Material sobre Diseño de Bases de Datos

Documentación

Módulo 1. Conceptos Básicos de Bases de Datos

Qué es una Base de Datos?

Es una gran colección de información almacenada y relacionada entre sí y que concierne a un tema de interés para una empresa. Las bases de datos nacieron para darle a los usuarios la posibilidad de tener la información centralizada, pero con el transcurrir del tiempo y de la tecnología, las necesidades del usuario han ido evolucionando a tal punto que encontramos hoy día bases de datos con la información dispersa en diferentes servidores; esto es lo que se llaman bases de datos distribuidas.

Entonces cual es la diferencia entre un sistema de archivos tradicional y una base de datos?

Si miramos la definición de Base de Datos nos encontramos con la misma definición de sistema de archivos. Con un conjunto de archivos se puede tener también un conjunto de datos relacionados entre sí, posiblemente centralizados, y que también conciernen a un tema en particular de la empresa. Es decir, una Base de Datos se puede considerar como un sistema de archivos desde el punto de vista de su objetivo final, más no desde el punto de vista de su utilización. ES decir, con un conjunto de archivos bien diseñados acerca de la nómina de una empresa yo puedo obtener los mismos resultados que con una base de datos del mismo tema y que contenga los mismos datos. Lo que sucede es que la forma de obtener estos resultados es mucho más práctico y fácil con una base de datos que con un sistema de archivos.
El elemento que hace que en una base de datos los resultados se obtengan de una manera más facil se llama el DBMS.

Qué es el DBMS?

La sigla DBMS significa DataBase Management System (Sistema Manejador de Bases de Datos) y es lo que comúnmente conocemos como "motor" de bases de datos.
Los términos DBMS y Bases de Datos se utilizan, en el argot cotidiano, sin distingo pero en realidad no significan lo mismo.
Cabe decir que una base de datos es parte de un DBMS. En otras palabras, un DBMS es un software que contiene 2 grandes módulos:

Repositorio de Datos: Es propiamente el sitio donde el usuario va a guardar

los datos propios. ES la base de datos como tal.
Software de Manipulación de Datos: Es un software que viene incorporado al DBMS y que sirve para manipular directamente los datos que existen en la base de datos. Por ser un software nativo del DBMS, el acceso a dichos datos se hace de una manera mucho más rápida y efectiva.

Es este segundo componente el que diferencia a un ambiente de Archivos con uno de Bases de Datos. En un sistema de archivos los datos deben ser manipulados por programas escritos en lenguajes de programación convencionales (de tercera generación), los cuales dependen mucho de la estructura física de almacenamiento de los datos. En cambio, en un sistema de Bases de Datos, los datos son manipulados por el mismo DBMS o por programas escritos en lenguajes de cuarta generación que vienen incorporados al módulo de software del DBMS.

Ejemplos de DBMS

El siguiente es un listado de DBMS existentes comercialmente:

Oracle
Informix
DB2
Sybase
Progress
SQL Server
MySQL
Paradox
Fox Pro
Access
Postgresql

Ventajas de los DBMS

El hecho de que la tecnología de bases de datos haya desplazado a los sistemas manejados por archivos tradicionales no es gratuito. Existen un conjunto de ventajas de las bases de datos sobre los archivos que hacen que desarrollar aplicaciones con esta tecnología sea mucho mas atractivo para la empresa. A continuación, se explican en detalle cuales son las ventajas de la tecnología de bases de datos:

1. Centralización de datos.

   Una de las características que tienen los sistemas de archivos es la situación que se denomina "aislamiento de datos". Esto se refiere al hecho de que, cuando en una empresa se empiezan a manejar archivos de datos, éstos se van convirtiendo en islas de información aislada. Este aislamiento se va dando debido a que cada dependencia de la empresa va almacenando en

sus propios archivos los datos que ella necesita, sin contar que podría ser que en otros archivos de la empresa ya existieran dichos datos. Es decir, cada dependencia se va convirtiendo en una "isla de información" aislada de las demás.
Con las bases de datos esto no sucede. La filosofía de centralización de datos que maneja esta tecnología hace que toda la información empresarial se encuentre en un solo repositorio y por lo tanto es poco factible que la información se encuentre replicada. Tenemos un único contenido de información que suple las necesidades de todas las dependencias de la empresa.
Por lo tanto, la centralización nos evita la REDUNDANCIA de datos, es decir, no vamos a encontrar datos duplicados (o n-plicados) dentro de la base de datos.

