define dato como: "Antecedente necesario para llegar al conocimiento exacto de una cosa o para deducir las consecuencias legítimas de un hecho."

El diccionario Webster: "Cosa conocida o asumida, de hechos o figuras a partir de las cuales se puede inferir una conclusión."

American National Stantards Institute (ANSI): "Una representación de hechos, conceptos o instrucciones de manera situable para la comunicación, interpretación o procesamiento por los humanos o por recursos automáticos."

Diremos entonces, que los datos consisten en símbolos escritos o almacenados en algún medio de escritura. Los símbolos representan ciertas cosas, ideas o valores que transmiten información de un contexto en particular. En términos de estructura, los datos consisten en valores de atributos de ciertas entidades.

Definición: [1]Una base de datos es una colección de datos persistentes que son usados por un sistema de aplicaciones de una empresa.

---

[1] C.J. Date, An introduction to Database Systems, Addison-Wesley, sèptima edicion, 2001, pág. 26

El término "empresa" es usado en esta definición de manera genérica, una empresa puede ser una persona (con una base de datos pequeña) o una corporación completa (con una base de datos grande y compartida) o cualquier otra cosa entre ambas. Algunos ejemplos son: una compañía manufacturera, un banco, un hospital, una universidad, un departamento de gobierno, etcétera.

Una colección de archivos de datos no califica automáticamente como una base de datos, para calificar los archivos deben ser:

- Integrados: Incorporando datos no redundantes.
- Interrelacionados: Ligados para proporcionar de manera completa y consistente información acerca de la empresa.
- Información orientada: Enfocado en las cosas que le interesan a la empresa.
- Independientes: Existiendo de manera separada de los procesos que ellos mismos soportan.

Otra definición para una base de datos (en adelante BD) es: [2]Una colección de elementos de datos interrelacionados que pueden procesarse por uno o más sistemas de aplicación.

1.2 Modelos de datos

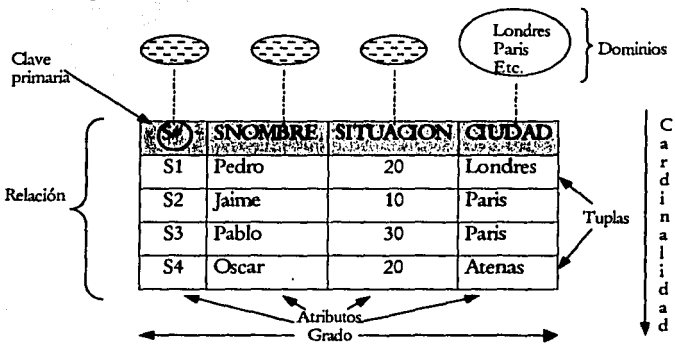
Para describir la estructura de una BD es necesario definir los modelos de datos, esto es, una colección de herramientas conceptuales para describir datos, relaciones entre ellos y restricciones de consistencia, hablaremos principalmente de los modelos de registros y en especial del modelo relacional.

1.2.1 Modelo relacional

C.J. Date desglosa en tres ámbitos la definición "Modelo Relacional":

I. Estructura de Datos
II. Integridad de los Datos
III. Manipulación de Datos

Cada uno de los tres ámbitos tiene sus propios términos, algunos de los más importantes se muestran en la Figura 1.1



S#	SNOMBRE	SITUACION	CIUDAD
S1	Pedro	20	Londres
S2	Jaime	10	Paris
S3	Pablo	30	Paris
S4	Oscar	20	Atenas

Figura 1.1 Elementos más importantes del Modelo Relacional

I. Estructura de Datos

La estructura de datos está caracterizada por el concepto de relación, para precisar este concepto se definen los siguientes términos que también forman parte de la definición de la sección estructural del modelo relacional:

Dominio: Es el conjunto de todos los valores posibles que puede tomar un atributo. Una característica inicial que se exige a los valores del dominio es la atomicidad, en el sentido de que no exista una descomposición de los valores del dominio que aporte algún significado.

Atributo: Es la aparición de un dominio en una *relación*. Se corresponde con lo que habitualmente llamamos campo o columna.

Tupla: Es cada una de las filas o registros de la relación.

Valores del Dominio: Es la aparición de uno de los posibles valores que constituyen el dominio. Su localización viene determinada por la intersección de una tupla y un atributo.

Relación o Tabla. Es el elemento principal del modelo. Una relación R, sobre una colección de dominios D_1, D_2, ..., D_n, consta de cabecera y cuerpo.

La *cabecera* consiste en un conjunto fijo de parejas atributo-dominio,

$$\{(A_1:D_1), (A_2:D_2),\ldots, (A_n,D_n)\}$$

donde cada atributo se corresponde exactamente con el dominio subyacente D_j $j=1,2,\ldots,n$

El *cuerpo* consta de un conjunto de tuplas, donde cada tupla consiste en un conjunto de parejas atributo-valor,

$$\{(A_1:V_{i1}), (A_2:V_{i2}),\ldots, (A_n,V_{in})\}$$

con i= 1,2,....m, siendo m el número de tuplas que contiene el conjunto. Para cada atributo Aj existe un par atributo-valor (Aj : Vij) . Para un par atributo-valor dado (Aj:Vij) es un valor concreto perteneciente al dominio Dj asociado al atributo Aj (que habíamos denominado valor del dominio). Los valores m y n se denominan *cardinalidad y grado* de la relación R respectivamente.

Clave primaria o llave primaria: Es el conjunto de atributos que identifican de manera única cada tupla de una relación. Pueden existir varios de estos conjuntos para una relación dada, pero solamente se seleccionará uno de estos como clave primaria. Se utilizará clave o llave de manera indistinta.

Propiedades de las Relaciones.

1. No hay tuplas duplicadas. Siempre deberá existir una clave primaria.
2. Los atributos no están ordenados. Esta propiedad proviene del hecho de que la cabecera de una relación es un conjunto matemático.
3. Las tuplas, no se encuentran ordenadas. Esta propiedad también se apoya en que la definición del cuerpo de una relación se corresponde con un conjunto matemático.
4. Los valores de los atributos son atómicos. Esto significa que una descomposición de valores del dominio no aporta algún significado.

Integridad de los Datos

Una base de datos consiste en una configuración de datos que pretende representar el mundo real. Sin embargo algunos valores de configuración no representan el mundo real, ejemplo: en una relación P se tiene para el atributo peso; peso = -5, dato que no tiene un significado, por que el peso debe ser una cantidad mayor o igual a cero, entonces es necesario incluir ciertas reglas de integridad, para ello se definen los siguientes conceptos:

- *Clave*: Es un conjunto no-vacío de atributos que identifican de manera única a cada tupla.
- *Clave externa*: Es un conjunto de atributos de una relación S, tal que dicho conjunto de atributos es clave en otra relación R.

Ahora definimos las reglas de integridad:

- Regla de Identidad: Ningún componente de la clave primaria de una relación puede aceptar un valor nulo.
- Regla Referencial. La base de datos no puede contener valores para la clave externa que no estén en correspondencia con los adoptados por la clave primaria a la que hacen referencia.

Manipulación de Datos.

La manipulación de datos se realiza principalmente a través de dos lenguajes: El álgebra relacional y el cálculo relacional, hablaremos por cuestión practica únicamente del álgebra relacional.

Álgebra Relacional.

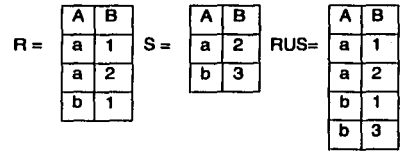
El álgebra relacional consiste en una colección de operadores de alto nivel que opera sobre las relaciones, cada operador toma una o dos relaciones de entrada y produce una nueva relación, se divide en dos tipos de operadores:

- Operadores tradicionales sobre conjuntos.
- Operadores especiales.

Operadores tradicionales sobre conjuntos:

Unión: Notación RUS

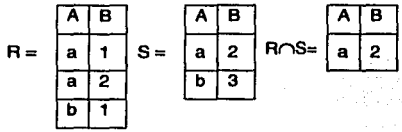
- Es la relación que contiene las tuplas de R además de las tuplas de S.



- Este operador debe asegurar la compatibilidad de los operadores, lo que implica que:
  - R y S tengan el mismo grado.
  - Los atributos de R y S tengan el mismo nombre.
  - El dominio de atributo-i de R es el mismo que el dominio del atributo-i en S, $\forall i$.

Intersección: Notación R$\cap$S.

- Es la relación con las tuplas en R y S también.
- Operación válida entre relaciones compatibles.



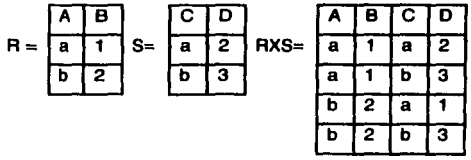
Diferencia: Notación R-S

- Crea una relación con las tuplas que están en R pero no en S.
- Operación válida entre relaciones compatibles.

Producto Cartesiano: RXS

- Permite combinar cualquier par de relaciones.

  $R \times S = \{tq \mid t \in R \text{ y } q \in S\}$



Operaciones especiales:

Selección: Este operador extrae de una relación R, un subconjunto de tuplas que cumplen una condición. Esta puede ser atómica o compuesta, es decir, que esta formada por una sola comparación o por varias. Lo denotaremos como:

$\sigma_x(R)$

Proyección: Produce un subconjunto de los atributos de una relación R. Lo denotaremos como:

$\Pi_{atributo1, atributo2, \ldots, atributo}(R)$

Reunión: Mediante este operador, a partir de dos relaciones R y S, se construye una compuesta de todas las posibles combinaciones de pares de tuplas concatenadas, pertenecientes una a la relación R y la otra a la relación S, tales que la s tuplas que componen el par satisfacen una condición común. En el caso de que la condición sea la igualdad "=", se hablará de equi-reunión. Se denotará como:

$R \bowtie S$

1.2.2 Otros modelos

Los datos en el modelo de red se representan mediante colecciones de registros y las relaciones entre los datos se representan mediante enlaces, los cuales pueden verse como apuntadores. La Figura 1.2 ilustra un ejemplo del modelo de red.

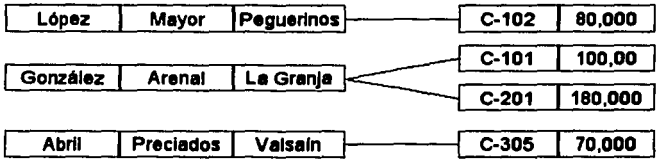


Figura 1.2 Un modelo de red

El modelo jerárquico es similar al modelo de red, los datos y las relaciones se representan mediante registros y enlaces. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles. La Figura 1.3 muestra un diagrama que ejemplifica el modelo jerárquico.

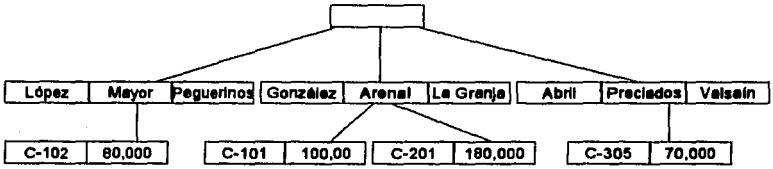


Figura 1.3 Modelo jerárquico

1.3 Sistema manejador de bases de datos y su entorno

[2]Un sistema de bases de datos es básicamente un sistema computarizado cuyo propósito general es mantener la información y hacer que ésta esté disponible cuando se solicite.
Un Sistema Manejador de Bases de Datos (DBMS del inglés DataBase Management System) es una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos.

1.3.1 Objetivos de un DBMS

Considérese parte de una empresa bancaria que guarda la información sobre todos los clientes y cuentas en archivos del sistema. Además, el sistema tiene diversos programas de aplicación que permiten al usuario manipular los archivos, como hacer abonos, añadir cuentas, consultar el saldo, etcétera. Según surge la necesidad se añaden nuevos programas de aplicación al sistema. El típico sistema de procesamiento de archivos descrito, está apoyado por un sistema convencional. Este sistema tiene un número de desventajas importante, como son:

[2] C. J. Date. Ob. cit. pág 47

1.- Redundancia e inconsistencia de los datos. Puesto que los archivos y los programas de aplicación son creados por distintos programas durante un periodo largo de tiempo, es probable que tengan distintos formatos, y pueden estar duplicados en varios sitios. Por ejemplo, la dirección de un cliente puede aparecer en más de un archivo. Esta redundancia aumenta los costes de almacenamiento y acceso, y puede llevar a la inconsistencia de los datos, esto es, las diversas copias de los mismos datos no concuerdan entre sí.

2.- Dificultad para tener acceso a los datos. Supóngase que se necesita averiguar los nombres de los clientes que viven en una ciudad. Puesto que esta solicitud no fue prevista, no hay ningún programa que la satisfaga. Tenemos dos opciones, coger la lista de todos los clientes y extraer la información manualmente, o escribir el programa de aplicación necesario. Obviamente ninguna de las dos opciones es satisfactoria. Lo que se trata de probar es que los entornos convencionales de procesamiento de archivos no permiten recuperar los datos necesarios de una forma conveniente y eficiente.

3.- Aislamiento de los datos. Puesto que los datos están repartidos en varios archivos, y estos pueden tener diferentes formatos, es difícil escribir nuevos programas para obtener los datos apropiados.

4.- Anomalías del acceso concurrente. Para mejorar el funcionamiento del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que los usuarios actualicen los datos simultáneamente. En un entorno así, las actualizaciones concurrentes pueden dar por resultados datos inconsistentes.

5.- Problemas de seguridad. No todos los usuarios del sistema de BD pueden tener acceso a todos los datos. Por ejemplo, el personal de nóminas no necesita acceder a la información sobre la dirección de los clientes, puesto que los programas de aplicación se añaden al sistema de una forma precisa, es difícil implantar tales restricciones de seguridad.

6.- Problemas de integridad. Los valores de datos almacenados en la BD deben satisfacer ciertos tipos de restricciones de consistencia. Por ejemplo, el saldo de una cuenta no debe caer por debajo de una cantidad prefijada. Estas restricciones se hacen cumplir añadiendo códigos apropiados en los programas. Sin embargo,

cuando se añaden restricciones nuevas, es difícil cambiar los programas para hacerlas cumplir.

Evitar estas dificultades, entre otras, son los objetivos que se plantea todo sistema manejador de bases de datos.

1.3.2 Arquitectura

Ahora mencionamos una arquitectura para un sistema de bases de datos, que sirve como un marco de referencia de los sistemas manejadores de bases de datos. Cabe aclarar que esta arquitectura no necesariamente sostiene o debe sostener a cada sistema de bases de datos, aunque dicha arquitectura parece idónea para la mayoría de los sistemas de bases de datos. La arquitectura ANSI/SPARC está dividida en tres niveles: interno, conceptual y externo como se muestra en la Figura 1.4.

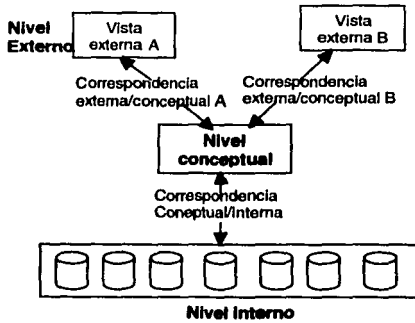


Figura 1.4 Arquitectura ANSI/SPARC

Nivel interno: es el nivel más bajo de abstracción e indica como serán almacenados físicamente los datos, esto se refiere a aspectos físicos de registros como paginas y bloques, la vista interna no sólo define varios tipos de registros almacenados, también define la existencia de los índices, cómo están almacenados los campos, que secuencia física tienen los registros almacenados, etcétera.

Nivel conceptual: describe que datos se almacenan en la base de datos y que relaciones existen entre estos datos. La base de datos completada se describe así en términos de un número pequeño de estructuras relativamente simples. Aunque la implementación de estructuras simples en el nivel conceptual puede involucrar estructuras muy complejas en el nivel interno los usuarios de este nivel no tienen que preocuparse de dicha complejidad.

Nivel externo: es el nivel de usuario individual, donde un usuario como mencionamos un poco más adelante, puede ser un programador, usuario final o el administrador de la base d datos, en este nivel y debido al gran tamaño de la base de datos, es muy común que el usuario sólo requiera de una porción de la base de datos.

1.3.3 Entorno

Como anteriormente mencionamos, un sistema de base de datos es un sistema computarizado que tiene como propósito mantener la información de interés y permitir que este disponible cuando sea requerida, donde la información de interés puede ser cualquiera que sea considerada de importancia para un individuo o para la organización entera. El entorno de un sistema de bases de datos; involucra cuatro elementos muy importantes: datos, usuarios, hardware y software.

Datos

Ningún sistema puede existir sin la presencia de los datos, los hechos históricos sobre los que se fundamentan las necesidades de información y de procesamiento de una empresa. Sin embargo, el factor esencial a considerar es que los datos que conforman una base de datos tienen que ser cuidadosa y lógicamente estructurados. Las funciones del negocio deben analizarse, los elementos de los datos y las interrelaciones deben identificarse y definirse, y estas definiciones deben almacenarse de manera precisa en el diccionario de datos. Un diccionario de datos es un subsistema que guarda la definición de todos los datos y relaciones de la base de datos, además de todas las otras estructuras. La información en el diccionario de datos es llamada metadatos.

Usuarios

Consideraremos tres tipos de usuarios:

- Programador de aplicaciones: este usuario es responsable de escribir programas que hacen uso de los datos y que resuelven desde problemas particulares hasta globales de la empresa, estos programas son hechos típicamente en lenguajes de programación orientados al uso de las bases de datos. Estos programas actúan sobre los datos de todas las maneras usuales, consultando información existente, insertando, modificando o borrando información o estructuras, todas estas funciones soportadas adecuadamente por un sistema manejador de bases de datos.
- Usuario final: este usuario interactúa con las aplicaciones o programas desarrollados por el programador, o puede hacer uso de una interfaz que existe como parte integral del DBMS.
- **Administrador de la Base de Datos: o DBA del inglés "Database Administrator": es un usuario o grupo identificable que tiene la responsabilidad central de los datos.** La Figura 1.5 muestra el entorno de una base de datos, e identifica al DBA.

Software

Un sistema de base de datos incluye dos tipos de software.

- Software Manejador de la Base de Datos de propósito general, usualmente llamado Sistema Manejador de la Base de Datos (DBMS).
- Software de Aplicación que usa al DBMS para facilitar y manipular la base de datos que contiene la información, la empresa u organización.

El sistema manejador de la base de datos es similar a un sistema operativo o a un compilador, este provee un cierto número de servicios a los diferentes usuarios de los datos contenidos en la base de datos, algunos de los servicios que podemos considerar son los siguientes:

- Una definición centralizada de los datos y control de los datos mediante un diccionario de datos.
- Mecanismos de Integridad y Seguridad de los datos.

- Acceso concurrente a los datos por múltiples usuarios.
- Consulta y Manipulación de Datos así como capacidad de elaborar reportes.
- Capacidad de desarrollar sistemas o aplicaciones por un programador.

El software de aplicación es desarrollado o hecho para resolver problemas específicos de la empresa, dicho software esta hecho en algún lenguaje de programación que permite el uso de los datos para generar reportes o documentos con información de la empresa. La Figura 1.5 muestra la integración del sistema manejador de base de datos, los niveles de la arquitectura y al administrador de la base de datos que es el único que interactúa con tres niveles.

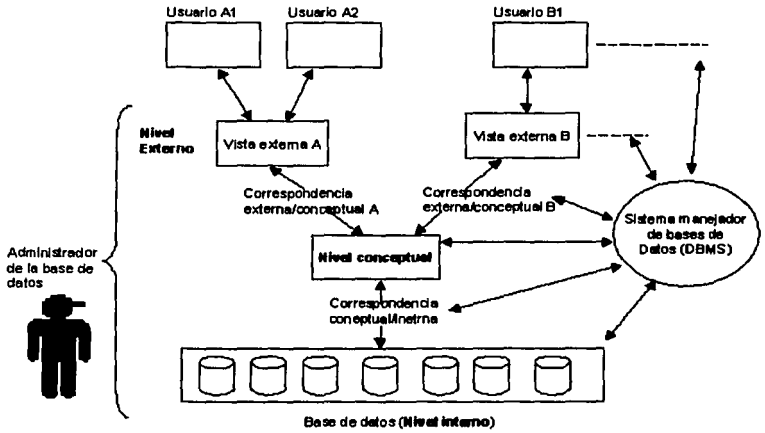
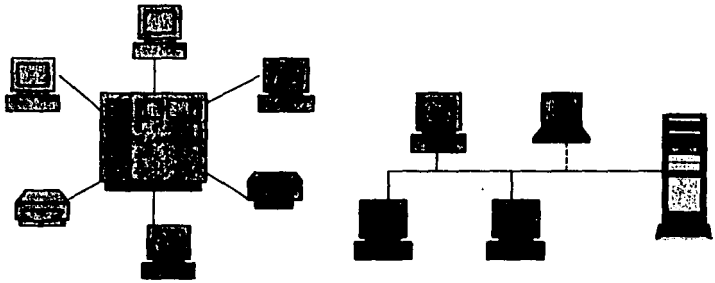


Figura 1.5 Entorno de una BD

Hardware

El hardware es el conjunto de dispositivos físicos en que reside la base de datos esto es: una o más computadoras, discos de almacenamiento, dispositivos de entrada y salida, impresoras, unidades de cinta magnética y otros equipos auxiliares.