2. Consistencia de los datos.

   Cuando tenemos islas de información, tal y como se expuso en el numeral anterior, la probabilidad de que existan inconsistencias en los datos es muy alta, debido a las múltiples copias que se manejan de los mismos. Por ejemplo, el departamento de Nómina puede tener grabado dentro de sus archivos la dirección de residencia de cada uno de sus empleados. Lo mismo puede suceder con el departamento de Contabilidad. Si en algún momento uno de los empleados cambia de dirección de residencia y solo es informado el departamento de Nómina de esta situación, el archivo de datos de Nómina quedará actualizado más no el archivo de Contabilidad. Por lo tanto, el archivo de datos de Contabilidad estará desactualizado. Y lo que es mas grave aún es que un mismo empleado aparecerá con dos direcciones diferentes en los archivos de la empresa. La consecuencia de esto es que cada vez que se necesite actualizar un dato, se deben actualizar tantas veces como copias existan del mismo.
   Por la centralización de datos, en las bases de datos es poco probable que esto suceda debido a que lo que se promulga es la no duplicación (o nplicación) de los datos. La consistencia de datos es mucho más fácil de mantener en un ambiente de bases de datos.

3. Seguridad

   La seguridad, en un ambiente de manejo de datos, tiene mucho que ver con lo que se denomina control de acceso. Es decir, se encarga de definir exactamente lo que cada usuario estará en capacidad de ver de todo el conjunto de datos que se tiene grabados.
   Manejar la seguridad en un ambiente de archivos es mucho más complejo que en una base de datos; esto debido a que los sistemas de archivos, por si mismos, no tienen implementados módulos que permitan hacer esto, mientras que las bases de datos sí. Lo que sucede en una base de datos es que se define una sola vez estos controles de acceso y éstos quedan grabados en el catálogo del sistema (sitio de la base de datos en el cual se

guardan los metadatos, es decir, datos acerca de los datos). Posteriormente si existe una aplicación que trata de violar esos controles de acceso, es decir si un usuario trata de ver información para la cual no está autorizado, el motor de la base de datos AUTOMATICAMENTE saca el mensaje de error indicando la negativa para poder realizar dicha acción.
En un ambiente de archivos esto no es automático. Recordemos que una cosa es la estructura de datos que escojamos para almacenar los datos (en este caso los archivos) y otra el lenguaje escogido para desarrollar las aplicaciones que manipularán dichos datos. Este control de acceso debería ser implementado en las aplicaciones, en todo programa en el cual se requiera hacer el control.
Por lo tanto, mientras que en una base de datos el módulo de seguridad ya existe, en un sistema de archivos debe ser implementado por el analista/programador.

4. Fácil acceso a los datos

Suponga el siguiente caso típico en las empresas: usted, que es analista/programador del departamento de sistemas, recibe un pedido urgente del gerente de la compañía en el cual le solicita generarle un informe con las ventas realizadas por producto en cada uno de los 3 meses anteriores. Esto con el fin de contar con material de trabajo para una reunión con la junta directiva en horas de la tarde del mismo día. Usted, muy diligente que es, empieza a hacer su trabajo y luego de unos minutos tiene generado el informe. Pero faltando 10 minutos para la reunión, llega el gerente y le cambia el panorama diciéndole que el listado solicitado en la mañana ya no es el que necesita sino que necesita otro con las ventas en cada una de las sucursales en los últimos 6 meses. Cual sería su reacción? De desespero o de tranquilidad?
Todo depende del ambiente tecnológico en el cual esté trabajando. Si es en ambiente de sistemas de archivos la reacción sería de desespero, pero si es en bases de datos habría una reacción de tranquilidad. La razón de esta diferencia es que el acceso a los datos en un ambiente de bases de datos es mucho más fácil que en un sistema de archivos, debido a la existencia del software de manipulación de datos que viene incorporado al DBMS y del cual ya hablamos en apartados anteriores. A través de este software y del lenguaje SQL que viene incorporado con el DBMS, hacer cualquier consulta y generar un informe sobre la base de datos es cuestión de unas cuantas instrucciones. Mientras que en un sistema de archivos, cualquier informe requerido representa desarrollar un programa completo para cumplir con las especificaciones requeridas. En el caso expuesto, en un sistema de archivos a usted como analista le hubiera tocado hacer OTRO PROGRAMA en el lenguaje utilizado para poder suplir la necesidad del gerente. Y hacerlo en 10 minutos! Mientras que en un ambiente de bases de datos, sería cuestión de reformular la consulta de SQL ya ejecutada para consolidar los datos por sucursal.
El software adicional que viene con el DBMS hace que el acceso a los datos

sea mucho mas inmediato y fácil.