En otras palabras la o las computadoras usadas para procesar los datos de la base de datos pudieran ser mainframes, minicomputadoras o microcomputadoras. El mainframe y las minicomputadoras se han utilizado tradicionalmente de forma autónoma para soportar el acceso de varios usuarios a una base de datos común, las computadoras personales se han utilizado frecuentemente con bases de datos autónomas controladas y manipuladas por un usuario único, no obstante, también pueden trabajar bajo una arquitectura cliente/servidor, garantizando el acceso a una base de datos común. Las unidades de disco constituyen el mecanismo de almacenamiento, las computadoras personales, las video-terminales y las impresoras se utilizan para introducir y recuperar información de las bases de datos, las unidades de cinta garantizan un respaldo barato y rápido de los datos que están almacenados en las unidades de disco y la infraestructura de conectividad para el caso en que las computadoras estén conectadas a una red, permitiendo así el uso de una base de datos común. La Figura 1.6 son ejemplos de lo antes mencionado.



(a) (b)

Figura 1.6 Algunos elementos de hardware

El éxito de los sistemas de los sistemas de bases de datos ha dependido fuertemente de los adelantos en la tecnología de hardware, para mantener y controlar la enorme cantidad de datos que en la actualidad se manejan, se requiere de una memoria y de un espacio de almacenamiento muy grande, además se necesitan computadoras, redes y periféricos rápidos para ejecutar un gran número de operaciones y un gran número de accesos,

afortunadamente el hardware es cada vez más potente y más barato, por lo que en la actualidad la mayoría de las empresas cuentan con sistemas manejadores de bases de datos.

Relación entre los componentes.

La Figura 1.7 resume la relación entre los cuatro elementos de un sistema de base de datos. Los especialistas en computación (programadores y administrador de la base de datos) en consulta a los usuarios finales identifican las necesidades de información y diseñan las estructuras para responder a estas necesidades, entonces, se especifican dichas estructuras en el DBMS mediante el diccionario de datos, el programador desarrolla las aplicaciones por las cuales el usuario final introduce los datos a la base de datos siguiendo procedimientos específicos. Los datos introducidos se almacenan en dispositivos de hardware tales como disco y cintas. De este modo puede verse que en un sistema adecuadamente diseñado y en funcionamiento los cuatro componentes: hardware, software, datos y personas, conforman un sistema único.

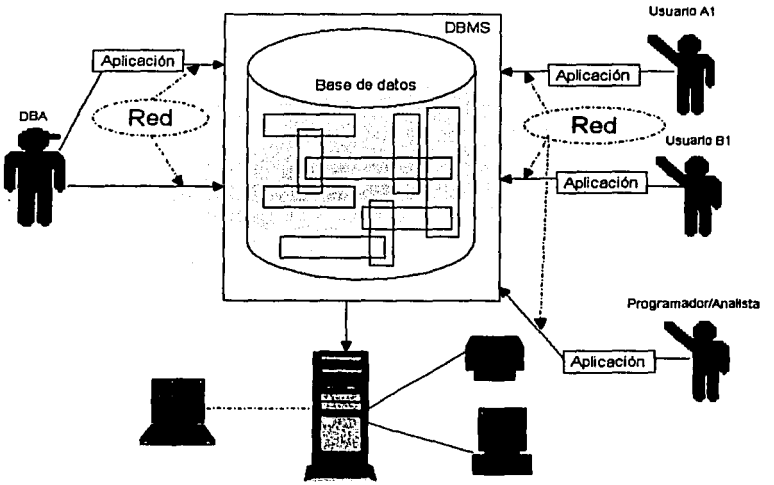


Figura 1.7 Entorno de un sistema manejador de bases de datos

Capítulo 2

FUNCIONES EN LA ADMINISTRACIÓN DE BASES DE DATOS

Los sistemas de información se consideran cada vez más como recursos que requieren de una buena gestión o administración, así como también de buenas características técnicas, este capítulo se presenta con la finalidad de dar a conocer las principales funciones de la Administración de Bases de Datos y las tareas que se deben llevar a cabo en cada una de ellas.

En pequeñas compañías u organizaciones una persona puede llevar a cabo todas las responsabilidades del administrador de la base de datos, aunque en algunas ocasiones estas responsabilidades son asignadas a un grupo de personas, debido a su gran tamaño o a alguna característica particular.

Breve Historia

En un principio, durante el desarrollo de los primeros DMBS, no se contemplaba la administración de la base de datos en la incursión del nuevo software, una excepción fue con el desarrollo de SC-1 de Wester-Electric, el papel de Administrador de la Base de Datos fue concebido tempranamente en el desarrollo de sus sistemas manejadores de BD, asignaron a una persona la responsabilidad central para coordinar el control de la base de datos corporativa. Además instituyeron una escuela para administradores de Bases de Datos antes de que la versión prototipo del DMBS fuera operacional.

A la administración de la base de datos le concierne básicamente el asegurar que la información sea precisa y consistente, y que esté disponible para los usuarios y para las aplicaciones cuando ésta se necesite y en la forma que se requiera.
Algunas de las funciones importantes del DBA son las siguientes:

1. Diseñar la BD
2. Asegurar la integridad de la BD

3. Seguridad de la BD
4. Respaldo y recuperación de la BD
5. Monitoreo del desempeño y afinación de la BD
6. Asegurar la disponibilidad de los datos

2.1 Diseño

Para un correcto diseño y una adecuada creación de la base de datos relacional, el DBA tiene que entender y apegarse a las prácticas del diseño relacional, es decir, la teoría relacional y la implementación específica en el DBMS a usar. El diseño de la base de datos requiere de alguna técnica de modelado, en este trabajo hablaremos del modelo entidad-relación, que es el más usado para modelar BD y que todo DBA debe conocer.

2.1.1 Modelado Entidad/Relación

El modelado de datos es el proceso de analizar las cosas que le interesan a la empresa, y cómo están relacionadas unas a otras, el resultado de proceso de modelado de datos es descubrir y documentar la información y recursos de la empresa. El modelo de datos entidad-relación (E-R) introducido por Peter Chen en 1976 se desarrolló para facilitar el diseño de las bases de datos. Este modelo está basado en la percepción del mundo real y consta de un conjunto de elementos llamados: Entidades, Relaciones y Atributos.

Entidades y conjuntos de entidades.
Una entidad es una "cosa" u "objeto" en el mundo real, que es distinguible de todos los demás objetos, por ejemplo: Pedro Ramírez, con CURP RAPE-153475MM12-1, es un ejemplo de una entidad, ya que indica únicamente una persona especifica. Una entidad puede ser concreta, como un libro, una persona o abstracta, como un préstamo o un concepto.

La entidad se representará por un rectángulo con el nombre de la misma en su interior. La Figura 2.1 ejemplifica este hecho.

Figura 2.1

Un conjunto de entidades es un grupo de entidades del mismo tipo, por ejemplo, en un banco, el conjunto de entidades cliente, los conjuntos de entidades no necesitan ser disjuntos, es posible definir los conjuntos de entidades de empleados y clientes de un banco, pudiendo existir una persona que es ambas o ninguna de las dos cosas.

Una entidad está representada por un conjunto de características o atributos. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. Así, cada entidad se describe por medio de un conjunto de pares (atributo, valor de dato), un par de cada elemento del conjunto de entidades.

Entonces, una BD incluye una colección de conjuntos de entidades cada uno de los cuales contiene un número cualquiera de entidades del mismo tipo.

Atributos.

Los atributos se denominan también propiedades y son como ya mencionamos características de una entidad. Un atributo en el modelo E-R se puede clasificar entre los siguientes tipos:

1. Atributos simples y compuestos:
2. Atributos univaluados y multivaluados:
3. Atributos nulos:
4. Atributo derivado:

Los atributos se representan por óvalos ligados al rectángulo mediante una línea recta La Figura 2.2 muestra los atributos de la entidad Cliente.

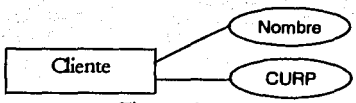


Figura 2.2

Relaciones y conjuntos de relaciones.

Una relación es una asociación entre varias entidades, por ejemplo, podemos definir una relación que asocia al cliente Fuentes con la cuenta 401.

Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Formalmente, es una relación matemática con n $>=$ 2 conjuntos de entidades (posiblemente no distintos). Si E_1, E_2, ...,E_n son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de

$$\{(e_1, e_2, ..., e_n) \mid e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n\} \qquad \text{donde } (e_1, e_2, ..., e_n) \text{ es una relación.}$$

La asociación entre conjuntos de entidades se referencia como participación; es decir, los conjuntos de entidades E_1, E_2, ...,E_n participan en el conjunto de relaciones R. Un ejemplo de relación en un esquema E-R representa que existe una asociación entre las llamadas entidades en el desarrollo del mundo real que se modela.

La mayoría de los conjuntos de relaciones en un sistema de BD son binarios, aunque ocasionalmente hay conjuntos de relaciones que implican más de dos conjuntos de entidades, como la relación entre cliente, cuenta y sucursal. Siempre es posible sustituir un conjunto de relaciones no binario por varios conjuntos de relaciones binarias distintos.

La relación se representa por un rombo y una línea recta, en el interior se escribe el nombre de la relación y en la línea que una a la relación con la entidad se escribe la acción, ambos hacen referencia al *rol* o *papel* que la relación desempeña. La figura 2.3 muestra un ejemplo de la relación Cliente – Cuenta.

Figura 2.3

El grado de una relación es el número de entidades participantes.

Restricciones al conjunto de relaciones.

Una modelación E-R de una empresa puede definir ciertas restricciones a las cuales deben ajustarse los contenidos de una BD. La cardinalidad y la participación son los dos conceptos más importantes que permiten modelar las restricciones que la base de datos debe contemplar en caso de ser necesario. La cardinalidad expresa el número de entidades con las que puede asociarse otra entidad mediante un conjunto de relaciones.

Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B, la cardinalidad de asignación debe ser una de las siguientes:

- Una a una. Una entidad en A está asociada a lo sumo con una entidad en B, y una de B a lo sumo con una de A.
- Una a muchos. Una entidad en A está asociada con un número cualquiera de entidades de B, pero una de B sólo puede estar asociada a una de A.
- Muchas a una. Una entidad en A está asociada con una sola de B, sin embargo, en de B puede estar asociada con varias de A.
- Muchas a muchas. Una entidad en A está asociada con un número cualquiera en B, y una en B está asociada con un número cualquiera en A.

La cardinalidad de asignación adecuada para un conjunto de relaciones determinado es dependiente del mundo real que el conjunto de relaciones está modelando.

En la Figura 2.3, la cardinalidad de dicha relación se representa por la punta de flecha, indicando ésta que se relaciona con una entidad del conjunto de entidades que esta relacionando, e indicando que se relaciona con mas de una entidad del conjunto de entidades si esta punta de flecha no esta presente.

Las dependencias de existencia constituyen otra clase importante de restricciones, específicamente si la existencia de la entidad X depende de la existencia de la entidad Y, entonces se dice que es dependiente por existencia de Y. Operativamente, esto significa que si se suprime Y, también se suprime X. La entidad Y se dice que es una entidad dominante, mientras que X se dice que es una entidad débil.

Clave.

Es importante poder especificar cómo se distinguen las entidades y las relaciones. Conceptualmente, las entidades individuales y las relaciones son distintas, pero, desde la perspectiva de una BD, la diferencia entre ellas debe expresarse en términos de sus atributos. El concepto de superclave nos permite hacer dicha distinción.

Una superclave es un conjunto de uno o más atributos que, considerados conjuntamente, nos permiten identificar de forma única a una entidad dentro del conjunto de entidades. Análogamente si la combinación de unos atributos K es una superclave, entonces también lo será algún subconjunto con K. A menudo estamos interesados en superclaves mínimas para las cuales ningún subconjunto propio es superclave, y se denominan claves candidatas.

Usaremos el término clave primaria para denotar una clave candidata que elige el diseñador de la BD como el medio principal de identificar entidades dentro de un conjunto de entidades.

Los conceptos de entidades fuertes y débiles están relacionados con las dependencias por existencia. Un miembro de un conjunto de entidades fuerte es, por definición una entidad dominante, mientras que un miembro de un conjunto de entidades débil es una entidad subordinada por definición. Aunque un conjunto de entidades débil no tiene clave primaria debe tener un medio de distinguir entre todas aquellas entidades en el conjunto de entidades que dependen de una entidad fuerte determinada. La clave primaria de un conjunto de entidades débil esta formada por la clave primaria del conjunto de entidades fuerte de la que depende su existencia.

Generalización.

Considérese el conjunto de entidades cuenta, con atributos número_cuenta y saldo. Ampliamos nuestro ejemplo anterior clasificando cada cuenta entre cuenta_ahorros y cuenta_cheques. Cada una de éstas se describe mediante un conjunto de atributos que incluye todos los atributos del conjunto de entidades cuenta más atributos adicionales.

Por ejemplo, las entidades cuenta_ahorros se describen además con el atributo tasa_interés, y las cuenta_cheques con el atributo saldo_deudor. Existen aspectos similares entre los conjuntos, en el sentido de que tienen varios atributos en común (los originales de cuenta). Esto puede expresarse por generalización, que es una relación de inclusión que existe entre un conjunto de entidades de un nivel más alto y uno o más conjuntos de entidades de nivel más bajo.

En términos de un diagrama E-R, la generalización se representa por medio de un componente triángulo etiquetado ISA (is a, es un/a) y representa, por ejemplo, que una cuenta de ahorros es una cuenta.

La Figura 2.4 muestra un ejemplo donde se ilustra el uso de todos los elementos básicos del modelo entidad-relación descritos en esta sección.

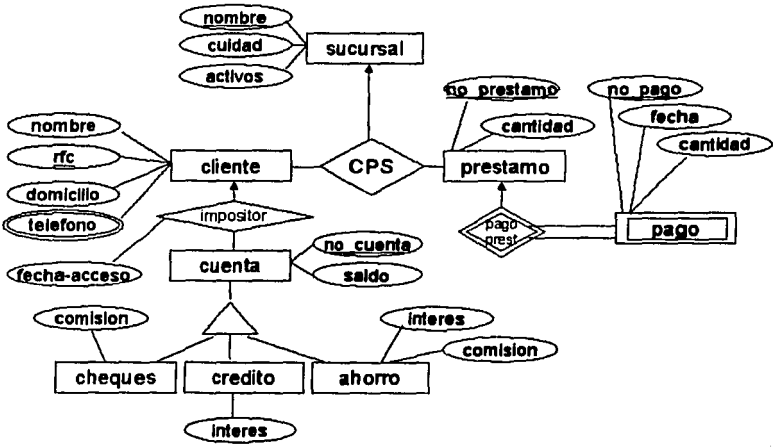


Figura 2.4 Ejemplo del modelo entidad-relación

Los siguientes puntos son restricciones al modelado:

- Los atributos pertenecen a las entidades o a las relaciones.
- Nombres únicos para las relaciones y entidades dentro del esquema.
- Nombres únicos para los atributos dentro de una entidad o relación, pero no dentro del esquema.
- Las relaciones deben darse entre al menos dos conjuntos de entidades aunque no necesariamente distintos.
- El nombre de un rol o papel, debe ser único y distinto tanto de la entidad como de la relación.

2.1.2 Normalización.

En términos simples, la normalización de una base de datos es el proceso de identificar el mejor lugar donde los datos pertenecen; es una aproximación a un diseño que minimiza la redundancia de información y optimiza las estructuras de datos sistemáticamente, además coloca los elementos de datos en grupos de manera adecuada.

El proceso de normalización fue propuesto por E.F. Codd a principios de los 70's, como el modelo relacional de datos, el proceso de normalización esta basado en principios matemáticos de la teoría de conjuntos. Un modelo de datos normalizado asegurará que cada entidad está bien formada y que cada atributo es asignado a la adecuada entidad.

Con un pequeño análisis se puede ver que la relación de la Tabla 2.1 no está bien diseñada. Por ejemplo, las dos tuplas para el trabajador 1412 repiten el mismo nombre y la información del tipo de oficio, esta redundancia en los datos o repetición no sólo ocupa espacio, sino que puede conducir a perder la integridad de los datos en la base de datos.

Id_trabajador	Nombre	Tipo_de_oficio	Id_edif	Id_supv
1235	M. Faraday	Electricista	312	1311
1235	M. Faraday	Electricista	515	1311
1412	C. Nemo	Plomero	312	
1412	C. Nemo	Plomero	460	

Tabla 2.1

El problema surge por el hecho de que un individuo puede estar trabajando en más de un edificio al mismo tiempo. Suponga que el tipo de oficio de C. Nemo esta erróneo, y sólo la primera tupla esta correcta, entonces se tendría una inconsistencia entre las tuplas que contienen información sobre C. Nemo, esto se llama una anomalía de actualización.
La descomposición es el proceso de dividir relaciones en múltiples relaciones para eliminar las anomalías y mantener la integridad de los datos, dada una relación $R(A_1, A_2, ..., A_n)$, se puede descomponer en dos relaciones $S(B_1, B_2, ..., B_j)$ y $T(C_1, C_2, ...,C_j)$, tales que:

i. $R\{A_1, A_2, ..., A_n\} = S\{B_1, B_2, ..., B_j\}$ $T\{C_1, C_2, ...,C_j\}$
ii. Las tuplas en la relación S son la proyección sobre $\{B_1, B_2, ..., B_j\}$ de las tuplas en R.
iii. De igual menera, las tuplas en T son la proyección de $\{C_1, C_2, ...,C_j\}$ sobre R.

Para hacer esto se usan las formas normales o reglas para relaciones estructuradas.

Primera forma normal (1FN)
Una relación está en primera forma normal si los valores en la relación son atómicos para cada atributo en la relación. Esto quiere decir simplemente que los valores de los atributos no pueden ser un conjunto de valores o un grupo repetitivo, la definición de Codd de una relación incluye la condición de que la relación este en 1FN, por tanto, todos los esquemas relacionales que se encontrarán estarán en primera forma normal. Para las próximas tres formas normales se aplica el concepto de dependencia funcional en entidades.

Dependencias funcionales (DFS)
Las dependencias funcionales proveen una manera para definir restricciones adicionales a un esquema relacional. La idea esencial es que el valor de la tupla en un atributo determina unívocamente el valor de la tupla en otro atributo por ejemplo, en la tabla 2.1, en cada tupla id_trabajador determina especialmente nombre, esta dependencia funcional se escribe de la siguiente forma:

F: id_trabajador → nombre

Más formalmente se define una dependencia funcional como sigue: si A y B son conjuntos atributos en la relación R, entonces:

$$F: A \rightarrow B$$

Significa que si cualesquiera dos tuplas *t* en R coinciden en su valor en A, deben coincidir en sus valores en B.

Una clave puede definirse como un conjunto de atributos tales que:

- Determinan funcionalmente cualquier otro atributo de la relación. Es decir, es imposible para dos tuplas distintas de R coincidir en todos los $\{ a_1, a_2, ..., a_n \}$
- Ningún subconjunto propio determina funcionalmente los otros atributos de R, es decir, debe ser mínimo.

Una superclave es un conjunto de atributos que contiene una clave.

Forma normal de Boyce-Codd (FNBC)

Un esquema de relación R está en la forma normal de Boyce-Codd respecto a un conjunto de dependencias funcionales F si para todas las dependencias funcionales en F^+(conjunto de DF's que son lógicamente implicadas o inferidas por F) de la forma $A \rightarrow B$, donde $A \subseteq R$ $B \subseteq R$ se cumple al menos una de las siguientes condiciones:

- $A \rightarrow B$ es una dependencia funcional trivial (es decir $A \subseteq B$)
- A es una superclave del esquema R

No toda descomposición FNBC preserva las dependencias. Más aun no siempre se pueden satisfacer los tres objetivos del diseño:

- FNBC
- Reunión sin pérdida
- Conservación de dependencias

Tercera forma normal (3FN)

La FNBC requiere que todas las dependencias no triviales sean de la forma $A \rightarrow B$, donde A es superclave, la tercera forma normal relaja ligeramente esta condición al permitir dependencias funcionales no triviales cuya parte izquierda no sea superclave.

Un esquema de relación R está en 3FN respecto a un conjunto F de dependencias funcionales si para toda dependencia funcional en F^+ de la forma $A \rightarrow B$ donde $A \subseteq R$ y $B \subseteq R$, se cumple al menos una de las siguientes condiciones:

- $A \rightarrow B$ es una dependencia funcional trivial
- A es una superclave para R
- Cada atributo en B - A está contenido en una clave de R

Dependencias multivaluadas

Una dependencia multivaluada $A \rightarrow\rightarrow B$ es no trivial si:

- Ninguna de la b's esta contenida en las a's
- R tiene más atributos que las a's y las b's

Un esquema de relación R está en cuarta forma normal "4FN" respecto a un conjunto F de dependencias funcionales y multivaluadas si para toda dependencia multivaluada en F^+ de la forma $A \rightarrow B$ donde $A \subseteq R$ y $B \subseteq R$, al menos una de las siguientes condiciones se cumplen:

$A \rightarrow B$ es una dependencia multivaluada trivial

A es una superclave en R

2.1.3 Del modelo lógico al modelo físico

El modelo de datos físico es creado transformando el modelo de datos lógico en una implementación física basada en el DBMS a usar para el desarrollo. Para tener éxito en la creación del diseño físico de la base de datos, el DBA deberá tener una buena experiencia en las características de DBMS, incluyendo:

- Amplio conocimiento en los objetos de la base de datos soportados por el DBMS y las estructuras físicas y los archivos requeridos para soportar dichos objetos.
- Considerar los detalles de la manera en que el DBMS soporta indexación, integridad referencial, restricciones, tipos de datos y otras características que aumenten la funcionalidad de los objetos de la base de datos.

- Conocimiento detallado de nuevas y obsoletas características para versiones particulares del DBMS
- Conocimiento de los parámetros de configuración del DBMS
- Amplio conocimiento o experto en el lenguaje de definición de datos para traducir el diseño físico en los objetos de la base de datos

Transformar entidades a tablas

La contraparte física de una entidad es una tabla, por lo tanto, el primer paso en transformar el modelo lógico de datos en la base de datos física es convertir cada entidad en el modelo de dato a una tabla en la base de datos, la base de datos final que se implemente no necesariamente tiene que conservar una relación estricta de uno a uno con las entidades y las tablas.

Transformar atributos a columnas

La contraparte física de un atributo es una columna en una tabla, cuando transformamos entidades en tablas transformamos atributos en columnas, se debe tratar de mantener la misma convención de nombres para las columnas físicas que el nombre usado en los atributos lógicos, sin embargo esto no siempre es posible pues depende de las características del DBMS.

Transformar dominios en tipos de datos

Para soportar el transformar atributos en columnas es necesario transformar cada dominio lógico del atributo a un tipo de dato y tal vez adicionalmente a restricciones.

Cada columna tiene entonces que ser asignada a un tipo de dato, ciertos tipos de datos requieren que se especifique un máximo de longitud, por ejemplo tu puedes especificar un tipo de dato carácter como `CHAR(20)`, tal vez será necesario aplicar una longitud a otros tipos de datos tales como gráficos, flotante, decimal, etcétera.

Los DBMS's comerciales no siempre soportan el tipo de dato asignado durante el modelo de datos lógico es por eso, que es muy importante considerar las características del DBMS durante el modelo de datos lógico, para así soportar los tipos de datos. Algunos de los mejores DBMS's soportan tipos de datos definidos por el usuario.

Como DBA es necesario determinar qué tipo de dato puede ser más eficientemente accesado, almacenado, mantenido y procesado por las aplicaciones que lo accesan, para llevar a cabo tal decisión, el DBA requiere un conocimiento técnico importante en la manera en que el DBMS almacena físicamente cada tipo de dato.

Restricciones.

Consideremos ahora un dominio de enteros de 1 a 10 para asignar esta columna física a un tipo de dato entero es insuficiente igualar el dominio que es entero, debemos considerar y agregar una restricción a los valores que pueden ser almacenados a la columna para especificar un rango de 1 a 10, esto lo llamaremos restricciones de columna.

Usando estas restricciones podremos especificar límites para los valores de los datos que pueden ser almacenados en una columna o un conjunto de columnas.

La nulavilidad de cada columna en la tabla tiene también que ser especificada, el modelo de datos lógico debe contener información sobre la nulavilidad de cada atributo, y esta información puede ser copiada para cada requisito en la columna física de la base de datos, algunos DBMS permiten asignar un valor por omisión para ser usada cuando un renglón es insertado y ningún valor es provisto para esa columna.

Para tipos de dato carácter o texto el DBA tiene que tomar en cuenta que la columna puede ser de longitud fija o longitud variable. Una columna de longitud fija ocupa un espacio fijo para almacenar cada renglón, y una columna de longitud variable especifica un tamaño máximo, pero el espacio usado por la columna puede variar para cada renglón. Además se debe considerar que los cambios que se hacen a cada renglón cuando la longitud de la columna es variable pueden tener efectos en el sentido que provoquen que los renglones se muevan dentro de la base de datos a nivel físico, es decir que un renglón ocupe más de dos bloques o ocupe más de un apuntador para especificar donde se encuentra la información, este fenómeno se le conoce como migración de registros.

Llave primaria

La especificación de una llave primaria es una parte integral en el diseño físico de entidades y atributos, cuando se diseña el modelo de datos lógico se asigna una llave primaria para cada entidad. La selección de esta llave primaria es muy importante ya que de escoger una

llave no apropiada la base de datos puede tener problemas de rendimiento. La mayoría de los DBMS proveen características que pueden asistir en la definición de llaves primarias, algunos ejemplos son ROWID, autoincrementable, etcétera.

Orden de las columnas

Antes de implementar una tabla física, el DBA debe de revisar el orden de las columnas, ya que aunque para el desarrollador el orden es irrelevante o para el modelador de datos el resultado es el mismo, para el DBMS el orden de las columnas puede provocar problemas de rendimiento, es decir, una consulta hecha en SQL puede ser más rápida u obtener los resultados de una manera más eficiente para un distinto orden de columnas en la tabla física.

Restricciones de referencia

La contraparte física de una relación es una restricción de referencia. Para definir una restricción de referencia se crea una llave primaria en una tabla padre y una llave foránea en una tabla dependiente, la restricción de referencia enlaza la llave primaria con la llave foránea.

No es suficiente identificar las llaves primarias y foráneas que relacionan las tablas, la funcionalidad de cada relación es afectada por los parámetros seleccionados para la restricción de referencia, y los valores de la columna de la llave foránea. Un conjunto de reglas se definen para cada relación y determinan el estatus de la columna de la llave primaria cuando se inserta o actualiza, y otro cuando un renglón de la llave primaria es borrado, por ejemplo cuando una llave primaria es borrada que hace referencia a los valores existentes en la llave foránea se pueden aplicar dos reglas:

- Los reglones de la tabla dependiente que hace referencia a la llave primaria sean borrados, esto lo conocemos como borrado en cascada.
- Los valores para la columna que hacen referencia al valor de la llave primaria sean asignados con un valor nulo.

Estas reglas que se aplican a las restricciones de referencia, se les conoce como reglas de integridad referencial, y es una buena practica de diseño.

Relación de estructuras lógicas y físicas

Diseñar e implementar una base de datos física a partir del modelo de datos lógico no es simplemente transformar entidades en tablas, atributos en columnas y relaciones en restricciones de referencia, sino que el DBA debe preocuparse también por preparar el espacio lógico y físico para los datos, las tablas deberán almacenarse en estructuras lógicas.

Estas estructuras lógicas son comúnmente llamadas espacio de datos o (tablespaces). Una base de datos comprende uno o más espacios de datos. Estos espacios de datos tendrán asociados a su vez uno o más archivos físicos (datafiles) ver Figura 2.5

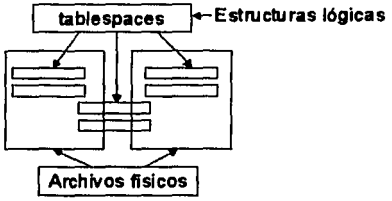


Figura 2.5 relación de estructuras lógicas y físicas

Vistas

Otro aspecto del diseño de la base de datos física es la creación de vistas en la base de datos, éstas son usadas para soportar los requerimientos específicos de las aplicaciones de datos. Las vistas no son requeridas para acceder a la base de datos física, pero pueden ayudar a soportar los requerimientos de los usuarios, ninguna estructura física es requerida para una vista, esta es una representación de datos que es almacenada en otras tablas o en otras vistas.

Las vistas pueden consistir de cualquiera de las siguientes combinaciones:

- Registros de tablas
- Registros de vistas
- Columnas de tablas
- Columnas de vistas

Existen seis características para uso de las vistas:

- Proveen seguridad a nivel de renglón y de columna, garantizan ruta eficiente de acceso
- Garantizan ruta eficiente de acceso
- Renombran tablas
- Renombran columnas
- Dan formato sencillo a cosas complejas
- Garantizan consultas adecuadas

2.2 Integridad

Garantizar la integridad de la base de datos de la empresa es un componente clave en el trabajo del DBA, una base de datos no es usada si los datos que contiene son inadecuados o no pueden ser accesados por problemas de integridad.

Se consideran dos tipos de integridad para las bases de datos:

- Integridad de la estructura de bases de datos
- Integridad semántica de los datos

La integridad de la estructura tiene como meta asegurar que cada objeto de la base de datos es creado, formateado y mantenido de forma adecuada. Cada DBMS usa su propio formato y estructura interna para soportar las bases de datos, los espacios de datos, las tablas e índices bajo su propio control. Los errores de sistema y aplicación pueden ocasionar fallas dentro de esta estructura interna, y es trabajo del DBA identificar y corregir tales fallas.
La integridad semántica de los datos se refiere al significado de los datos y su relación que necesita ser mantenida entre los diferentes tipos de datos, un ejemplo de integridad semántica es la calidad de los datos en la base de datos, supóngase que el 25% de los clientes en la base de datos tienen su dirección equivocada o su número telefónico, este es un ejemplo de una base de datos con calidad pobre. Redundancia es punto en la integridad

semántica, pues existe la posibilidad que la redundancia de los datos no este sincronizada, es decir que existen diferentes valores de un mismo dato.

El DBA debe implementar la integridad semántica de los datos en el diseño de la base de datos como una buena iniciativa de proceso, de acuerdo con las características del DBMS para así corregir y evitar problemas de integridad que se puede acarrear en la base de datos tiempo después.

Los DBMS proveen opciones y procedimientos para definir y asegurar la integridad semántica de los datos almacenados en la base de datos, el DBA debe entender como el DBMS habilita de manera automática la integridad semántica de los datos.

2.2.1 Integridad de la estructura de la base de datos

La integridad y consistencia de la estructura de la base de datos es tal vez el aspecto más importante de la administración de la base de datos. El DBMS usa estructuras internas y apuntadores para mantener los objetos de la base de datos en correcto estado
Un problema común en las bases de datos es la corrupción de índices, estos como sabemos son apuntadores a las direcciones físicas de los datos de una tabla, si el índice no apunta al dato correcto éste es inútil.
Otro problema de integridad de la estructura es la corrupción del "header" del bloque que contiene cierto dato, el header permite al DBMS leer rápida y fácilmente el contenido del bloque, si el bloque está corrupto el DBMS tal vez no pueda leer los datos almacenados en dicho bloque, el tales situaciones, se requiere usualmente, que los archivos de la base de datos sea recuperados de un respaldo de la misma.
Daño en los archivos de respaldo es otro de los problemas y que representan un alto riesgo en la integridad de toda la base de datos.
Cada DBMS proporciona diferentes utilerías para revisar los diferentes aspectos de la integridad de la estructura de la base de datos, por ejemplo Sysbase y SQL Server

proporcionan la utilería DIC, DB2 proporciona las utilerías CHECK y REPAIR, Infromix TBCHECK y Oracle el paquete DBMS_REPAIR.
Todos ellos realizan el análisis de daños físicos en la base de datos es decir daños en la estructura, en algunos casos estos programas corrigen los daños en los bloques afectados.

Los pasos que podemos seguir para corregir daños en la estructura física son:

- Detectar y elaborar reporte de los daños.
- Evaluar el costo y beneficio de usar la utilería.
- Tratar de hacer los objetos usables, cuando sólo una parte de ellos este dañada.
- Reparar los daños y reconstruir los datos perdidos.

Cuando hablamos de daños en la estructura, dos objetivos son los más importantes:

a) Consistencia de la base de datos
b) Evitar perdida de datos

Depende del DMBS, las facilidades para cumplir con estos objetivos, pues la tarea puede ser tan sencilla como reconstruir un índice o tan compleja como recuperar toda la base de datos.

2.2.2 Integridad semántica de la base de datos

La integridad semántica va de la mano con las características y procesos del DBMS, mientras que la integridad de la estructura de la base de datos se refiere a la consistencia de las estructuras físicas que contienen a los datos, la integridad semántica de los datos se refiere a la consistencia de los datos mismos.
Los DBA's luchan constantemente con la pregunta de que es mejor para asegurar la integridad de los datos: usar las características del DBMS o usar el código de la aplicación. En general, usar las características del DBMS ofrecen la mejor solución, pues es más sencillo descubrir y mantener una regla de integridad semántica y en general se necesita menos código.

Integridad de Entidad

Integridad de entidad es el nivel más básico que proporcionan las base de datos relacionales, y se refiere a que cada tupla de una entidad debe ser identificada de manera única, es decir que requiere la especificación de una llave primaria, aunque algunos DBMS's permiten crear entidades sin la especificación de una llave primaria, esto se considera una mala práctica por que hace difícil la identificación de registros.

Junto con la creación de una llave primaria el DBMS requiere la creación de un índice de valores únicos en la o las columnas de la llave primaria, la mayoría de los DBMS's crean automáticamente dicho índice. Un ejemplo de una restricción de llave primaria es el siguiente:

```
CREATE TABLE EMP
(empno          INTEGER          PRIMARY KEY,
fecha_ingreso   DATE                 DEFAULT SYSDATE,
emp_direccion   VARCHAR(70),
emp_tipo        CHAR(8),
emp_depto       CHAR (3)  NOT NULL,
salario         NUMBER(7,2)  NOT NULL,
comision        NUMBER(7,2),
);
```

Restricciones de unicidad

Una restricción de unicidad de valores es similar a una llave primaria, los valores almacenados en la columna o combinación de columnas debe ser única dentro de la tabla, esto es, que ningún otro registro debe contener dicha combinación.

Las restricciones de unicidad difieren de las restricciones de llave primaria en que ellos no pueden ser usados para soportar las restricciones de referencia, más aún, las columnas de la restricción de unicidad pueden tener el valor de nulo.

Tipos de datos

Tipos de datos y longitud de datos son la restricción de integridad más fundamental aplicada a los datos en la base de datos, es decir que simplemente especificando el tipo de dato para cada columna cuando una tabla es creada, el DBMS automáticamente se asegura que sólo los tipos de datos correctos serán almacenados en dicha columna. El DBA debe escoger el tipo y longitud del dato para cada columna de manera prudente, y es lo mejor escoger el tipo de dato lo más cercano al dominio de la columna definido en el diseño. Por ejemplo

una columna numérica debe ser definida como uno de los siguientes tipos de datos numéricos: entero, decimal o punto flotante.

Tipos de datos definidos por el usuario

Los tipos de datos definidos por el usuario, proveen un mecanismo que extiende los tipos de datos que pueden ser almacenados en la base de datos y la manera en que los datos serán tratados, en otras palabras, el DBA puede crear tipos de datos para especificar los valores legales de una columna, los siguientes son algunos ejemplos de creación de tipos de datos definidos por el usuario:

```
CREATE DISTINCT TYPE canadia_dollar AS DECIMAL(11,2);
CREATE DISTINCT TYPE US_dollar AS DECIMAL(11,2);
CREATE DISTINCT TYPE euro AS DECIMAL(11,2);
CREATE DISTINCT TYPE japanese_yen AS DECIMAL(15,2);
```

Con la creación tipos de datos definidos por el usuario, el DBMS prohíbe operaciones no definidas, como podría ser el siguiente ejemplo:

```
TOTAL_AMT = US_dollar + canadian_dollar
```

De ésta manera vemos que la creación o el uso de tipos de datos definidos por el usuario nos ayuda a evitar problemas de integridad de datos

Valores por omisión

Cuando una tabla es creada, es posible especificar valores por omisión a las columnas de la tabla, éstos serán usados cuando se haga uso de la instrucción INSERT de SQL y se provea de un valor explícito para alguna es estas columnas.

Los valores por omisión permiten a los programadores ignorar columnas durante el uso de INSERT pues el DBMS proporcionará manera automática los valores por omisión.

Restricciones de verificación (check constraints)

Una restricción de verificación es una restricción definida en el DBMS sobre los valores que pueden ser almacenados en una o varias columnas de la tabla, esta restricción es definida explícitamente en la definición de la tabla y es formulada de la misma manera que la instrucción WHERE de SQL. Cualquier intento de modificar el dato de una columna (es

decir, usar la cláusula INSERT o UPDATE) que pertenece a una restricción de verificación provocará la evaluación de la misma antes de realizar la operación.
La restricción de referencia consiste en dos componentes: el nombre de la restricción y la condición de verificación. La condición de restricción puede ser definida usando cualquiera de los predicados básicos (<, >, <=, >=, <>), como también BETWEEN, IN, LIKE y más aún AND y OR.

Sin embargo, hay algunas restricciones que se tienen cuando se define una restricción de verificación:

- La restricción de verificación sólo puede hacer referencia a las columnas en la tabla que está definida.
- Un conjunto limitado de cláusulas SQL es permitido dentro de la restricción de referencia.
- El primer operador de la restricción es el nombre de la columna y el segundo es el nombre de otra columna o de una constante.
- Si el segundo operador es una constante, el tipo de dato debe ser compatible con tipo de dato definido para el primer operador, y si el segundo operador es una columna, esta debe ser del mismo tipo de dato que la columna que esta definida como el primer operador.

El siguiente ejemplo muestra la definición de algunas restricciones de verificación:

```
CREATE TABLE EMP
(emp_no         INTEGER         PRIMARY KEY,
                  CONSTRAINT        check_emp_no
                  CHECK (empno      between 100 and 25000),
emp_direccion VARCHAR(70),
emp_tipo        CHAR(80)
                  CHECK (empno      between 100 and 25000),
emp_dept        CHAR(3)      NOT NULL with default,
salario         DECIMAL(7,2)    not null,
                  CONSTRAINT      check_salario
                  CHECK (salario      < 50000.00),
comision        DECIMAL(7,2),
bono            DECIMAL(7,2)
                  CONSTRAINT      com_salario
                  CHECK (salario      > comision),
          CONSTRAINT        com_bono
          CHECK (comision = 0 or bono = 0    )
);
```

CONSTRAINT check_emp_no CHECK (empno between 100 and 25000): Revisa que el atributo empno tenga valores entre 100 y 25000.

CONSTRAINT check_salario CHECK (salario < 50000.00): Revisa que el salario sea menor a 50000.
CONSTRAINT com_salario CHECK (salario > comision): Revisa que la comisión sea menor que el salario.

CONSTRAINT com_bono CHECK (comision = 0 or bono = 0): Revisa que la comisión sea iguala a cero o que el bono se a iguala cero.

Disparadores (triggers)

Los triggers son procedimientos controlados por eventos que son agregados a tablas dentro de las bases de datos o al sistema manejador. Un trigger es código que es ejecutado en respuesta a una sentencia de modificación, es decir un INSERT, DELETE o UPDATE.

Un trigger como ya mencionamos, esta escrito en alguna extensión de SQL que puede ser visto como una combinación del algún lenguaje de programación y SQL estándar, algunos ejemplos son:

Pascal y SQL	Oracle
Java y SQL	Oracle, DB2, SQLServer
C y SQL	Oracle
Tcl y SQL	Postgres

Por ejemplo en postgres se tiene la siguiente sintaxis:

```
CREATE TRIGGER trigger [ BEFORE | AFTER ] [ INSERT | DELETE | UPDATE [
OR ... ] ]
ON relation FOR EACH [ ROW | STATEMENT ]
EXECUTE PROCEDURE procedure
(args);
```

donde el procedimiento puede ser escrito en alguno de los lenguajes que soporte el DBMS, mediante el siguiente comando:

```
CREATE [ OR REPLACE ] FUNCTION name ( [ argtype [, ...] ] )
RETURNS rettype
AS 'definition'
LANGUAGE langname
[ WITH ( attribute [, ...] ) ]
CREATE [ OR REPLACE ] FUNCTION name ( [ argtype [, ...] ] )
RETURNS rettype
AS 'obj_file', 'link_symbol'
LANGUAGE langname
[ WITH ( attribute [, ...] ) ]
```

Los trigger son bastante flexibles, actualmente los sistemas manejadores implementan varios lenguajes para poder explotar al máximo esta capacidad, algunos de los propósitos por los cuales pueden ser creados son:

- Especificar restricciones de integridad complejas
- Imprimir mensajes adicionales
- Acceder y modificar más de una tabla al mismo tiempo

El siguiente es un ejemplo de un trigger en Oracle invocando un procedimiento en Java para registrar en la tabla LOGTAB los atributos del registro que se va a eliminar en la tabla TAB:

Se crea el procedimiento **Before_delete** que define la clase y el método a usar de un programa en java.

```
CREATE OR REPLACE PROCEDURE Before_delete (Id IN NUMBER, Ename
VARCHAR2)
IS language Java
name 'thjvTriggers.beforeDelete (oracle.sql.NUMBER, oracle.sql.CHAR)';
```

Trigger que define el procedimiento a usar después de borrar en la tabla TAB.

```
CREATE OR REPLACE TRIGGER Pre_del_trigger BEFORE DELETE ON Tab
FOR EACH ROW
CALL Before_delete (:old.Id, :old.Ename)
```

```
thjvTriggers.java

import java.sql.*
import java.io.*
import oracle.sql.*
import oracle.oracore.*
public class thjvTriggers
{
public state void
beforeDelete (NUMBER old_id, CHAR old_name)
Throws SQLException, CoreException
   {
   Connection conn = JDBCConnection.defaultConnection();
   Statement stmt = conn.CreateStatement();
   String sql = "insert into logtab values
   ("+ old_id.intValue() +", '"+ old_ename.toString() + ", BEFORE
   DELETE');
   stmt.executeUpdate (sql);
   stmt.close();
   return;
   }
}
```

Integridad Referencial

La integridad referencial es un método para asegurarse que los datos sean correctos dentro de la base de datos, ésta se refiere a la integridad y usabilidad de las reglas determinadas por

una relación. Los conceptos más importantes son: tabla padre y tabla hija; la tabla padre es la tabla que contiene la llave primaria y la tabla hija es aquella que contiene la llave foránea en una restricción de referencia, el siguiente es un ejemplo de una restricción de referencia:

Existen tres tipos de reglas que pueden ser asociadas a una restricción de referencia:

- Regla INSERT: esta indica que pasará si se intenta insertar un dato en la o las columnas de la llave foránea sin la correspondiente llave primaria, dos casos se presentan para esta regla y que aplican a las columnas de la llave foránea:
  - No se permite insertar
  - Se permite el valor de nulo
- Regla UPDATE: controla las actualizaciones tal que, el valor de la llave foránea no puede ser actualizado a un valor que no corresponda a un valor en la llave primaria de la tabla padre, tres casos se presentan para las columnas de la llave primaria:
  - Restringido: No se permite actualizar si existen valores que correspondan en la llave foránea.
  - Neutral: Todos los valores de la llave foránea se asignan a nulo.
  - Cascada: Todos los valores de la llave foránea se actualizan en cascada al valor nuevo de la llave primaria.
- Regla DELETE: define que pasa si se intenta borrar un registro de la tabla padre y existe una restricción de referencia, es decir una tabla hija y la definición de la restricción.
  - Restringido: No se permite borrar si existen valores en la llave foránea.
  - Neutral: Todos los valores de la llave foránea asociados a los valores de la llave primaria de registro que se borra, se asignan a nulo.
  - Cascada: Todos los registros, asociados por la llave foránea y que contenga el valor de la llave primaria del registro a eliminar, serán eliminados.