5. Atomicidad de transacciones

   Como veremos más adelante, una transacción en bases de datos corresponde a una unidad lógica de proceso que afecta la base de datos. Por ejemplo, una consignación bancaria, una transferencia de fondos, una aplicación de un inventario físico, entre otras cosas.
   Si en un sistema de archivos hay una falla en la mitad del procesamiento de una transferencia de fondos, posiblemente los datos acerca de las cuentas involucradas en la transferencia podrían quedar inconsistentes. Se puede dar la posibilidad de que la falla se produzca inmediatamente después de haber debitado el valor de la cuenta origen y justo antes de acreditarla a la cuenta destino. Esto haría que la cuenta origen tuviera su saldo restado pero no sumado a la cuenta destino. Obviamente, en este momento los datos estarían inconsistentes y le tocaría al analista corregir esta situación, cambiando los datos de los archivos en forma manual o, si el problema es de muchos más datos, generando un programa de emergencia que permita recorrer los archivos en búsqueda de datos inconsistentes para ser corregidos.
   Por el contrario, en un ambiente de bases de datos esto no sucederá. Cuando una transacción sufre una falla en el transcurso de su ejecución, el motor tiene la capacidad AUTOMATICA de deshacer lo que dicha transacción había hecho hasta el momento dejando la base de datos en el estado en el que estaba al iniciar la ejecución de la transacción.
   Se le llama atomicidad de las transacciones por el hecho de que el motor trata al procesamiento de transacciones como si fuera una sola unidad de procesamiento, a pesar de que éstas constan de múltiples instrucciones.

6. Manejo del Control de Concurrencia

   Se entiende por concurrencia a la situación en la cual varios usuarios al mismo tiempo están accediendo, para consulta o actualización, a los mismos datos. En este contexto, si en un sistema de archivos ocurre esto, y los usuarios involucrados están tratando de hacer una actualización sobre el mismo dato, se podría generar un problema de consistencia de datos ya que las "transacciones" involucradas no se están viendo entre si.
   En un motor de bases de datos, está situación está siendo controlada por módulos del DBMS que cumplen esta función. Es decir, AUTOMATICAMENTE el motor hace una planeación de como va a ejecutar transacciones concurrentes sin que estas se interfieran entre sí, pudiendo dejar datos inconsistentes.
   El tema de Control de Concurrencia será manejado en módulos posteriores de una manera más detallada.

7. Independencia de la Estructura de Almacenamiento

Cuando se tienen n programas que acceden a un archivo en particular y se da la situación de cambiarle la estructura física de ese archivo (adicionarle un nuevo campo, modificar el tamaño de un campo existente, borrar un campo, etc.), se debe modificar cada uno de los n programas para que reconozca la nueva estructura del archivo. Es así como en COBOL, por ejemplo, si esto sucede se debe modificar la estructura del archivo expuesta en el DATA DIVISION. Es decir, las aplicaciones son dependientes de la estructura de almacenamiento.
Con bases de datos esto no sucederá. Si hay que cambiar la estructura de una tabla, lo que se verá afectado será el catálogo del sistema (y esto lo modifica AUTOMATICAMENTE el motor de la base de datos) y no las aplicaciones, a no ser que éstas vayan a utilizar el campo (o campos) modificados de la tabla. En bases de datos, hay plena independencia entre la estructura de almacenamiento y los programas de aplicación.

Cuando no se justifica manejar bases de datos?

Hay ocasiones en las cuales manejar una base de datos en la empresa no sería de gran utilidad. No son muchas las justificaciones pero en las siguientes ocasiones habría que pensar en permanecer en un sistema de archivos, mas que en migrar a un ambiente de bases de datos:

a) Cuando la cantidad de datos a manejar no es importante, es decir, cuando tenemos la certeza de que el numero de datos que vamos a mantener no va a crecer demasiado.

b) Cuando el número de consultas a realizar no es muy alto, o cuando la cantidad de registros a consultar es muy limitada.

c) Cuando se van a mantener muchos archivos maestros y pocos archivos de novedades. Es decir, cuando la información es muy estática y por lo tanto no se modifica muy constantemente.

Niveles de Abstracción de Datos

Cuando hablamos de Bases de Datos, hay diferentes formas de mirar su estructura. Dichas formas dependen del usuario involucrado y de lo que se desea ver de la base de datos. Estas distintas formas de mirar la base de datos se conoce como los Niveles de Abstracción de Datos.

Los siguientes son los 3 niveles de abstracción de datos que existen:

a) Nivel Físico: Es aquel nivel donde lo que se desea ver de la base de datos es su estructura física, es decir, la forma como físicamente está almacenada. El usuario que está localizado en este nivel de abstracción está interesado en

conocer como se almacena físicamente una base de datos en el disco duro, con qué estructuras de datos realiza esta función, cual es el rendimiento de los recursos del sistema (memoria RAM, disco duro, procesador, etc.) ante las consultas que se hacen sobre la base de datos, cual es el rendimiento de la base de datos comparado con el hardware disponible, entre muchas otras cosas.
La persona cuya labor dentro del departamento de sistemas de una compañía está enmarcada dentro de este nivel de abstracción de datos, es el Administrador de la Base de Datos (DBA: Database Administrator). En un apartado más adelante se estará detallando cuales son las funciones de este cargo.