Cada una de las reglas anteriores, aplican según la definición del trigger, donde está puede ser antes o después de que ocurra el evento, el ejemplo anterior lo muestra así.

```
CREATE OR REPLACE TRIGGER Pre_del_trigger BEFORE DELETE ON Tab FOR EACH
ROW
CALL Before_delete (:old.Id, :old.Ename)
```

Algunos DBMS's disponen de triggers a nivel del sistema manejador, el siguiente es un ejemplo de ello:

```
CREATE TRIGGER log_errors AFTER SERVERERROR ON DATABASE
   BEGIN
      IF (IS_SERVERERROR (1017)) THEN
         <special processing of logon error>
      ELSE
         <log error number>
      END IF;
   END;
```

2.3 Seguridad

La seguridad es un aspecto importante desde el punto de vista de la empresa, pues es la manera de deslindar responsabilidades de la calidad los datos así como de la integridad de los mismos, la seguridad básica pero no por ello sencilla, es proporcionada por la mayoría de los sistemas manejadores de bases de datos, ésta se refiere a la autentificación del acceso a la base de datos y los recursos que en ella se encuentran, la creación de un usuario va entonces acompañada de una clave de usuario "login" y contraseña "password", que permite a un usuario acceder a la base de datos. Es importante establecer políticas con respecto a claves de usuario y contraseñas, con la finalidad de reforzar más la seguridad y evitar que personas o aplicaciones accesen a la base de datos sin autorización, cambiar la contraseña cada cierto tiempo, que esta no sea tan corta, que contenga letras y números son algunos ejemplos de ello.

Algunos DBMS's permiten especificar perfiles de usuario donde es posible indicar parámetros de acceso como son:

- Tiempo que debe transcurrir para cambiar la contraseña
- Numero de intentos fallidos para ingresar a la BD
- Tiempo de ocio dentro de la BD
- Número de usuarios conectados con la misma clave y contraseña al mismo tiempo.
- Etcétera.

El DBA implementa el nivel de seguridad disponible y apropiado, de acuerdo a las características de los DBMS's y las políticas de la empresa, algunas de las características de las cuales podemos hacer uso y que podemos encontrar en un DBMS son:

- Otorgar y revocar permisos y/o privilegios
- Roles y grupos
- Vistas y procedimientos almacenados
- Auditorias

2.3.1 Otorgar y revocar permisos y/o privilegios

El DBA controla la seguridad de la base de datos usando un lenguaje de control de datos "DCL" de inglés "data control lenguaje", este lenguaje es usado para controlar qué usuarios tienen acceso a la base de datos y a que objetos y comandos en la misma. Este lenguaje de control está compuesto por dos tipos de instrucciones:

- Grant: Asigna un privilegio a un usuario u objeto de la base de datos
- Revoke: Quita un permiso a un usuario u objeto de la base de datos.

La instrucción Grant requiere de dos listas: una lista de privilegios que se van a otorgar y una lista de usuarios que los recibirán. Para usar la instrucción grant el usuario debe tener alguna de las siguientes características:

- Ser dueño del objeto
- Tener los permisos en un nivel de administración para otorgar el o los privilegios
- Haber obtenido los privilegios y el permiso de transmitirlos (WITH GRANTOPTION)

El segmento de instrucción "WITH GRANTOPTION" permite a un usuario pasar los privilegios que en ese momento está recibiendo.
Existen diferentes tipos de privilegios que pueden ser otorgados o revocados, esto depende de las características de cada DBMS, los siguientes son algunos tipos de privilegios que están actualmente presentes en los DBMS.

- Tabla: control sobre el acceso y modificación de los datos en una tabla.
- Objeto: control sobre la creación y borrado de objetos (tablas, índices, clusters, etc.) de la base de datos
- Sistema: control sobre la ejecución de cierto tipo de actividades del sistema manejador.
- Programa: control sobre la creación, modificación y uso de programas en la base de datos.
- Procedimiento almacenado: control sobre la ejecución de funciones específicas y procedimientos almacenados.

```
GRANT UPDATE on Titulos (autor_id) to user7;
GRANT CREATE table, CREATE index to user4, user 6;
GRANT TRACE to user8;
GRANT EXECUTE on proc1 to public;
```

La instrucción revoke es usada para remover o quitar privilegios que fueron otorgados previamente. Adicionalmente si el objeto es eliminado, los privilegios serán automáticamente revocados por el DBMS, el siguiente ejemplo muestra cómo es usada la instrucción:

```
REVOKE UPDATE on titulos (autor_id) from user7;
```

2.3.2 Roles y Grupos

Adicionalmente al otorgamiento de privilegios a usuarios, algunos DBMS's proporcionan la capacidad de asignar privilegios específicos a un "rol". Un rol es esencialmente una colección de privilegios, una vez definido un rol puede ser usado para otorgar uno o más privilegios preasignados al rol. Es el DBA generalmente el encargado de realizar esta acción.

El siguiente ejemplo muestra lo antes mencionado:

```
CREATE role MANAGER;
GRANT select, insert, delete, update on employee to MANAGER;
Grant select, insert, delete, update on job_title to MANAGER;
GRANT EXECUTE on payroll to MANAGER;
```

```
GRANT MANAGER to user1;
```

Los grupos son similares a los roles, sin embargo, cada DBMS proporciona grupos ya definidos y que no pueden ser cambiados. Estos son algunos de los grupos que podemos encontrar en un DBMS

- Administrador de Sistema: Algunas veces abreviado como SA (del inglés System Administrator) o SYSADM, este grupo permite al usuario que lo posee ejecutar todos los comando de la base de datos y puede accesar a todos los objetos de todas las bases de datos.
- Administrador de Base de Datos: DBADM o DBA, este grupo permite el acceso a una base de datos específica.
- Mantenimiento de Base de Datos: DBMAINT, este grupo permite realizar tareas de mantenimiento sobre los objetos de unas base de datos, como estadísticas de índices, tablas, ejecutar comandos para observar el desempeño y características de los objetos.
- Administrador de Seguridad: abreviado como SSO, este grupo permite realizar todas las tareas de referentes a seguridad del DBMS, como son administración de cuentas y claves, auditorias, configuración de seguridad, etcétera.
- Control de Operaciones: OPER o SYSOPR, este grupo permite realizar tareas tales como respaldo y recuperación, o terminar tareas que estén en proceso de ejecución.

2.3.3 Vistas

La seguridad de la mayoría de las bases de datos está implementada mediante las características nativas del DBMS. Es posible simplificar o hacer más sencillo de implementar algunos aspectos de seguridad de la base de datos usando las vistas para proteger los datos, como ejemplo consideremos la siguiente definición de la taba empleados:

```
CREATE TABLE  emp
   (nombre      VARCHAR(30),
   ap_paterno   VARCHAR(30),
   ap_materno   VARCHAR(30),
```

```
direccion    VARCHAR(100),
estado       VARCHAR(30),
cp           INTEGER,
id_depto     INTEGER,
salario      DOUBLE);
```

Si otorgamos el privilegio de select sobre la tabla podemos tener algunos problemas de seguridad, pues tal vez no sea bueno que se permita ver todos los atributos de la tabla como son: salario, dirección, etcétera.

Una vista puede ayudar a evitar estos problemas de seguridad, otorgando el privilegio de select sobre la vista, de esta manera restringimos los datos que se pueden consultar o actualizar.

Una vista que oculta columnas de la tabla es referida como de restricción vertical.

```
CREATE VIEW vi_empleados
AS
   SELECT nombre, ap_patreno, ap_materno
     FROM emp;
```

Y es referida de restricción horizontal si se ocultan registros de la tabla.

```
CREATE view vi_empleados_depto20
AS
   SELECT nombre, ap_patreno, ap_materno, direccion, cp, salario
     FROM emp
     WHERE id_depto = 20;
```

2.3.4 Auditorias

Auditar es una de las características de algunos DBMS que permite al DBA rastrear el uso de los recursos y privilegios en la base de datos. Cuando se está auditando, el DBMS produce un archivo con el rastro o bitácora de las operaciones en la base de datos, cada operación auditada que es registrada, contiene información de a qué objeto se le aplicó dicha operación, qué usuario la realizó y en qué fecha, aunque esto depende del nivel de auditorias proporcionadas por el DBMS en uso.

Un DBMS que proporciona características de auditorias permite comúnmente los siguientes tres niveles de auditoria:

- A nivel base de datos
- A nivel de objeto de base de datos
- A nivel de usuario de la base de datos

Uno de los problemas más importantes que se presentan al habilitar las características de auditoria de un DBMS, es que degrada el rendimiento del sistema manejador en general, es por ello que si se piensa habilitar el sistema manejador con estas características se debe tener en consideración los siguientes puntos:

- Auditar puede consumir una gran cantidad de recursos del sistema.
- Colocar las tablas del diccionario de datos que almacenan la información de las auditorias en discos separados.
- Prever el espacio para el crecimiento de las tablas y/o archivos donde se almacenará la información a auditar.

2.4 Respaldo y Recuperación

Debido a que la información almacenada sobre los medios de cómputo, en específico en las bases de datos, está sujeta a la pérdida o a la corrupción ocasionada por una amplia gama de sucesos, es importante proporcionar mecanismos para la recuperación de los datos correctos en las bases de datos.

El DBA debe estar preparado para resolver situaciones donde una falla impacte la disponibilidad, integridad o usabilidad de la base de datos. Esta función del DBA es uno de los elementos principales de su trabajo. La habilidad del DBA de reaccionar correctamente, depende directamente de qué tan bien tenga elaborado un plan de respaldo y recuperación.

Las fallas que se pueden presentar en torno a las bases de datos, y que requieran la recuperación de la misma, las podemos dividir en las siguientes tres categorías:

- Falla de la instancia. Es el resultado de una falla en el DBMS o sistema operativo u otro software relacionado con el DBMS. En estos casos la falla usualmente no daña a los datos, pero la base de datos no está disponible, y normalmente basta con reinicializar nuevamente los servicios o el software afectado.
- Falla de la aplicación o transacción. Ocurre cuando el programa se ejecuta en un tiempo incorrecto o con datos incorrectos o en orden incorrecto. Usualmente esta falla corrompe los datos y se requiere de una recuperación de los mismos mediante un respaldo existente.
- Falla de hardware. Este tipo de falla ocasiona generalmente la pérdida de datos y se requiere de una buena estrategia para recuperar la base de datos.

2.4.1 Respaldos

Un respaldo es una copia de los datos y tiene como objetivo proteger y garantizar la integridad de los datos, provocado por algún error inesperado de las fallas antes mencionadas, de está manera, en el caso de la pérdida original de los datos, se puede utilizar el respaldo para reconstruir o restaurar los datos perdidos.

Un respaldo, puede, de acuerdo a la arquitectura de tres niveles para un DBMS, ser de dos tipos:

- Respaldo físico: realiza una copia de todos los archivos físicos, concernientes al uso y manejo del DBMS, para el caso de Oracle, una copia de los archivos: control files, data files, log files e init.ora.
- Respaldo lógico: realiza copias de estructuras lógicas como pueden ser: tablas, tablespaces, procedimientos almacenados, etcétera.

Los respaldos son almacenados en un formato específico por cada DBMS, por lo que es poco probable que se puede reconstruir información en un DBMS con respaldos hechos por otro DBMS diferente.

Un respaldo también puede ser divido en respaldo completo o incremental.

- Respaldo completo: es una copia completa de todos los datos y objetos en la base de datos al momento de realizarse.
- Respaldo incremental: consiste, en copiar o respaldar sólo aquellos datos que cambiaron desde el último respaldo completo o incremental.

La ventaja de realizar un respaldo incremental, es que en general toma menos tiempo en realizarse, y ocupa menos espacio en disco. Esta ventaja puede ser útil en aquellas empresas que poseen grandes cantidades de datos, como un banco, donde realizar respaldos ocupando el menor tiempo posible es un aspecto importante.

La desventaja de realizar un respaldo incremental, es que, toma más tiempo recuperar la base de datos, ya que un registro puede ser actualizado en más de una ocasión para diferentes respaldos incrementales.

Las siguientes recomendaciones para crear un respaldo de una base de datos, ayudaran a garantizar que la recuperación de la base de datos sea posible:

- Realizar al menos dos copias de cada respaldo, para evitar que no sea posible recuperar la base de datos por causa de daño físico del archivo de respaldo.
- Coordinar la estrategia de respaldo con la estrategia recuperación.
- Guardar los archivos de bitácora (log's) en varios directorios, o realizar respaldo frecuentes de los archivos de bitácora.
- Evaluar la posibilidad de generar los archivos de bitácora en dispositivos externos, pues aunque requiere de mayor tiempo son más seguros.
- Después de realizar un respaldo, usar las características del DBMS (si tiene) para verificar que el respaldo fue hecho correctamente, como ejemplo las utilerías db2ckbkp de DB2 y BCP y Sybase. En caso de no poseer utilerías, comprobar el respaldo de la manera que sea posible.

El DBA debe conocer las capacidades de respaldo del DBMS en uso y planear una estrategia de respaldo tomando en consideración los siguientes puntos:

- La necesidad de acceso concurrente y modificación durante el proceso de respaldo.
- El tiempo disponible para el proceso de respaldo y el impacto del acceso concurrente sobre el tiempo que toma el proceso.
- El tiempo que toman las utilerías de recuperación para reestablecer las bases de datos
- La necesidad de acceso a la bitácora (log's) de la base de datos

Algunos DBMS utilizan los términos de respaldo en caliente (hot backup) y respaldo en frío (cold backup) para describir el acceso concurrente que puede ocurrir mientras los datos están siendo respaldados.

Un respaldo en frío siempre va acompañado del cierre de la base de datos, es decir , dejar la base de datos no disponible para todos los usuarios de manera intencional, para así posteriormente respaldar los archivos más importantes de la base de datos.

Un respaldo en caliente es realizado mientras la base de datos está en línea, es decir, que el acceso concurrente es posible. Dependiendo de las capacidades del DBMS en uso, los respaldos en caliente pueden ser problemáticos por las siguientes razones:

- Son más complejos de implementar
- Pueden provocar sobrecarga en las capacidades del CPU, incrementar las escrituras y lecturas a disco
- Requieren que el DBA con un conjunto de scripts específicos
- Requieren de muchas pruebas para asegurarse que los respaldos están bien hechos para un posterior proceso de recuperación de la base de datos

2.4.2 Recuperación

La recuperación es el procedimiento necesario para reestablecer la disponibilidad de los datos una vez que ocurre una falla que afecte la disponibilidad, para la recuperación de una

base de datos, es necesario contar con un respaldo de ésta. Es responsabilidad del DBA recuperar los datos lo más rápido posible, pues como mencionamos anteriormente la indisponibilidad de los datos puede afectar de manera económica o moral a la empresa, es decir, se puede dejar de ganar dinero o perderlo, o la empresa puede perder credibilidad por la ineficiencia de los servicios.

Una recuperación exitosa, es cuando el DBA recupera los datos a un estado o memento que él desea, y que cumple con los requerimientos de integridad de la base de datos, el estado deseado se refiere al el estado de los datos en el tiempo, por ejemplo: recuperar la base de datos justo antes de la falla, un día antes, de la semana pasada, etcétera. Si el DBA planeó una estrategia de respaldo adecuadamente y contó con todos los elementos necesarios para ello, podrá recuperar la base de datos hasta el momento que se requiera.

Pasos general para la recuperación

1. Identificar la falla.
2. Analizar la situación.
3. Determina que se requiere para la recuperación.
4. Identificar las dependencias entre los objetos de la base de datos a recuperar.
5. Localizar los respaldo requeridos para la recuperación.
6. Reconstruir el respaldo.
7. Procesar los archivos bitácora.

Tipos de recuperación

Los tipos de recuperación de la base de datos, se clasifican de acuerdo a la necesidad del momento en que se quiere recuperar los datos, es decir, se puede pedir que la base de datos sea recuperada hasta la semana pasada, hasta el día de ayer o justo antes del momento de la falla, también se puede requerir que se recupere una estructura de la base de datos, es decir, una tabla, un índice, un dato, un objeto, etcétera, es por ello que podemos clasificar la recuperación de la siguiente manera:

1. Recuperación hasta el momento de la falla. La Figura 2.6a recupera los datos hasta el memento justo antes de la falla.

2. Recuperación a un punto en el tiempo. La Figura 2.6a señala también cuando se recuperan los datos

3. Recuperación de una estructura de la base de datos (tabla, índice, tablespace, etcétera). La Figura 2.6b indica la posibilidad re recuperar un objeto o estructura de la base de datos.

4. Generar código SQL para deshacer transacciones equivocadas. La Figura 2.6b muestra este hecho.

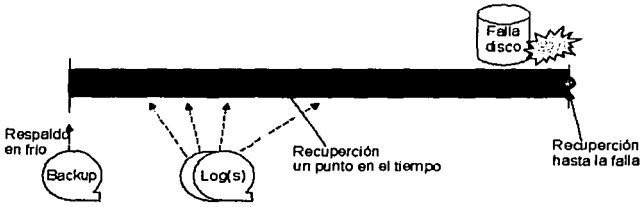


Figura 2.6a

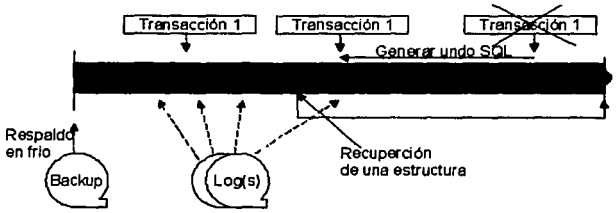


Figura 2.6b

2.5 Desempeño

Casi cualquier persona que ha tenido contacto con una computadora ha experimentado cierto tipo de rendimiento o desempeño, más aun los sistemas manejadores de bases de datos en ocasiones, durante en ciclo de vida de las aplicaciones tienen una reputación de un desempeño pobre, es por ello que el monitoreo y la afinación del rendimiento son aspectos que pretender prevenir la degradación del desempeño de las aplicaciones, y más aun, pretenden mejorar dicho desempeño.

2.5.1 Gestión del rendimiento

En todas las empresas se habla del rendimiento o desempeño de las aplicaciones, cuando se habla de monitorear o mejorar el desempeño o rendimiento de la base de datos, es hablar de un elemento de un conjunto, y no de todo el conjunto de elementos implicados en la o las aplicaciones, la Figura 2.7 muestra algunos elementos básicos que pudieran estar implicados en el desempeño de las aplicaciones como son: PC, red, servidor, sistema operativo, DBMS, y tabla.

Figura 2.7

Como lo muestra la Figura 2.7, son varios los elementos implicados en el desempeño de la base de datos, no significa que el DBA tenga que tener amplio conocimiento en todos y cada uno de estos elementos, sin embargo, se sugiere que el DBA tenga conocimientos básicos sobre ellos, ya que si el desempeño de una aplicación se ve afectado surgen los siguientes comentarios:

- El sistema está lento

- La base de datos tiene problemas o está lenta
- La red está lenta
- Se cayó el sistema

Estas son algunas frases que distintos tipos de usuarios pueden decir en el momento de la afectación al desempeño de la aplicación, pero ¿cuál de estas es realmente la situación que afecta el desempeño?, es por esta razón que el DBA debe tener conocimientos básicos sobre los posibles elementos que pueden afectar el desempeño de una aplicación, y un amplio conocimiento sobre el DBMS.

No existe una definición del desempeño de la base de datos, pero podemos decir que el término "desempeño de la base de datos" es la velocidad a la cual el DBMS responde a la demanda de información. Existen dos momentos para ver el desempeño de una base de datos, uno cuando éste se ve afectado negativamente y otro antes de que esto ocurra, lo que significa que hay dos maneras de ver el desempeño, una reactiva y otra proactiva, la primera es aquella que se realiza cuando el desempeño de la base de datos se ve afectado negativamente, y la segunda es aquella que realiza actividades antes de que esto suceda.

Podemos decir que son cinco los factores que intervienen en el desempeño de una base de datos:

1. Carga de trabajo. Es la combinación de las transacciones en línea, procesos en lote, consultas, análisis multidimensional, etcétera..
2. Recursos. Se refiere al hardware y software que interviene en el sistema.
3. Optimización. Se refiere a los elementos disponibles para optimizar la ruta de acceso a los datos, como son: instrucción SQL, parámetros del DBMS, forzar a usar índices, etcétera.
4. Contención. Es la condición donde dos o más componentes de la carga de trabajo están tratando de usar un mismo recurso.

La gestión del desempeño de la base de datos está compuesta por tres elementos: monitoreo, análisis y afinación.

- Monitoreo. Es el proceso de identificar posibles problemas y consiste en buscar dentro del ambiente general del DBMS y de las estructuras físicas un mal desempeño de las misas
- Análisis. Una ves que mediante el monitoreo se han identificado posibles problemas de desempeño, se realiza el análisis pertinente.
- Afinación. Una ves realizado el análisis y visto las posibles opciones de solución, se realizan los cambios adecuados para prevenir la degradación del desempeño de la base de datos.

2.5.2 Tipos de afinación

Una aplicación de bases de datos, requiere de constante interacción entre varios recursos computacionales, para poder operar eficientemente y de acuerdo a los requerimientos. Realmente la afinación de la base de datos puede ser vista como la afinación de tres elementos: afinación del sistema, afinación de la base de datos y afinación de aplicaciones. Entonces, al hablar del "desempeño de la base de datos", el DBA tiene que pensar en desempeño del sistema, desempeño de la base de datos y desempeño de las aplicaciones en la base de datos, la Figura 2.8 muestra la relación entre estos componentes.

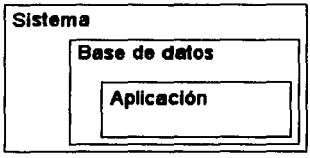


Figura 2.8 Elementos involucrados en la afinación

Desempeño del sistema

El "sistema" en este caso se refiere a los componentes de hardware y software requeridos por el DBMS para operar, y las aplicaciones para accesar a la base de datos, la Figura 2.9

muestra algunos des estos componentes. El DBA no tiene que ser un experto en todos estos componentes, pero si tiene que tener un conocimiento básico sobre ellos y conocer perfectamente la arquitectura del DBMS en la que se considere experto, la Figura 2.10 muestra la arquitectura de Oracle como ejemplo, también se sugiere tenga la autorización y los recursos para poder alterar algún componente del sistema o trabajar con equipos dentro de la organización para poder realizar los cambios necesarios al sistema.

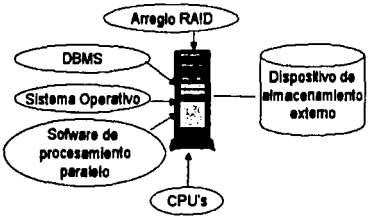


Figura 2.9 Componentes más comunes que se afinan

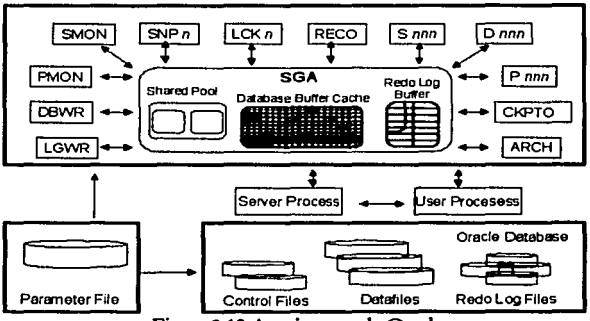


Figura 2.10 Arquitectura de Oracle

Un desempeño pobre del sistema puede degradar el desempeño de todas las bases de datos y aplicaciones desarrolladas en dicho sistema como lo muestra la Figura 2.9.

Desempeño de la base de datos

El desempeño de la base de datos se enfoca en afinar y optimizar el diseño, los parámetros y la construcción física de las estructuras de la base de datos, específicamente tablas e índices, y los archivos en los cuales se almacenan los datos. El administrador de la base de datos debe conocer las características del DBMS para aplicar las técnicas apropiadas para optimizar el desempeño de las estructuras de la base de datos, la mayoría de los mejores DBMS's, aunque posiblemente con diferente nombre (nos tomamos la libertad de no traducir el nombre), manejan las siguientes técnicas:

- Partitioning. Dividir una tala o un índice, y almacenar las divisiones en diferentes archivos.
- Indexing. Elegir los índices adecuados y opciones para permitir consultas eficientes.
- Denormalization. Cambiar el diseño lógico y la forma de almacenar los datos para mejorar el desempeño de las consultas.
- Block size. Determinar el tamaño correcto del bloque para hacer más eficiente la lectura y escritura de los datos.
- Free space. Determinar el espacio libre para el crecimiento de los datos.
- Reorganization. Remover ineficiencias de la base de datos reorganizando las estructuras de la base de datos.
- Clustering. Forzar la secuencia física de los datos en disco.
- Compression. Reducir automáticamente los requerimientos de almacenamiento.
- File placement and allocation. Colocar los archivos correctos en el lugar correcto.

Hablar a fondo cada una de estas técnicas sería muy extenso y es por ello que únicamente ahondaremos un poco en algunas de estas técnicas, para tener un mejor entendimiento sobre lo que relaciona con el desempeño de la base de datos.

Partitioning. Una tabla es una manifestación lógica de un conjunto de datos que están físicamente en algún medio de almacenamiento, y el administrador de la base de datos tiene que decidir la forma en la que cada tabla debe ser almacenada, cada DBMS proporciona

diferentes mecanismos para llevar acabo dicha tarea, algunas de las opciones podrían ser las siguientes:

- Una tabla en un solo archivo.
- Una tabla en varios archivos.
- Varias tablas en un sólo archivo.

La segunda opción es la que se utiliza para *particionar* la tabla, generalmente de usa para tablas índices de gran tamaño.

Indexing. Crear los índices adecuados de una tabla, es tal vez una de las técnicas más usadas y por consiguiente de alto riesgo, ya que la creación de un índice puede inclusive degradar el desempeño de la tabla. Los índices son comúnmente usados para:

- Localizar registros por valor(es) en una(s) columna(s).
- Hacer los joins más eficientes.
- Ordenar datos para satisfacer la consulta.
- Correlacionar datos entre las tablas.

Pero consideremos los siguientes casos para ejemplificar que no siempre un índice es bueno.

Cuando una tabla es pequeña realizar consulta utilizando el índice pude ser más costoso que un *full table scan*(recorrido de toda la tabla), pues acceder al índice requiere también de lecturas a disco.

Si una consulta sobre una tabla utiliza la cláusula WHERE, entonces siempre se realizará una búsqueda sobre la tabla y no se usará el índice.

Si los datos de la tabla sobre la que sé esta indexando tiene muchas operaciones de actualización, el desempeño de estas se degrada pues también se requiere actualizar los índices.

Denormalization. Desnormalizar es lo opuesto a normalizar, en otras palabras es el proceso de colocar un mismo datos en diferentes estructuras, este proceso incrementa claramente el espacio en disco pero se pretende mejorar el desempeño de las consultas. Algunas de las razones por las que podemos contemplar la desnormalización son:

- Cuando el costo de join es alto.
- Cuando reportes importantes son costosos en tiempo para su generación.
- Cuando tablas son requeridas concurrentemente por dos o más ambientes.
- Para consolidar relaciones uno a uno, o uno a muchos en una sola tabla.
- Para tomar ventaja de las características del DBMS.

Block size. Independiente del sistema operativo que tiene especificado un tamaño de bloque(normalmente 2Kb), los DBMS más robustos tienen la habilidad de especificar un tamaño del bloque, este puede ser de 2, 4, 8, 16, 32, 64Kb, la selección del tamaño del bloque depende de las características de las aplicaciones que usen dicha base, básicamente podemos hablas de dos tipos de ampliaciones: aplicaciones transaccionales (OLTP Online Transaction Processing) y aplicaciones de apoyo a la toma de decisiones (DSS Decision Support System), los sistemas OLTP requieren que el tamaño del bloque sea pequeño, ya que normalmente son transacciones simples, y los sistemas DSS requieren que el tamaño del bloque sea grande para evitar la búsqueda de muchos bloques.

Free space. El espacio libre se refiere al espacio vacío y disponible para agregar nuevos datos, la especificación de espacio libre reduce la frecuencia de reorganización y contención, e incrementará la eficiencia de la operación de inserción de nuevos registros. Un parámetro típico de los DBMS's es el PCTFREE, el cual permite especificar el porcentaje de espacio libre en cada bloque, disponible para las actualizaciones de los registros ya existentes, de esta manera evitar la migración de registros es decir que un registro este en más de un bloque, en general especificar de manera adecuada el espacio libre tiene las siguientes ventajas:

- La actualización e inserción de registros es más rápida.
- Se evita la migración de registros.
- La reorganización es menos frecuente.

Y también consideremos las siguientes desventajas:

- Se requiere de mayor capacidad de almacenamiento, es decir las estructuras requieren más espacio.
- La búsqueda de registros revisando la tabla (scan table) toma más tiempo, pues menos registros en un bloque requiere más operaciones de lectura. Para el caso de sistemas DSS donde la actualización de registros es mínima o cero, el parámetro de PCTFREE se asigna a cero, por obvias razones.

Desempeño de aplicación

El desempeño de aplicación se refiere a el o los lenguajes relacionados directamente al DBMS como pueden ser SQL, PLSQL, pg_sql, etcétera, poniendo particular interés en SQL ya que se ha visto que más de 50% de los problemas de desempeño de la base de datos son causados por código ineficiente Algunos puntos clave que se pueden analizar cuando el desempeño de la aplicación se ve afectada son:

- Es el correcto el tipo de SQL (planeado o no planeado, dinámico o estático) para el tipo de aplicación en cuestión.
- Lenguaje de programación. Verificar si el lenguaje de programación es capaz de lograr el desempeño requerido, y si el ambiente del lenguaje es optimizado para el acceso a la base de datos.
- Diseño y procesamiento de transacciones. Verificar si las transacciones dentro del programa son diseñadas adecuadamente para asegurar las propiedades de una transacción (atomicidad, consistencia, aislamiento y durabilidad).
- Estrategia de bloqueo. Verificar si la aplicación implementa correctamente los bloqueos de datos y por cuanto tiempo.
- Estrategia de COMMIT. Verificar si la aplicación optimiza de comando COMMIT, de manera que se beneficie la escritura a disco y el bloqueo de los datos no perjudique.

Además de los aspectos antes mencionados y de acuerdo a las características de cada DBMS, se puede considerar el ejecutar estadísticas para ayudar a determinar la mejor ruta de acceso a los datos, análisis de la sentencia de consulta, lanzar consultas en paralelo, etcétera.

2.6 Disponibilidad

La disponibilidad es el punto central para los administradores de la base de datos, pues en él convergen tópicos como seguridad, respaldo y recuperación, rendimiento, etcétera. Si los datos no están disponibles, las aplicaciones no podrán operar, si las aplicaciones no pueden operarse , la empresa o institución no podrá vender productos o proveer servicios, es por ello que asegurar la disponibilidad de los datos es la tarea más importante del DBA, este deberá entonces hacer todo lo posible para que los datos estén disponibles.

El requerimiento de los datos varía de acuerdo a las necesidades de cada área dentro de la empresa, así como de empresa en empresa, por ejemplo se tiene una base de datos con la información acerca de la población total en cada localidad y se está realizando un trabajo de estadísticas de ésta población y otras variables económicas. Este trabajo tiene una duración de tres meses, entonces, si en cierto día no se dispone de los datos, no es "grave". Otro ejemplo sería el de un banco, donde si en un día de actividades normal, los datos no estuvieran disponibles, las repercusiones serían graves.

En la actualidad, con el desarrollo tecnológico, el importante crecimiento y el uso de Internet, permiten que las empresas aspiren a una alta o muy alta disponibilidad de los datos, es decir, actualmente se piensa en aplicaciones que hagan uso de una base de datos disponible las 24 horas de día, los 365 días del año. Los ejemplos anteriores pretenden introducir al concepto de disponibilidad, para formalizarlo daremos la siguiente definición:

> Disponibilidad: es la condición donde dado un recurso, este puede ser accesado por sus consumidores. Esto significa que si una base de datos está disponible los usuarios de sus datos, pueden acceder a ella. Cualquier condición que haga inaccesible al recurso, provoca lo opuesto a disponible, es decir no disponible.

Una pregunta que puede surgir en este momento es: si se hace un acceso a la base de datos requiriendo de un dato, y esta tarda una hora o un día en proporcionarlo, ¿ la base de datos está disponible o no?

Disponibilidad o rendimiento de la base de datos son dos términos que a veces se confunden, pues existen similitudes entre ellos. La diferencia consiste en la disponibilidad del usuario para acceder a la base de datos, es posible acceder a una base de datos con un rendimiento pobre, pero no es posible acceder a una base de datos que no está disponible. Sin embargo, si el rendimiento es tan pobre, que los usuarios de la base de datos no pueden realizar su trabajo, se dice que la base de datos no está disponible, es decir que un problema de desempeño puede llegar a ser un problema de disponibilidad de la base de datos.

La disponibilidad comprende cuatro componentes que en combinación aseguran el uso de la base de datos.

- Manejabilidad: es la habilidad de crear y mantener un ambiente efectivo para brindar servicio a los usuarios de la base de datos.
- Recuperabilidad: es la habilidad de restablecer el servicio si ocurre u error o la falta de algún componente.
- Fiabilidad: es la habilidad de brindar servicio de la base de datos a niveles específicos para un período específico.
- Utilidad: es la habilidad de determinar la existencia de problemas, diagnosticar sus causas y reparar los mismos.

Capítulo 3

PROBLEMAS EN TORNO A LA ADMINISTRACIÓN DE LA BASE DE DATOS, SUS CAUSAS Y ALGUNAS SUGERENCIAS

El presente capítulo, presenta algunos problemas en torno a la administración de una base de datos, están basados en mi experiencia como Administrador de Bases de Datos y Subdirector de Desarrollo de Sistemas en la "Semarnat" Secretaría de Medio Ambiente y Recursos Naturales, puestos que desempeñé en dicha secretará por un periodo de 3 y 4 años respectivamente durante 1998-2002. Cabe aclarar que el puesto de DBA lo desempeñé a la par del puesto de subdirector.

La clase de problemas que se exponen en este capítulo, no son exclusivos de la Semarnat, pues como Subdirector tuve la oportunidad de participar en proyectos que el gobierno federal pretendió implementar en todas las dependencias de éste, y dichos problemas se presentaban en otras dependencias del gobierno, es posible que algunos de estos problemas se presenten en otras instituciones y empresas privadas, en las que hagan uso de algún DBMS, aunque no me atrevo a afirmarlo.

No pretendo cubrir todos los problemas en torno a la administración de la base de datos, ni dar la receta para el bueno manejo en torno a las bases de datos, simplemente exponer algunos problemas, algunas causas y algunas sugerencias para atacar dichos problemas, para llevar a cabo esto agrupamos los problemas en tres diferentes temas:

- Estándares
- Políticas y procedimientos
- Estructura organizacional

Los cuales consideramos son importantes aspectos que toda institución o empresa que haga uso de un DBMS debe considerar.

Cabe aclarar que algunos de los ejemplos se presentan en más de uno de los temas antes mencionados, pues en ocasiones, estos están presentes o son problemas que pertenecen a más de un tema.

3.1 Problemas en torno a estándares

En el desarrollo de software, que es donde el uso de un DBMS se ve implicado y por tanto la administración del mismo, la falta de estándares provoca la falta de comunicación, es por ello que es muy importante implementar estándares, y en lo que respecta a la administración de la base de datos dos estándares son importantes:

- Estándar para el modelado de la base de datos
- Estándar para el nombrado de objetos en la base de datos

3.1.1 Estándar para el modelado de la base de datos

El estándar para el modelado de la base de datos permite, como mencionamos en el Capítulo 2, analizar las cosas que le interesan a la empresa, y cómo éstas están relacionadas unas a otras, y es el modelo entidad-relación el estándar que normalmente se aplica. El DBA tiene la obligación de saber realizar un modelo para una base de datos o poder interpretar uno para la construcción de la misma, sin embargo algunos problemas pueden surgir como son los siguientes:

Si el DBA no realiza el modelo entidad-relación

> El DBA puede no realizar el modelo entidad-relación, para la realización del modelo entidad-relación se necesita el análisis de datos de la empresa y como estos están relacionados, lo que implica que el encargado de dicho análisis debe saber perfectamente la teoría del modelo entidad-relación, si el encargado de realizar el análisis de los datos no posee los conocimientos y la experiencia para realizar el análisis o el modelo entidad-relación, la construcción de la base de datos será

errónea, es decir la implementación física de los datos no será la correcta y el software o el desarrollo del software se verá afectado negativamente.

Si el DBA realiza el modelo entidad-relación

Si el DBA realiza el modelo entidad-relación implica que debe de tener un insumo para realizar el análisis de los datos de la empresa, como son entrevistas con los usuarios para la recolección de requerimientos o algún modelo que represente los requerimientos de los datos. Esta tarea exige que el DBA ocupe tiempo para poder realizar el análisis de los datos, si la institución o empresa consta de diferentes áreas que requieren el desarrollo de software, es decir que el DBMS albergará varias y diferentes bases de datos y el DBA tiene la obligación de realizar el modelo entidad-relación de cada una de estas bases, llevaría a que el DBA no tendría tiempo para realizar las funciones que le conciernen como son respaldos, actualizaciones, monitoreo, afinación, entre otras tareas..

Causas

La falta de dominio del modelo entidad-relación, puede llevar a la implementación incorrecta de la base de datos.

La pérdida de información desde el levantamiento de los requerimientos de los datos hasta la implementación física de los mismos ("teléfono descompuesto"), provoca que el DBA tenga problemas por la incongruencia de la integridad semántica de los datos.

No se hace uso dentro de la empresa o institución de otros estándares que permitan la comunicación entre los procesos del desarrollo del software, que llevan a la creación de un modelo entidad relación y que respalden la creación de este.

La carga de trabajo de análisis de los datos de una empresa o institución para la realización de modelo entidad-relación en el DBA, repercute en el desempeño del mismo en el manejo o dominio de las características del DBMS

Sugerencias

Se sugiere la especificación de un grupo o persona como administrador de datos (DA del inglés data administrator) y de un grupo o persona como administrador de la base de datos(DBA), para definir actividades distintas en los aspectos del proceso de los datos del negocio, y los aspectos técnicos de los datos. Los aspectos del proceso de los datos están relacionados con la administración de los datos, mientras que los aspectos técnicos están relacionados con la administración de la base de datos.

Administración de datos

La administración de datos separa los aspectos del manejo de los datos de los aspectos tecnológicos usados para el manejo de los mismos. El administrador de datos es responsable de entender las reglas del negocio, es decir la definición de los procesos y traducirlos en un modelo de datos lógico. En general el DA está involucrado en las siguientes fases de ciclo de vida del desarrollo de un sistema o aplicación:

- Definición de requerimientos
- Análisis
- Diseño

Mientras que el administrador de la base de datos se involucra en el diseño, desarrollo, pruebas e implantación, como podemos ver. El DA es entonces responsable de las siguientes tareas:

- Identificar y catalogar los requerimientos de los datos definidos por los usuarios.
- Producir los modelos de datos lógico y conceptual para representar exactamente la relación entre los elementos para los procesos de la dependencia.
- Crear un modelo de datos de la dependencia que incorpore todos los datos usados por todos los procesos de la misma.
- Establecer las políticas de los datos para la dependencia

- Identificar los dueños y administradores de los datos.
- Establecer estándares para el control del uso de los datos

Administración de bases de datos

A la administración de la base de datos le concierne básicamente, el asegurar que la información precisa y consistente esté disponible para los usuarios y las aplicaciones, cuando sea requerida, de este modo en la administración de la base de datos se interactúa tanto con las aplicaciones como con los usuarios. La primera obligación del grupo de administración de bases de datos es entender el modelo de datos construido por el administrador de datos y comunicarlo a los desarrolladores. El modelo de datos lógico es el mapa que el DBA usará para crear la base de datos física, es decir, que el DBA se encarga de transformar el modelo de datos lógico en un diseño de base de datos físico eficiente. En todo este proceso de comunicación del proceso de los datos es importante hacer uso de los estándares necesarios como UML y otros, para así evitar la pérdida de información.

En la administración de la base de datos como ya mencionamos, se tienen funciones más técnicas, y algunas de las más importantes son las siguientes:

- Diseñar la BD
- Asegurar la integridad de la BD
- Asegurar la disponibilidad de los datos
- Seguridad de la BD
- Respaldo y Recuperación de la BD
- Monitoreo del rendimiento y afinación de la BD

Asegurar la usabilidad y disponibilidad de los datos de la institución requiere de una variedad de tareas del trabajo conjunto entre el DA y el DBA en diferentes áreas o etapas del desarrollo de un software, la Figura 4.1 muestra la relación del DA y el DBA, y muestra las responsabilidades de cada uno de estos grupos (DA y DBA), así como la intersección de colaboración de ambos.

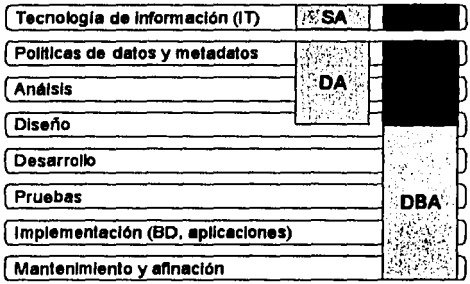


Figura 3.1

El diseño de un sistema o aplicación, es la fase donde el DA y el DBA trabajan juntos para llevar un diseño lógico de datos a un diseño físico

Cabe aclarar que en un área de sistemas o de desarrollo de software, no son únicamente los grupos de DA y DBA los que intervienen en el ciclo de desarrollo, pueden existir otros como el grupo de calidad, planeación, etcétera.

3.1.2 Estándar para el nombrado de objetos en la base de datos

El nombrado de objetos también es otro problema, en algunas empresas o instituciones la creación de bases de datos u objetos de la base de datos es una responsabilidad descentralizada, es decir, que no sólo el DBA crea objetos en la base de datos. Cuando no existe un estándar para la creación de objetos en la base de datos, se pueden presentar problemas de:

- Incompatibilidad de nombres
- Mala documentación en las bases de datos
- Retraso en la creación de sentencias SQL por inconsistencia en el nombrado de los objetos

El siguiente ejemplo es muestra de ello, supongamos que utilizamos Oracle como DBMS.

```
Create table estados                create table mun
  (cve_edo     number,                  (id          char(5),
   estad       varchar2(100));           municipio   varchar2(100),
                                         id_edo      number
                                         parti_pol   varchar(100));
```

Donde utiliza:

- *cve_edo* y *id* en el mismo contexto
- *cve_edo* y *id_edo* como cosas distintas al referirse a la llave primaria y foránea
- *estad* esta incompleto y dificultará su uso
- *mun* para nombrar la tabla de municipios
- *parti_pol* está incompleto y dificultará su uso (se refiere a partido político)

El Anexo A es una sugerencia de estándar para el nombrado de objetos en una base de datos. Cabe mencionar que existe una amplia gama de posibilidades que se pueden adoptar para el nombrado de objetos de una base de datos, es tal vez por esta razón, que no existe un estándar oficial para el nombrado de objetos, pues en ocasiones depende de las políticas de la empresa o institución, del estilo del grupo de desarrollo de software, de las características del DBMS o de algún otro elemento.

Es también tal vez por esa gran gama de posibilidades, que en muchos grupos de desarrollo de software, donde una base de datos esta presente, no exista un estándar para el nombrado de objetos de una base de datos, sin mencionar también los diferentes estilos de los programadores de sentencias SQL, que en ocasiones dificulta la lectura y mantenimiento de las aplicaciones. Como los ejemplos lo muestran, es importante contemplar el uso de un estándar para el nombrado de objetos en una base de datos en un grupo de desarrollo de software.

3.2 Políticas y procedimientos

La falta de políticas y procedimientos provoca la existencia de problemas de coordinación y definición de responsabilidades en torno a la administración de bases de datos. Algunos de

los problemas que se pueden presentar por la falta de políticas y procedimientos se describen a continuación.

3.2.1 Políticas

La descentralización de funciones del DBA es uno de los problemas más comunes, la descentralización de funciones se refiere a la distribución de las tareas y responsabilidades de la administración de la base de datos. Ejemplos de descentralización de funciones y tareas pueden ser:

- Crear/mantener cuentas de usuarios con los permisos para habilitar y deshabilitar los servicios del DBMS.
- Crear/mantener cuentas con los permisos para poder realizar respaldos a las base de datos.
- Crear/mantener cuentas con los permisos de acceso para la creación de objetos en la base de datos.
- Implementar auditorias.

Y algunos de los problemas que se pueden presentar por causa de la descentralización son los siguientes:

- Si no se realizan los respaldos o se asegura la integridad de los datos antes de deshabilitar los servicios de una base de datos de manera intencional y habilitarlos posteriormente, puede provocar la pérdida de datos.
- Se pueden crear objetos con el uso indebido de los tipos de los datos debido al desconocimientos de las características del DBMS.
- Se pueden implementar auditorias con programación cuando el DBMS tiene características para poder auditar. La manera de implementar la auditoria puede repercutir en el rendimiento de la base de datos, sin dar todos los detalles teóricos a continuación damos un ejemplo, supongamos dos casos:

1. Se agregan columnas a las tablas para registrar en dichas columnas fecha de actualización y el usuario que la realiza.
2. Se implementa un trigger para que grabe en otra tabla la fecha de actualización y el usuario que realiza la operación.

Un bloque es la unidad mínima física de almacenamiento, dependiendo del tamaño del bloque, éste puede almacenar uno o más registros, inclusive puede necesitar de más de un bloque para almacenar un registro. La Figura 3.2 ilustra un bloque.

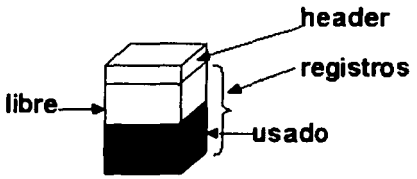


Figura 3.2 Bloque de la base de datos

Supongamos que implementamos la auditoria como en el primer caso y que realizamos una consulta que obtiene un cierto número de registros, entonces tenemos que:

La consulta puede ser más lenta, pues puede requerir de un mayor número de bloques que cumplen la condición de la consulta, pues mientras mayor sea el tamaño del registro, éste requerirá de mayor espacio para almacenarse y por consiguiente de un mayor número de bloques.

- Se pueden crear objetos de la base de datos sin apego a un estándar de nombres.

```
Create table estados
  (cve_edo      number,
  estad         varchar2(100));
```

```
create table mun
  (id           char(5),
   municipio    varchar2(100),
   id_edo       number
   fecha_creacion     date
   hora_creacion      varchar2(10));
```

En este ejemplo:

- No se respeta un estándar para el nombrado de la base de datos
- El atributo hora_creacion tiene un tipo de dato incorrecto
- Se pretende implementar auditoria de las inserciones de los registros agregando atributos a la tabla, cuando usar triggers es una mejor opción

Causas

Sin las políticas necesarias para la descentralización de funciones del DBA, vemos claramente que las responsabilidades no estarán bien definidas y esto provocará problemas en la administración de las bases de datos.

3.2.2 Procedimientos

Los problemas debido a la falta de procedimientos, se refieren a la falta de automatización y documentación de los procedimientos que requiere la base de datos, así como de los procedimientos necesarios para los casos que el servicio o disponibilidad de la base de datos es afectada de manera inesperada o esperada, algunos de los procedimientos son:

- Actualizaciones del DBMS
- Migración de bases de datos de una versión a otra del DBMS, o de un DBMS a otro
- Respaldos a las bases de datos
- Recuperación de las bases de datos

La automatización de los procedimientos antes mencionados es responsabilidad del DBA y por consiguiente los problemas en torno a ellos son también su responsabilidad. Algunos ejemplos de los problemas que se pueden ocasionar por la falta de procedimientos son:

- Si se pierden datos, ¿de quién es la responsabilidad?

  Suele pasar que un usuario final llame al encargado del sistema para decirle: "Yo capture algunos datos ayer y no aparecen" o "Yo realice algunos cambios a la

información y no aparecen". En la actualidad los sistemas manejadores de base de datos son estables en cuanto a integridad de datos se refiere, por lo que en estos casos la perdida de datos puede ser ocasionada por algún usuario, por el programa o por el DBA. Es el DBA quien tiene la responsabilidad de deslindar responsabilidades sobre los datos en la base de datos, pero también es responsable de investigar anomalías como la pérdida de datos.

- Si existe una falla de hardware o de sistema operativo ¿cómo y quién se encarga de recuperar la base de datos?

  Es responsabilidad del administrador del equipo el reportar la falla y hacerse cargo hasta que el equipo sea funcional nuevamente, sin embargo es responsabilidad del DBA el tener un respaldo de los datos para restablecer la base de datos. En muchos casos el responsable del equipo es el DBA, pues un servidor en el que reside una base de datos es casi es el único uso del mismo.

- Si existe una falla de software, es decir, del sistema manejador de bases de datos ¿cómo y quién se encarga de recuperar la base de datos?

  Es responsabilidad del DBA el tener un respaldo de la base de datos para poder recupere los datos y también es su responsabilidad restablecer todas las funcionalidades del DBMS, es por ello muy importante que el DBA posea los conocimientos sobre uso y administración del DBMS.

- Si se pierde el centro de datos ¿De quién es la responsabilidad de la pérdida de datos?

  En el caso de la pérdida del centro de datos, generalmente no es responsabilidad de alguien en particular, la perdida del centro de datos generalmente se debe a fenómenos naturales como inundaciones, terremotos, etcétera.. Existe la posibilidad de implementar un ambiente de replicación de bases de datos, lo que significa que en caso de la pérdida del centro de datos por fenómenos naturales, falla eléctrica o

algún otro elemento, la base de datos y la información no se pierde, de hecho las aplicaciones siguen funcionando pues se comunican con la base de datos replicada. Esta arquitectura de replicación de bases de datos es muy costosa en muchos sentidos, económica, rendimiento de aplicaciones, administración, etcétera. Sin embargo puede evitar problemas como la "caída de un sistema en plenas elecciones electorales para presidente".

- Si la ejecución de una aplicación es lenta ¿dónde está el problema o cuál es la solución?

  Es muy común escuchar la siguiente frase: "El sistema está lentísimo". Esta situación puede contener en si misma deferentes problemas, los más importantes pueden ser: Degradación del rendimiento de la base de datos, problemas de la red y problemas en el programa desarrollado. La degradación del rendimiento de la base de datos es el caso que a nosotros nos interesa y es responsabilidad del DBA analizar los factores que ocasionan dicho estado de una base de datos, aunque también el DBA debe saber cuando no es un problema de la base de datos o del sistema manejador

- Si el programa deja de funcionar a causa de un incremento excesivo de datos ¿de quién es la responsabilidad?

  La planeación del incremento de los datos, así como el nivel transaccional, es decir número de operaciones diarias y concurrencia de transacciones debe ser responsabilidad del diseñador de la base de datos y el DBA es responsable de implementar y prepara físicamente a la base de datos para dicha planeación. Los aspectos de volumen e incremento de datos, así como el nivel transaccional, son elementos importantes en la decisión de la selección del DBMS.

Causas

No existen los procedimientos necesarios para la recuperación de la base de datos, más general no existen los procedimientos ni la documentación de los mismos para los temas

como son: migración, actualización de DBMS, respaldos, recuperación y planeación del crecimiento de las bases de datos.

El Anexo B es una guía para la construcción de un plan de contingencia, que podemos decir es la documentación de los temas de respaldo y recuperación, describe cómo el DBA puede definir y documentar los procedimientos para la recuperación de las bases de datos y los mecanismos de respaldo necesarios.

3.3 Estructura organizacional.

En una empresa dedicada al software como en cualquier otra, la estructura organizacional es un elemento importante para el funcionamiento correcto del área o empresa. La organización es el proceso de coordinar personas y otros recursos para trabajar juntos, a fin de lograr un objetivo, la estructura organizacional es el sistema de tareas, relaciones jerárquicas y canales de comunicación que vinculan el trabajo de todos los individuos y grupos en la organización. La representación coloquial de la estructura organizacional es mediante el organigrama, que describe gráficamente la disposición formal de los puestos de trabajo dentro de la organización.

Los problemas en torno al DBA que se pueden presentar en una estructura organizacional son muy diversos, y pueden ser motivados por diferentes razones como son: jerarquía, personalidad, aspectos culturales, etcétera. Es claro que no haya una receta de cocina de la estructura organizacional que un área o empresa dedicada o involucrada en el desarrollo de software debe tener, y por consiguiente del lugar exacto que el DBA debe ocupar dentro del organigrama.

Mediante la estructura organizacional podemos conocer:

- La división de trabajo. Puestos y títulos muestran las responsabilidades de trabajo
- Relaciones de supervisión. Las líneas muestran quien es responsable ante quien
- Canales de comunicación. Las líneas muestran los flujos formales de comunicación.

- Subunidades principales. Se muestran los puestos que son responsables ante un gerente o coordinador común.
- Niveles de administración. Se muestran los niveles verticales de administración.

Probablemente, el DBA tenga complicaciones en torno a cualquier estructura organizacional, claro que los problemas que pueda tener en torno a la estructura, dependen del lugar que ocupe el DBA dentro del organigrama. A continuación hablaremos de los problemas del DBA en torno a algunas de las estructuras organizacionales más importantes, haciendo énfasis en la estructura burocrática, que sobre la cual tengo mayor experiencia.

3.3.1 Estructura funcional

Estructura funcional: es aquella donde se agrupa a las personas que poseen habilidades similares, y que ejecutan tareas parecidas, la Figura 3.3 es un ejemplo de ello.

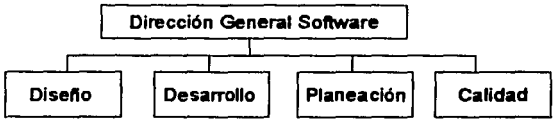


Figura 3.3 Estructura funcional

Algunos problemas en torno al DBA en esta estructura son:

- Puede incurrir en la pérdida de responsabilidades por exceso de reuniones con las diferentes áreas.
- Existen o se crean barreras funcionales, que consisten en la falta de comunicación y coordinación entre las diferentes áreas.

3.3.2 Estructura por proyecto o equipo

Estructura por proyecto: se agrupa a las personas por proyecto o equipo, la Figura 3.4 muestra un ejemplo.

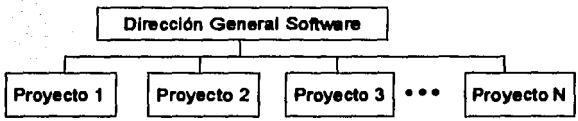


Figura 3.4 Estructura por proyecto

En esta estructura claramente cada proyecto requiere de un DBA, lo que resultaría excesivamente costoso, y en el supuesto de que fuera uno o dos DBA's para todos los proyectos, algunos de los siguientes problemas pueden surgir:

- Duplica recursos, para el caso de DBA's, no es lo mismo pagar el salario de un DBA que de varios.
- Si los proyectos comparten el DBMS, la administración se descentraliza y se vuelve más complicada de coordinar y la definición de responsabilidades de los administradores puede ser incongruente, es decir, si un error ocurre con los datos o con el DBMS, ¿la responsabilidad es de varios DBA's o de ninguno?

3.3.3 Estructura burocrática

La estructura burocrática: se basa en la lógica, el orden y el uso legítimo de la autoridad formal. El sociólogo Max Weber fue quien originalmente la describió como el "tipo ideal" de organización, pues consideraba que es una forma ordenada, justa y sumamente eficiente.

Las características de la estructura burocrática son:

- clara división de trabajo;
- jerarquía estricta de autoridad;
- reglas y procedimientos formales.

Para ejemplificar la estructura burocrática, nos permitiremos tomar el caso práctico del organigrama de la Secretaria de Medio Ambiente y Recursos Naturales "Semarnat", en la Dirección General de Informática y Telecomunicaciones "DGIT".

Primeramente citamos algunas de las atribuciones de la DGIT:

- Establecer lineamientos generales en materia de informática.
- Proponer a las unidades administrativas competentes de la Secretaría, sistemas, procedimientos y estrategias en materia de tecnología informática.
- Dictaminar la adquisición, instalación, operación y mantenimiento de los equipos informáticos y de telecomunicaciones, y sobre la contratación de servicios, incluyendo sus programas, licenciamiento, servicios, en Internet, equipos auxiliares y de transmisión.
- Desarrollar e instrumentar en colaboración con las unidades administrativas, los controles sobre la información derivada de los procesos informáticos desarrollados directamente por la Secretaría o por terceros.
- Diseñar y programar el modelo único de la base de datos de los sistemas de información de la Secretaría.
- Implantar políticas y programas de seguridad en los sistemas informáticos y de telecomunicaciones, y en los sitios de Internet e intranet de la Secretaría.
- Instrumentar y administrar los medios y aplicaciones para la publicación de información vía Internet, intranet, extranet y medios tradicionales, así como las interfases y aplicaciones necesarias para integrar los sistemas de la Secretaría a los sistemas intergubernamentales e interestatales, según sea requerido.

En la Semarnat y cualquier otra secretaría de estado (al menos en nuestro país) así como sus áreas o subunidades, tiene una estructura burocrática. Las Figuras 3.5a y 3.5b muestran los organigramas de la Semarnat y la DGIT respectivamente, en el organigrama de la Semarnat se encierra en un circulo la DGEI, en el organigrama de la DGIT se omite la estructura de la dirección de infraestructura tecnológica y los números indican el número de jefes de departamento que tiene una subdirección, es decir el número de subordinados que a cargo

de una subdirección en específico, de esta manera podemos decir que una subdirección cuenta con un subdirector y algunos jefes de departamento, como se ilustra en la Figura 35b.

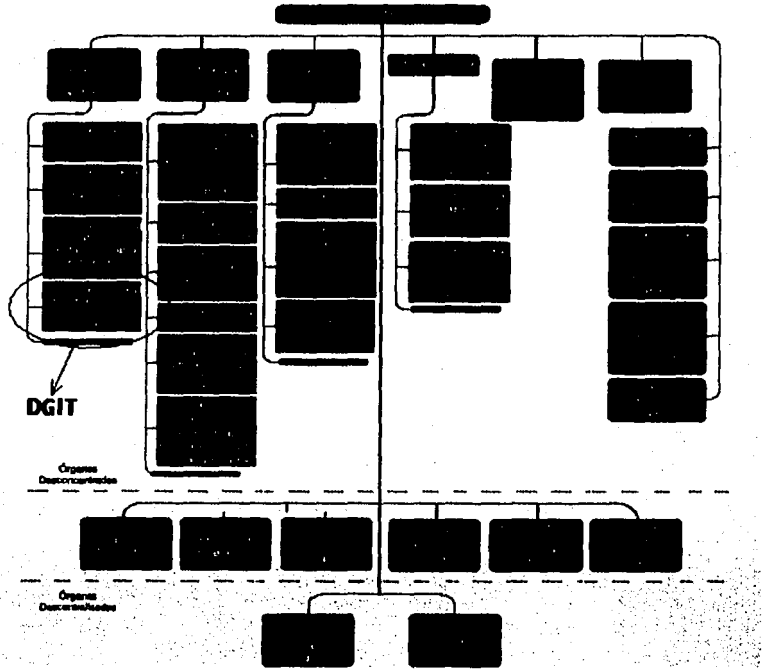


Figura 3.5a Organigrama de la Semarnat

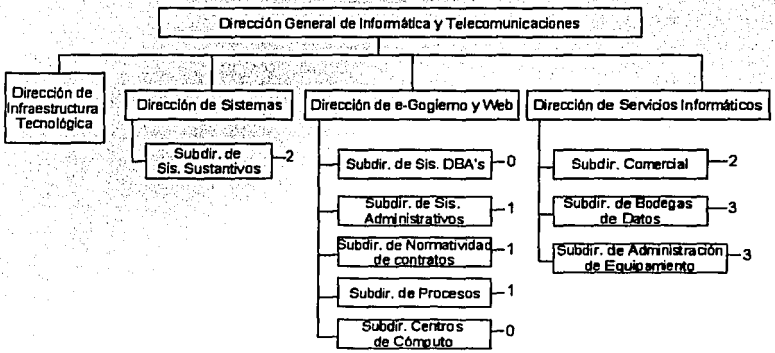


Figura 3.5b Organigrama de DGIT

Para el caso de la Semarnat, durante 1997-2001, el DBA se enfrentó a muchos problemas en torno a la estructura burocrática, sin pretender abarcar todos los problemas generados en torno a ésta estructura pues éstos pueden ser de diferente naturaleza, trataremos de mencionar algunos que a nuestro juicio se presentan en empresas con estructura burocrática.

1. El levantamiento de requerimientos es realizado por una persona que no va a realizar el análisis de los mismo, no existe la figura del administrador de datos y DBA no realiza el levantamiento, ni el análisis de los datos, esto es debido a cuestiones "políticas" o de estructura, realmente lo que pasa es que el responsable del área de desarrollo de sistemas, como prestador de servicios dentro de la misma institución, pretende asistir a todas las reuniones posibles, con la intención de ser más conocido, y de esta manera buscar relaciones para obtener mejores puestos, descuidando el levantamiento y análisis de requerimientos.

2. No existen reuniones con expertos, sobre las decisiones del diseño y arquitectura del sistema, en general sólo una persona toma estas decisiones, así como la de la selección de herramientas de desarrollo y DBMS a usar, lo que implica que el DBA no es quien toma la decisión sobre el DBMS a utilizar.

3. La selección del DBMS es importante, así como el esquema de licenciamiento y soporte por parte del fabricante, son aspectos importantes a considerar en el desarrollo de sistemas, y no es el DBA la persona a la que se consulta cuando se programa y planea el presupuesto para dichos asuntos, mucho menos para hablar de las aplicaciones que se implementarán durante el año, para calcular el requerimiento de memoria y disco duro necesario, o del crecimiento de usuarios y datos.

4. El DBA está, dentro del organigrama como lo muestra la Figura 3.4b, en una posición que bajo el diseño de la estructura burocrática, pertenece a una pequeña área, lo que disminuye el flujo de trabajo y coordinación con las otras subunidades, el DBA no tiene jefes de departamento lo que implica que si el DBA se va de vacaciones y ocurre un error deje indisponible la base de datos, el área se vuelve un caos y el DBA se siente indispensable, por no decir "todo poderoso".

5. El aspecto del diseño, manejo y prueba de flujo de datos, así como nivel transaccional de la aplicación, son algunos de los aspectos que se deben de considerar en el desarrollo de sistemas o en la contratación de servicios. Es el DBA la persona con los conocimientos técnicos suficientes para colaborar en dicha tarea, no ocurriendo esto de esta manera a causa de verticalidad de las estructuras burocráticas; esto es, las áreas pueden contratar los servicios de proveedores y no existe la coordinación para llevar el proyecto a un término satisfactorio, ésta coordinación la podemos también adjudicar a la falta de un marco de trabajo común para el área de software.

6. El DBA no recibe diseños que le permitan integrar el modelo único de datos, de hecho no existe un área que integre los diferentes modelos de los diferentes procesos y trámites de la Semarnat, en ocasiones el DBA sólo se remite a crear cuentas de acceso para los consultores externos sin la coordinación y colaboración entre el diseño y construcción de una base de datos.

Causas

Como anteriormente mencionamos, los problemas del DBA en torno a la estructura organizacional pueden provenir de diferentes orígenes, y es claro que el organigrama de una empresa es de acuerdo a las características de la misma, sin embargo, tomando los casos anteriormente descritos, los problemas mencionados se presentan por las características de la estructura organizacional:

> La estructura funcional divide claramente las funciones permitiendo tener áreas específicas, sin embargo no es claro en qué área el DBA debe de estar, o si éste pertenece a todas las áreas.
>
> La estructura por proyectos o equipos define muy bien los equipos de trabajo por proyecto, pero cada equipo requiere de un DBA, aunque este puede ser el mismo para diferentes proyectos, y esto puede resultar costoso, otra desventaja de esta estructura es que no integra el diseño de todos los proyectos de la empresa, pues sólo le interesa el diseño del proyecto, no siendo de esta manera en la estructura funcional, donde el área de diseño esta encargada de realizar el diseño de los proyectos considerando y construyendo al mismo tiempo el diseño general de los datos de la empresa en su totalidad.
>
> Para el caso de la estructura burocrática se complica aún más, pues en este país la palabra "burocracia" tiene un significado negativo y hasta despectivo, cuando burocracia significa: aparato de gobierno, y burócrata significa: empleado de gobierno

Muchas pueden ser las causas de los problemas del DBA en la estructura organizacional, y las soluciones a ellos dependen y son particulares de cada institución o empresa, considerando que una institución o empresa requiere de un área de sistemas o de desarrollo de software, mencionamos a continuación algunas sugerencias que se pueden tomar en cuenta para definir la estructura organizacional de un área de sistemas.

La combinación de una estructura funcional y una estructura de equipos puede cumplir con más objetivos, aprovechando las ventajas de cada una de estas estructuras. La Figura 3.6 muestra como puede quedar definida.

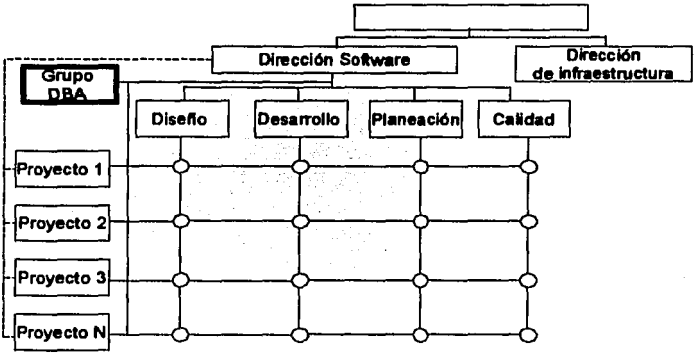


Figura 3.6 Combinación de estructura funcional y de equipos

Los integrantes de los equipos de cada proyecto son personas que pertenecen a cada una o alguna de las áreas, rompiendo de esta manera con las barreras funcionales y con el incumplimiento de objetivos globales, es decir que tenemos las siguientes ventajas:

- Mejor cooperación interfuncional.
- Mejor servicio al cliente a causa de las áreas funcionales
- Mejor definición de responsabilidades por los lideres de proyecto y la comunicación directa en el aspecto funcional y de proyectos.

Cabe aclarar que esta estructura tiene la desventaja de que los integrantes de los equipos tiene que reportar a más de un jefe.

Se propone también el grupo DBA, el cual claramente se recomienda consta de al menos dos persona, este grupo debe ser coordinado por el DBA con mayor experiencia y conocimientos, no se puede evitar que cada proyecto requiera de un DBA, Sin embargo y para aminorar la carga de trabajo del grupo DBA, se podrán descentralizar algunas funciones a las áreas funcionales. Aunque el grupo DBA forma parte del staff de la Dirección de Software, se recomienda trabaje físicamente en el área de infraestructura tecnológica, para aprovechar los conocimientos de otras áreas como: monitoreo del tráfico en la red, protocolos de comunicación, administración de sistemas operativos, etcétera, esto debido a que como mencionamos en el capítulo anterior, la gestión del rendimiento de la base de datos, ya sea reactiva o proactiva, requiere del entendimiento de la infraestructura implicada en los sistemas globales, y no sólo de la base de datos, es decir desde la computadora personal que utiliza una interfaz para comunicarse solicitar los datos, hasta el DBMS que almacena y selecciona el método de búsqueda de los datos, es por ello que el DBA debe de posees conocimientos básicos sobre protocolos, red, sistema operativo, etcétera.

Anexo a la estructura matricial antes mencionada sugerimos dos aspectos que van en contra de la estructura burocrática, pero que a nuestro parecer mejoran la estructura organizacional de una institución, empresa o área dedicada al software.

- Hacer que las cadenas de mando sea más cortas. Esto claramente va en contra de la verticalidad de la estructura burocrática.
- Mayor delegación de responsabilidades y autoridad.

Los beneficios que se pretenden lograr son:

- Disminuir la distancia de la comunicación, el conocimiento y la accesibilidad entre los niveles superior e inferior, para mejorar el tiempo y forma de la toma de decisiones (evitar el "teléfono descompuesto").
- Procurar que la delegación de tareas vaya de la mano con la delegación de autoridad y responsabilidad, tal es el caso de los líderes de proyecto, evitando de esta manera la "reunionitis" de los niveles más altos de las áreas funcionales e incluso del

director del área de software y la falta de atención al cliente, pues en ocasiones éste no es atendido por la sobrecarga de trabajo o compromisos del director o personal ejecutivo.

Capítulo 4

UN MAL EJEMPLO

La Secretaria de Medio Ambiente y Recursos Naturales "Semarnat", a través del área de PROCYMAF (Proyecto de Conservación y Manejo Sustentable de los Recursos Forestales) realizó el desarrollo del Sistema Nacional de Información Forestal "SNIF", lo que a continuación se menciona, es de acuerdo a mi punto de vista y como implicado en el grupo de trabajo del desarrollo del sistema. Cabe aclarar que sólo se presenta un resumen del desarrollo del sistema.

Se ponen en **"negritas"** en éste capítulo algunas frases o enunciados con la finalidad de señalar situaciones de problema o que llegarían a serlo, y a continuación se describe un poco más la frase o enunciado en negritas

Antecedentes

La Secretaría de Medio Ambiente y Recursos Naturales, es la dependencia de gobierno que tiene como propósito fundamental, constituir una política de Estado de protección ambiental que revierta las tendencias del deterioro ecológico y siente las bases para un desarrollo sustentable en el país.

El Proyecto de Conservación y Manejo Sustentable de los Recursos Forestales en México "PROCYMAF" es un proyecto financiado parcialmente por el Banco Mundial que tiene como objetivo instrumentar la estrategia de manejo forestal sustentable descrita en el Plan Nacional Forestal y en el Programa Estratégico Forestal para México a través del impulso de esquemas para: (i) mejorar el aprovechamiento y conservación de recursos naturales por parte de comunidades y ejidos forestales; y, (ii) generar y aumentar las opciones de ingresos de los propietarios con base en sus recursos forestales.

Objetivos, equipo y roles

Los objetivos fueron: proporcionar un sistema para la captura, consulta y actualización de la información de los recursos forestales. Este sistema abarcaba la captura desde cualquiera de las entidades de República Mexicana (la dependencia cuenta con delegaciones estatales), el almacenamiento de esta información en una base de datos, y la consulta de la misma, la consulta podría extraer la información de un sólo repositorio o base de datos, además se podría consultar información concerniente al área forestal, como normas, leyes, cursos, disposiciones, etcétera. De esta manera, el sistema quedó definido por dos módulos o subsistemas:

- Módulo de captura y actualización de los datos, el que sería accesado por las áreas de recursos forestales, en cada delegación estatal de la secretaría y las oficinas centrales de recursos forestales.

- Módulo de consulta al público en general, que permitiría no sólo la consulta de los datos capturados por cada delegación, sino que además se puede consultar documentos referentes a legislación, publicaciones, programas forestales, etcétera.

El área de PROCYMAF recibió presupuesto para poder **contratar a una empresa** para que se hiciera cargo del desarrollo, esto solo incluía el desarrollo, pues debería construirse con herramientas de software y hardware que existieran en la secretaría.
En la Semarnat no existe la coordinación entre los proveedores de servicios de software y el área de sistemas de la dependencia, para asegurar la calidad de los productos.

Los equipos de desarrollo y roles quedaron definidos de la siguiente manera:

- La empresa asignó dos persona al desarrollo del sistema, una se encargaría del desarrollo del módulo de captura y actualización, y otra del módulo de consulta al público en general, por lo que se desempeñaron como líderes y desarrolladores en cada subsistema respectivamente, desempeñando las siguientes funciones:

    - Análisis de requerimientos.

- Diseño y construcción del sistema.
- Selección de herramientas de desarrollo y base de datos.
- Pruebas e implementación del sistema.
- Documentación del sistema.

- Dos personas del área de PROCYMAF como usuarios para que los desarrolladores recabaran los requerimientos del sistema, con las siguientes funciones:

  - Disponibilidad para sesiones de preguntas
  - Probar los módulos ya desarrollados

- Una persona(yo) como enlace para informar de la infraestructura de la Semarnat y ver si los requerimientos no-funcionales cumplieran con las necesidades del desarrollo, además de ser el encargado de la administración de la base de datos una vez implementado el sistema.

  - Reportar las herramientas de desarrollo y bases de datos disponibles en la secretaría.
  - Reportar el tipo de licenciamiento de las herramientas de desarrollo y bases de datos disponibles en la secretaría.
  - **Dar cuentas de acceso y contraseñas, y espacio de trabajo en los servidores que se solicitara**
    Definir y descentralizar tareas sin la correcta planeación y estrategia puede llevar a una base de datos mal organizada.

  - Administrar la base de datos una vez implementado el sistema.

Diseño, construcción y pruebas

Así como el análisis, el diseño, la construcción y las pruebas estuvieron definidas por las dos personas encargadas del desarrollo, para cada unos de los módulos respectivamente, a continuación se da un resumen del resultado de cada una de las etapas del trabajo realizado

por parte de los desarrolladores. Este resumen fue realizado con base en la documentación entregada al final del proyecto, también contiene algunos comentarios enfatizando lo que a mi punto de vista fueron errores de decisión o de desarrollo.

Módulo de captura y actualización:

Arquitectura del Sistema

Se definió una arquitectura cliente/servidor para el desarrollo del Sistema Nacional de Información Forestal, la Figura 3.7 muestra el panorama general de la arquitectura seleccionada.

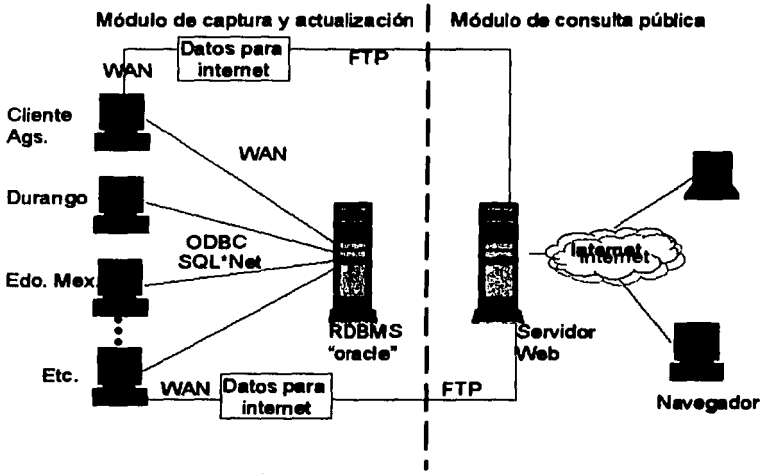


Figura 3.7 Arquitectura del SNIF

- **Como sistema manejador se utilizó Oracle**

  De la selección del sistema manejador tenemos que:
  - Las entidades federativas no requerían información entre ellas.
  - La oficina central no requería información instantánea.

- Los datos que se capturaban claramente no estaban inmediatamente disponibles en Internet.

Por lo que podemos concluir que la selección del software manejador de la base de datos y de la arquitectura fue incorrecta pues el requerimiento de los datos así lo dice. Mucha gente puede decir que Oracle es el mejor DBMS que existe, sin embargo, si una persona quiere llevar el control escolar de su jardín de niños que tiene cien alumnos, ¿Será el mejor DBMS?, o mejor dicho ¿será el más adecuado?.

La empresa consultora no disponía de sistema manejador de bases de datos Oracle, decidió realizar todo el desarrollo con FoxPro como el manejador de bases de datos, posteriormente utilizaría una utilería de migración de FoxPro a Oracle. La migración de una plataforma de base de datos a otra una vez terminado un software no es inmediata y a veces requiere de un arduo trabajo.

- La interfaz del cliente se desarrolló en Visual FoxPro versión 6

El sistema imprime varios documentos usando a Word (Word 97 ó posterior) en el background, transparente para el usuario:

- Certificados de registro en el Registro Forestal Nacional
- Formas de inscripción en el Registro Forestal Nacional
- Oficio que notifica el código de inscripción

Para que el sistema pueda operar en cada estación de trabajo se necesita:

- **Conexión a la red de la secretaría**

Como ya mencionamos, creemos que la elección de la arquitectura fue incorrecta, pues un requerimiento del sistema era que, la persona encargada de la captura de la información forestal de la delegación, tuviese la conexión disponible con el servidor central de la base de datos, no siendo siempre esto posible por cuestiones de infraestructura.

- Instalación del Sistema Nacional de Información Forestal, lo que instala el ejecutable y demás archivos necesarios y hace los cambios requeridos en el Registro de Windows
- Instalación del SQL*Net y de los controladores del ODBC de Oracle
- Word 97 ó posterior

El subsistema de captura constó de los siguientes submódulos:

- Registro Forestal Nacional:
  - Prestadores de servicios personas físicas
  - Prestadores de servicios personas morales
  - Centros de almacenamiento y transformación de materias primas forestales
  - Actividades de manejo de germoplasma forestal
  - Forestación con fines comerciales
  - Cambios de uso de suelo
  - Aviso de reincorporación
  - Informe técnico de saneamiento forestal
  - Aprovechamiento en terrenos agrícolas ó pecuarios
  - Aprovechamiento no maderable
  - Programa de manejo simplificado
  - Programa integrado de manejo
  - Producción y precios
- **Incendios**

  La información de recursos forestales está administrada y supervisada por diferentes áreas de la dependencia, es importante contar con el apoyo y la colaboración de todas y cada una de las áreas implicadas en un proyecto, y no desarrollar "elefantes blancos".

- Plantaciones

Crear la variable m.usu que identifica al usuario. Este dato se usa para **'firmar' cada transacción**, junto con la fecha y la hora

Se implementó un nivel de auditoria mientras el desarrollador construyó con FoxPro como DBMS, pues se desconocían las características para implementar la auditoria en Oracle, lo que llevo al desarrollador a crear campos innecesarios en las tablas, cuando en este caso se pudo realizar de manera más eficaz utilizando las características de Oracle.

El esquema de seguridad se implementa de la siguiente manera:

- El programa de ENTRADA crea variables públicas que representan el nivel de acceso del usuario a los diferentes módulos del sistema
- La pantalla de registro da valores a estas variables, leyendo los permisos que tiene el usuario (después de que haya proporcionado correctamente su contraseña).

El menú PRINCIPAL tiene la misma estructura de los módulos que sigue el sistema:

- Registro Forestal Nacional, Producción y precios, Incendios y Plantaciones. Cada uno de éstos tiene como opciones, el registro de datos, la consulta y la impresión.
- Además, tiene una opción de Administración que da acceso al cambio de catálogos (que requiere un permiso especial), su consulta e impresión y a dos funciones adicionales (ambas requieren permiso de cambiar catálogos):
- **Datos para Internet, que genera archivos de texto para que sean accesados por el módulo de consulta al público.**

El módulo de consulta no realiza consultas al repositorio donde se capturan los datos, lo que indica que disponibilidad de los datos no es inmediata a la captura.

- Reasignación de personas físicas y morales, que permite cambiar la Delegación que tiene acceso a los datos. Esto se usa en el caso de que por ejemplo una Persona Moral Proveedora de Servicios cambie de ubicación y requiera ser atendida por una Delegación diferente a la que lo dio de alta.

Todos los accesos a datos se hacen a través de vistas remotas y locales, con la excepción de los once catálogos cerrados: ACTIVIDAD, CONJ_PREDIAL, FINALIDAD, METODO_MANEJO, PROMOVENTE, SISTEMA_SIL, TABLA_OFICIO, TENENCIA, TIPO, TIPO_APROV, TIPO_TITULAR.

Estas tablas están incluidas en el ejecutable, por lo que si se necesitara hacerles una modificación, sería necesario **volver a distribuir el programa.**

Si son tablas que comparten todas las entidades federativas, ¿por qué no crearlas en la base de datos? de esta manera un cambio en alguna de estas tablas no necesariamente implicaría volver a distribuir el programa.

Todas las vistas remotas utilizan la **misma conexión a Oracle**, que se llama SNIF.

Aunque esto es posible en Oracle se debe de tener cuidado en el número de accesos y en la posibilidad de implementar la auditoria de usuarios, pues si un programa accesa a la base de datos con la misma cuenta no es posible distinguir entre un usuario final y otro.

Todos los accesos a datos se hacen a través de vistas remotas. Las **vistas remotas están filtradas** por una variable de fecha que por omisión corresponde a los últimos quince días, de manera de limitar el flujo de datos

Es importante saber como funcionan los diferentes programas y protocolos de conexión entre programas de desarrollo y las bases de datos, en este caso, inicialmente el programador realizaba consultas a la base de datos donde el número de registros resultantes era numeroso, aproximadamente de 5,000 registros, el módulo de captura y actualización desarrollado hacia la petición de datos a la base de datos y este no desplegaba la pantalla al usuario hasta que todos los registros eran procesados y regresados al cliente, aunque claramente no todos los registros eran desplegados en la pantalla al mismo tiempo, la manera en que los programas implementan el flujo de datos repercute en el rendimiento de los programas que utiliza el usuario final.

Estructura de las Tablas

El sistema cuenta con 16 submódulos: Prestadores de servicios personas físicas, Prestadores de servicios personas morales, Centros de almacenamiento y transformación de materias primas forestales, Actividades de manejo de germoplasma forestal, Forestación con fines comerciales, Cambios de uso de suelo. Aviso de reincorporación, informe técnico de saneamiento forestal, aprovechamiento en terrenos agrícolas o pecuarios, aprovechamiento no maderable, programa de manejo simplificado, programa integrado de manejo, producción y precios, Incendios, plantaciones y los catálogos, de todos los módulos resultaron 64 tablas con aproximadamente 40 campos cada una.

Para resaltar los elementos de interés, es decir, elementos que a mi juicio son errores de estructura de tabla, tipos de datos y nombrado de campos o tablas, entre otros, sólo nos referiremos a la tabla germoplasma que corresponde al submódulo de: Actividades de manejo de germoplasma forestal:

GERMOPPLASMA

Clave	Numérico 6	Campo llave consecutivo
Nombre	Caracter 100	Nombre de la empresa/entidad o persona
RFC	Caracter 13	Registro Federal de Causantes con homoclave de la empresa
Propietario	Caracter 40	Nombre del propietario
RFC_Prop	Caracter 13	Registro Federal de Causantes con homoclave del propietario
Representante	Caracter 40	Nombre del representante legal
RFC_Rep	Caracter 13	Registro Federal de Causantes con homoclave del representante
Actividad	Caracter 1	Tipo de actividad. 1= Recolección, 2=Producción, 3=Almacenamiento
Finalidad	Caracter 1	Tipo de finalidad: 1=Distribución, 2=Comercial, 3=Investigación
Calle	Caracter 35	Dirección de la Instalación
Colonia	Caracter 35	Colonia de la Instalación
Poblad*	Caracter 35	Población de la Instalación
CP	Caracter 5	Código Postal de la Instalación
Estado**	Numérico 2	Clave del estado en el que se ubica la Instalación (por catálogo)
Municipio	Numérico 4	Clave del municipio en el que se ubica la Instalación (por catálogo)

* Existen campos con nombres incompletos

** En otra tabla la columna se llama est

Calle_fis	Caracter 35	Dirección fiscal de la Instalación
Estado_fis	Numérico 2	Estado (por catálogo) fiscal de la Instalación
Municipio_fis	Numérico 4	Municipio ó delegación (por catálogo) fiscal de la Instalación
CP_fis	Caracter 5	Código Postal fiscal de la Instalación
Telefono	Caracter 15	Teléfono en el domicilio fiscal de la Instalación
Fax	Caracter 15	Número de fax en el domicilio fiscal de la Instalación
Fecha_Sol	Fecha	Fecha de solicitud
Fecha_Reg	Fecha	**Fecha de registro***
Seccion	Caracter 2	Datos de registro
Libro	Caracter 2	Datos de registro
Volumen	Numérico 3	Datos de registro
Fojas	Numérico 3	Datos de registro
Numero	Numérico 4	Datos de registro
Folio	Numérico 4	Número del sistema de gestión
Determinante	Numérico 4	Número de oficialía de partes
Observaciones	**Memo****	Observaciones
Inactivo	Carácter 1	Registro activo (en blanco) ó inactivo, C=Cancelación, R=Revocación y S=Suspensión
Delegacion	Numérico 2	Delegación que dió de alta el registro
Fe_ini	Fecha	Fecha de inicio de la inactividad

* No se hace uso de valores por omisión

** Este no es un tipo de dato reconocido por Oracle

Fe_fin	Fecha	Fecha de terminación de la inactividad
Causa	Carácter 200	Causas de la inactividad
Fe_tra	Fecha	Fecha en que se registró la transacción
Ho_tra	**Caracter 8***	**Hora en que se realizó la transacción****
Usu	Numérico 6	Clave del usuario que registró la transacción
Tipo	Caracter 1	Tipo de transacción efectuada. A=Alta, C=Cambio

Catálogos: (MUNI, EST, LIC, ESPE, UNIV, ECOS, USURIOS, etcétera)

Los nombres de algunas tablas no son descriptivos, ejemplo de ello son: EST, MUNI, ESPE, etcétera.

En este último capítulo, se presentó un caso verídico, no con la finalidad de evidenciar una institución o un área, sino los problemas que se presentan en el desarrollo de software y que están asociados a la administración de bases de datos, dichos problemas pueden no ser tan sencillos de resolver para este caso en particular y para cualquier otro, pues esto puede depende de muchos aspectos como las características de intrínsecas de cada institución o área

Son varios los factores que pueden afectar negativamente el desarrollo de un proyecto de software y que estén asociados con la administración de bases de datos, aspectos técnicos, organizacionales, administrativos, etcétera, y estamos seguros de no cubrir todos estos factores, y como administrador de bases de datos, fue mi preocupación el proporcionar algunos elementos que ayuden a la administración de bases de datos y su entorno.

* Este tipo no es correcto para representar la hora

** Todas las tablas implementan la auditoria de transacciones en las tablas junto con la programación, y no se explotan las características de Oracle, esta manera de implementar la auditoria de los usuarios degrada el desempeño de la base de datos.

CONCLUSIONES

Los problemas en torno a la administración de bases de datos son algunos de los factores que puede afectar el desarrollo de software, mi formación como Actuario me permitió involucrarme en el área de la informática como administrador de bases de datos, durante mi experiencia como DBA, pude ser testigo y víctima de algunos de estos problemas, y es por ello, que en la presente tesis mencionamos y señalamos la importancia de las funciones y tareas en la administración de bases de datos, y proponemos sugerencias y soluciones para algunos problemas en torno a la administración de bases de datos.

Para ello, dividimos el trabajo de análisis en tres temas:

I. Estándares
II. Políticas y procedimientos
III. Estructura organizacional

I. Estándares

Problemas: Los estándares que consideramos importantes en torno a la administración de bases de datos son: estándar para el modelado de la base de datos y estándar para el nombrado de objetos en la base de datos, algunos de los problemas que podemos mencionar son:

Estándar para el modelado de la base de datos

- Definir quién realiza el modelado de los datos: ¿El DBA, el analista u otra persona?:

  Si el DBA no realiza el modelo de los datos, se corre el riesgo de que la persona que lo hizo, no posea los conocimientos necesarios sobre la teoría del modelo entidad-relación o la experiencia en el modelado.

  Si el DBA es quien realiza el modelo de los datos requiere del tiempo necesario para dicha tarea, así como de los insumos para ello, lo que podría implicar la desatención de sus actividades principales.

- La creación errónea del modelo de datos provoca problemas de integridad semántica de los datos.

Si no existe la coordinación entre los otros grupos del desarrollo del software y el uso de otros estándares para la comunicación del proceso de los datos, cuando se lleva a la implementación física el modelo de datos aparecerán problemas de integridad semántica.

Y problemas en torno a la falta de un estándar para el nombrado de objetos en la base de datos son:

- Incompatibilidad en el nombrado de objetos en la base de datos.
- Dificultad en la descentralización de funciones.
- Mala documentación de las bases de datos.
- Retraso en creación de sentencias SQL.

Causas:

- No existe en ocasiones la persona o grupo dedicado al modelado de datos.
- La falta de experiencia en el modelado de proceso y datos, así como del modelo entidad-relación.
- No se hace uso de otros estándares que permitan la comunicación entre los procesos del desarrollo de software, para poder llegar a un buen modelo entidad-relación.
- No existe o no se hace uso de un estándar para el nombrado de objetos en la base de datos

Sugerencia:

- Especificación de un grupo o persona como administrador de datos y otro como administrador de la base de datos, definiendo respectivamente sus actividades y funciones, así como la relación entre ambos.
- El apéndice A es una sugerencia de estándar para el nombrado de objetos en la base de datos.

II. Políticas y procedimientos

La falta de políticas y procedimientos provoca básicamente problemas de coordinación y definición de responsabilidades en torno a la administración de bases de datos.
En la falta de políticas, la descentralización de funciones y tareas del DBA, es el problema más común:

- Usuarios con permisos para habilitar y deshabilitar los servicios del DBMS. Sin las debidas precauciones pude provocar pérdida en la integridad (de estructura o semántica) de los datos.
- Usuarios con los permisos para realizar respaldos de las bases de datos. Dichos usuarios a veces no verificarán que el respaldo se haya hecho correctamente.
- Usuarios con los permisos para la creación de objetos en la base de datos. Es tal vez el más común, algunas situaciones en la puede desembocar son:
    - Uso indebido de los tipos de datos en la creación de los mismos.
    - Creación de objetos en las bases de datos sin apego a un estándar para el nombrado de los mismos.
    - Afectar el rendimiento de las aplicaciones mediante la construcción de programas para implementar auditorias (más inserciones, borrados, o actualizaciones).
    - Afectar el rendimiento de la base de datos al no almacenar de la mejor manera los objetos y los registros en la base de datos.

La falta de procedimientos pone en riesgo la disponibilidad e integridad de los datos, y la falta de documentación de los mismos hace al DBA indispensable, lo cual no es bueno. Entre la falta de procedimientos y documentación de los mismos podemos encontrar:

- Actualizaciones del DBMS.
- Migración de bases de datos entre versiones del DBMS y otros DBMS.
- Respaldos de las bases de datos.
- Recuperación de las bases de datos.

Algunos problemas que podemos mencionar por la falta de procedimientos son:

- Deslindar responsabilidades en la perdida de bases de datos.
- Tiempo requerido que una base de datos afectada por una falla o acontecimiento tardará en restituirse y estar disponible.
- Degradación del desempeño de las bases de datos.

Sugerencias: El Anexo B es una guía para la construcción de un plan de contingencia de una base de datos, la creación de un documento así obliga a la definición de procedimientos para los temas de respaldo y recuperación, así como la documentación de los mismos.

III. Estructura Organizacional

En lo que respecta al grupo o persona encargado de la administración de bases de datos, es muy importante que este tenga una adecuada ubicación de la estructura organizacional, de no estar en la ubicación correcta se pueden presentar los siguientes problemas:

- Mala división de trabajo.
- Mala comunicación con otros grupos.
- Entre otros problemas

Analizamos tres estructuras organizacionales: funcional, por proyecto o equipo y burocrática, algunas desventajas en cada una de ellas son:

- Existen o se crean barreras funcionales, que consisten en la falta de comunicación y coordinación entre las diferentes áreas incluyendo a la administración de las bases de datos (funcional).
- Duplica recursos, para el caso de DBA's, no es lo mismo pagar el salario de un DBA que de varios(por proyecto).
- Jerarquía estricta y administración demasiado vertical (burocrática).

Sugerencias: se propone una combinación de la estructura funcional y la estructura por equipos aprovechando las ventajas de cada una de ellas, para así obtener las siguientes características:

- Mejor cooperación interfuncional.

- Mejor servicio al cliente a causa de las áreas funcionales
- Mejor definición de responsabilidades por los lideres de proyecto y la comunicación directa en el aspecto funcional y de proyectos.

Se propone también que el grupo de administración de bases de datos forme parte de un staff, el cual se recomienda conste de al menos dos personas, y éste grupo debe ser coordinado por el DBA con mayor experiencia y conocimientos.

Podemos concluir que no basta con tener un buen sistema manejador de bases de datos, o contar con el mejor administrador de bases de datos, sino como mencionamos al principio, son muchos los factores que se requieren para tener una buena administración de las bases de datos, algunos de los cuales estamos seguros no se mencionan en este trabajo, como podría ser:

- La definición de las políticas existentes en torno a la administración de bases de datos.
- Un estándar para la documentación de los sistemas manejadores de bases de datos.
- Otros.

Pero esperamos que las sugerencias y soluciones aquí planteadas sean de utilidad y resuelvan algunos de los problemas en torno a la administración de bases de datos.

ANEXO A

Estándar para el nombrado de objetos en una base de datos

Esta sugerencia para el nombrado de objetos de una base de datos intenta y trata de ser sencilla, fácil e independiente del DBMS, los objetos que consideramos que requieren de un estándar para su creación son los siguientes:

- Tablas
- Columnas
- Tipos de datos definidos por el usuario
- Llaves primarias
- Llaves foráneas
- Restricciones de verificación y valores por omisión
- Vistas
- Procedimientos almacenados
- Funciones definidas por el usuario
- Triggers
- Índices

Tablas.

Una tabla es la representación física de un conjunto de entidades, recomendamos entonces nombrar las tablas de acuerdo a la entidad que representan, y ya que la tabla es un conjunto de entidades nombrarlas en plural, algunos ejemplos son los siguientes:

Clientes

Ordenes

Nombrar la tabla de empleados "empleados" y no "emp", es también otra sugerencia, de hecho usar un tamaño mínimo sugerido para el nombrado de tablas es bueno, un tamaño

mínimo recomendado puede ser de 8 caracteres, al referirse al tamaño mínimo puede ser no tan estricto.

Para las tablas que pueden ser la representación de una relación, utilizar nombres completos pero en singular y separar los nombres de las entidades en dicha relación por un guión bajo "_", ejemplo: cliente_cuenta.

Columnas

Las columnas representan los atributos de una entidad, es decir, que las columnas describen las propiedades en la entidad, entonces especificar los nombres de las columnas lo más descriptivos posibles, algunos ejemplos son:

apellido_paterno

domicilio

Se puede definir un tamaño "mínimo sugerido" para el nombrado de columnas, sin embargo en este como en todos los casos, será "mínimo sugerido " ya que pueden surgir casos en los que sea más difícil usar el nombre largo algunas abreviaciones que comúnmente usamos, como son las siguientes:

CP

RFC

Dos casos para el nombrado de columnas son de especial interés, las columnas que representan una llave primaria y las columnas que representan una llave foránea. Cuando se va nombrar una columna que es la llave primaria de una tabla se define el nombre de la columna de la siguiente manera: id+"_"+"nombre de la tabla en singular".

Id_empleado

El prefijo "id" puede ser cambiado por algún otro como "cve", esto es meramente estilo, pero se recomienda usar sólo uno. Esto claramente sólo se aplica cuando la llave primaria esta definida físicamente por una sola columna.

Para el caso en que la columna que se va a nombrar representa una llave foránea, y esta es como en el caso anterior, es decir compuesta por sólo una columna, se define de la siguiente manera: id+"_"+"nombre de la tabla en singular en la cual es llave primaria", en otras palabras tendrá el mismo nombre que en la tabla en la que es llave primaria, el siguiente ejemplo muestra lo antes mencionado:

```
Create table ordenes
    (id_orden integer primary key,
     id_empleado integer
    constraint fk_ordenes_empleados id_empleado references
empleados(id_empleado));
```

Tipos de datos definidos por el usuario

Los tipos de datos definidos por el usuario, son una extensión de los tipos base existentes en las características del DBMS, y son usados para mantener la consistencia entre los tipos de datos de diferentes tablas y para definir y restringir las operaciones entre los datos. Definir entonces los nombres para tipos de datos definidos por el usuario lo más descriptivo posible, adicionalmente se puede agregar una abreviación para el tipo de dato, es decir: nombre+"_"+"abreviación de tipo"

siglas_str

Empleado_record

Restricciones

Llaves primarias

Para el caso en que las llaves primarias que están compuestas por más de un campo, definir el nombre de la restricción de llave primaria utilizando el prefijo "pk" y el nombre de la tabla en plural, como en el siguiente ejemplo:

```
Create table pedidos
    (fecha date,
     hora date,
     id_cliente integer
    constraint pk_pedidos primary Key (fecha, hora, id_cliente));
```

Llaves foráneas

Para las llaves foráneas que hacen referencia a llaves primarias compuestas por más un campo, definir el nombre de la restricción de llave foránea utilizando el prefijo "fk" y el nombre de la tabla en plural: "fk"+"_"+"tabla que contiene la llave primaria"+"_"+"tabla con las columnas que componen la llave foránea"

```
Create table ordenes
     (id_orden integer primary key,
      id_empleado integer
     constraint fk_ordenes_empleados id_empleado references
empleados(id_empleado));
```

Verificación y valores por omisión

Para las restricciones de verificación y de valores por omisión, definir el nombre de la restricción utilizando los prefijos "chk" y "def" respectivamente, y posterior al prefijo un nombre descriptivo para la restricción separados por un guión bajo como lo muestra el siguiente ejemplo:

"chk_rfc" para una columna que checa el campo del RFC tenga mínimo 10 caracteres

Vistas

Para el nombrado de vistas se sugiere utilizar la misma definición hecha para las tablas, utilizando el guión bajo("_") para separar las diferentes palabras que compongan el nombre del objeto.

Clientes_cuentas

Procedimientos almacenados

Los procedimientos almacenados son bloques de código escritos en una combinación de SQL y algún lenguaje de programación, estos son escritos para realizar alguna acción específica, es por ello que para el nombrado de los procedimientos almacenados se sugiere utilizar el verbo que describe la acción más un nombre que ayude a describir el objetivo del procedimiento, es decir: "verbo"+"_"+"nombre, tabla, columna, etcétera".

execute obtener_detalles_venta;

Funciones definidas por el usuario

La todos los DBMS's o la mayoría de ellos tienen implementados un conjunto de funciones como características nativas del DBMS, para las funciones definidas por el usuario definir los nombres de manera similar a los procedimientos almacenados.

Triggers

Los triggers son procedimientos que se ejecutan cuando ocurre algún evento como insertar, borrar o actualizar un registro, y dado que son procedimientos se sugiere definir el nombre de ellos similar a los procedimientos con un prefijo que indique el tipo de evento que se requiere para ejecutarse.

on_ins_ revisa_saldo revisa el saldo cada ves que se inserta un registro

on_del_borra_historial_cuenta borra el historial de la cuenta que se esta eliminando

on_upd_revisa_edad revisa la edad cada ves que se actualiza el registro.

Índices

Los índices son creados para mejorar el tiempo de búsqueda de los datos, y por consiguiente dependen de una tabla o vista, la definición del nombrado de índices será usando el prefijo "idx" más los nombres de las columnas que componen el índice, es decir: "idx"+"_"+"nombres de las columnas".

Idx_ estado_municipio

En ocasiones se recomienda, para los casos en que el índice sea una llave primaria o una lleve foránea, agregar una abreviación al final "pk" y "fk" respectivamente, quedando entonces de la siguiente manera: "idx"+"_"+"nombres de las columnas" +"_"+"pk,fk."

Idx_curp_pk

ANEXO B

Guía para un plan de contingencia de una base de datos

Existe una amplia gama de posibilidades que puede provocar la falla o pérdida de los datos o del servicio del DBMS, hemos hablado ya de los mecanismos de respaldo y de recuperación de una base de datos, aspectos que surgen debido a éstas posibilidades y que nos ayudan a resolver los problemas de la falla o pérdida de los datos, o el servicio del DMBS. Sin embargo, cómo sabemos que el administrador de la base de datos posee los conocimientos necesarios para enfrentar una situación que implique la recuperación de la base de datos, o si el DBA esta de vacaciones o fuera de la oficina y ocurre un evento que provoque la falla o pérdida de datos o del DBMS, ¿qué hace una institución o empresa en ese caso?

Es claro que el DBA es la persona que tiene grandes responsabilidades, como lo mencionamos en el Capítulo 1, es el que tiene la responsabilidad central de los datos, es común que en una institución o empresa el DBA se haga una persona casi imprescindible, localizable las 24 horas del día, los 365 días del año, y esto debido a que si algo le pasa a la base de datos "sólo el DBA es capaz de resolverlo", esto es verdad en cierta medida, más no del todo.

Como el DBA tiene los conocimientos necesarios para enfrentar una posible falla o pérdida de datos o del DBMS, es responsable de elaborar e implementar un plan de recuperación de la base de datos cuando ocurre un evento que provoque la falla o pérdida de datos o del DBMS, esto garantiza que el DBA posee los conocimientos para enfrentar una situación dada y permite documentar las acciones a tomar, lo que podría en algún momento dado o en algunos casos ayudar a la recuperación de la base de datos sin la presencia del DBA.

A continuación proporcionamos una guía para la elaboración de un plan de contingencia, en el caso de la falla o pérdida de datos o del DBMS. Esta guía debe aplicarse por cada DBMS.

Guía para la elaboración de un plan de contingencia de una base de datos

I. Generales.

a. Introducción.

Describir brevemente el plan de contingencia, hacia quien va dirigido, que área o departamento lo elaboró y la motivación del documento.

b. Alcance.

Establecer el alcance del documento, es decir, las bases de datos y procedimientos del DBMS en cuestión pues sólo hará referencia a estas, y a las aplicaciones que tengan relación con las bases de datos en dicho DBMS. Mencionar que el documento no es de alto detalle, en cuanto a tareas y acciones se refiere, sino que es una guía para la recuperación de la base de datos, es decir, que para poder aplicar dicha guía se requerirá de ciertos conocimientos, como son: sistema operativo, bases de datos, etcétera.

c. Objetivos.

Definir que tipo de problemas, en cuanto a falla o pérdida de datos se pretenden resolver en dicho documento y que tipo de problemas no se abordan.

II. Análisis de impacto y de riesgo.

a. Escenarios de desastre.

Establecer los escenarios de desastre que se pretenden resolver la pérdida de datos o del DBMS, algunos ejemplos son: terremotos, huracanes, pérdida de energía eléctrica, pérdida del servidor donde reside el DBMS, pérdida de un filesystem, pérdida de un disco duro, etcétera.

b. Identificar aplicaciones, bases de datos, procesos y funciones críticas.

Identificar mediante entrevistas con los ejecutivos o directores las bases de datos y aplicaciones de mayor importancia para la institución. Documentar dicha identificación, el criterio y entrevistado responsable de valorizar y dar los niveles de importancia.

c. Establecer métricas para la pérdida de aplicaciones, bases de datos, procesos y funciones críticas por cada una de ellas con relación al tiempo.

Mediante las entrevistas de identificación de aplicaciones, bases de datos, procesos y funciones críticas, tratar de establecer métricas cuantitativas o cualitativas sobre la pérdida de datos o de la base de datos.

d. Identificar puntos vulnerables.

Identificar puntos vulnerables, ejemplo: puertos abiertos del servidor donde reside el DBMS, vulnerabilidad del sistema operativo, configuración del arreglo de discos, etcétera.

III. Desarrollar la estrategia de recuperación.

a. Priorizar las aplicaciones, bases de datos, procesos y funciones críticas

Dar un nivel de prioridad o de importancia cuantitativo a las aplicaciones y bases de datos. En este punto sólo se requiere dar una lista con orden de importancia, ya no es necesario describir él porque del orden, el orden de esta lista servirá para establecer el orden de importancia en la recuperación de las aplicaciones y bases de datos, es decir, se puede tomar la decisión de recuperar la base de datos parcialmente.

b. Por cada situación de desastre y para cada aplicación, bases de datos, procesos y funciones crítica:

i. Definir las tareas, procesos y recursos para solventar el problema

Definir las tareas, procesos y recursos, implica tener preparado la información necesaria para la recuperación de la base de datos, como puede ser: respaldos de la base, archivos de registro de operaciones (log files), scripts para la recuperación de la base de datos, archivos de configuración del DBMS, etcétera., también es indispensable especificar la ubicación de todos estos archivos

ii. Identificar los componentes que integra el elemento afectado

Identificar las diferentes plataformas y proveedores de ellas, que están implicadas en cada aplicación y base de datos.

iii. Realizar lista de responsables con todos los teléfonos posibles

Realizar una lista de los responsables con todos los teléfonos posibles, como son: teléfono de casa, teléfono del hotel donde se hospeda, etcétera, así como los teléfonos de las posibles personas que pudieran ayudar a resolver un problema de los descritos en el alcance por la ausencia del DBA.

En este punto es muy importante que, para cada situación de desastre y para cada aplicación o base de datos crítica, se especifique las tareas y procesos paso a paso y puntualmente.

IV. Pruebas.

a. Desarrollar plan de pruebas

Desarrollar el plan de pruebas contemplando o simulando los escenarios de desastre y la estrategia de recuperación de las bases de datos.

b. Conducir las pruebas

Realizar las pruebas de acuerdo al plan de pruebas.

c. Modificar el plan si es necesario

En caso de alguna falla en la estrategia de recuperación, modificar los puntos necesarios hasta quedar satisfechas de acuerdo al plan de pruebas.

BIBLIOGRAFÍA

C. J. Date. "An Introduction to Database Systems". Addison-Wesley, 7th edition. 2001.

Craig S. Mullis. "Datebase Administration the Complete Guide to Practices and Procedures". Addison-Wesley. 2002.

Gary W. Hansen, James V. Hansen. "Diseño y Administración de Bases de Datos". Prectice Hall. 1997.

Abraham SilbersChatz, Henry F. Korth, S. Sudarshan. "Database System Concepts". Mc.Graw Hill, 4th edition. 2001.

Michael Ault. "Oracle 8i Administration and Management". Jhon Wiley & Sons, Inc.. 2000.

Walker Royce. "Software Project Management a Unified Framework". Addison-Wesley. 1998.

Mathias Kirchmer, Mathaias Kirchmer. "Business Process Oriented Implementation of Standard Sofware". Springer Verlag, 2nd edition. 1999.

Jhon R. Schermerhorn. "Administración". John Wiley & Sons, 7th edition. 2001.

Roger S. Pressman. "Ingeniería de Software, un enfoque práctico". McGraw-Hill. 1998.