b) Nivel Lógico (o Conceptual): En este nivel lo que se desea ver de la base de datos es principalmente qué datos están grabados en ella y cuales son las relaciones existentes entre estos datos. A este nivel no interesa la parte física de almacenamiento. Acá solo se habla de tablas, claves primarias y foráneas (y por lo tanto, de relaciones). Es a este nivel donde la persona involucrada realiza el diseño conceptual (o lógico) de la base de datos.
La persona que se encuentra en este nivel de abstracción se denomina el Analista / Programador. Esta persona se encargará de hacer el diseño general y detallado de la base de datos en su concepción lógica, es decir, en cuanto a qué datos va a almacenar y como van a estar relacionados estos datos.

c) Nivel de Vistas: Antes de explicar este nivel de abstracción, es conveniente hacer una claridad acerca del término "Vista". En bases de datos, este término tiene doble significado. Hay un primer significado que corresponde a "estructuras" que se graban en la base de datos y que son el resultado de una consulta hecha sobre una tabla (o varias de ellas). Más adelante, en este manual, se hablará sobre este significado del término.
El segundo significado es el que necesitamos utilizar acá y corresponde al subconjunto de datos que un usuario en particular está interesado en ver de la base de datos. Es decir, acá estamos hablando del usuario final y cada uno de los datos de su interés. En otras palabras, hay tantas vistas como usuarios finales tenga la base de datos.
Si pensamos un momento nos damos cuenta de que al usuario final de una base de datos no le interesa saber como están físicamente almacenados sus datos en el disco duro (cuestión que, por lo demás, es demasiado técnica) ni tampoco estará interesado en saber cuales tablas componen la base de datos y qué claves primarias y foráneas poseen esas tablas (esto, como ya lo dijimos, es menester del Analista / Programador). Solo le interesa poder encontrar la información requerida por él en un momento determinado.

Por la explicación anterior, nos podemos dar cuenta de que el nivel que tiene menos abstracción es el físico y el que más abstracción posee es el de vistas. Pero abstracción con respecto a qué? A los detalles del aspecto físico del manejo de los datos de la base de datos (almacenamiento, procesamiento, recuperación).

La gran conclusión a este aspecto es que con las bases de datos le estamos ocultando detalles de manejo físico de los datos a los usuarios que no están interesados en este asunto, lo que no sucede con los sistemas de archivos donde, por ejemplo, el Analista/Programador debe estar enterado de la estructura física de los archivos y sus cambios para poder modificar las aplicaciones que manejan dichos archivos.

Conceptos que no se deben confundir

Existen dos conceptos que, por su similitud linguística, tienden a confundirse en el marco de las bases de datos: ESQUEMA y EJEMPLAR. Vamos a definir cada uno de los dos términos y analizaremos sus diferencias.

Esquema: Estructura de la Base de Datos, es decir, de qué tablas está conformada la base de datos, qué relaciones hay entre estas tablas, qué campos componen cada tabla, de que tipo de dato y tamaño es cada campo. En otras palabras, es el formato o esqueleto de la base de datos donde se van a introducir los datos de la misma. Es el boceto o maqueta de las estructuras de la base de datos que van a contener los datos. En resumidas cuentas, el esquema de la base de datos no tiene propiamente que ver con los datos que hay en una base de datos sino con las estructuras que los contienen.

Ejemplar: Instancia de la Base de Datos. Esta palabra instancia está muy ligada a la programación orientada a objetos y se usa cuando se dice que un objeto es una instancia de una clase. Acá, el significado en forma análoga es el mismo. Cuando decimos que el ejemplar es una instancia de la Base de Datos nos referimos a la situación que tiene una Base de Datos, con respecto a sus datos, en un momento determinado del tiempo. Es decir, el ejemplar corresponde a los datos que tiene grabada una Base de Datos en un instante t del tiempo. Es como si le tomáramos una fotografía a la base de datos. A diferencia del esquema, el ejemplar de una base de datos tiene que ver mucho con los datos que hay contenidos en ella.

Como podemos observar con las dos definiciones anteriores, mientras que el esquema de una base de datos es casi estático, el ejemplar de la misma es muy dinámico. Lo anterior teniendo en consideración que la estructura lógica de una base de datos no cambia todos los días (no todos los días se adicionan nuevas tablas a una base de datos, ni se le adicionan campos a las tablas, etc.), pero los datos contenidos en ella sí están en continuo cambio.

En el contexto de este manual, nos vamos a dar cuenta de que tanto el concepto de Esquema como de Ejemplar son fundamentales dentro de la terminología técnica de las bases de datos.

Lenguajes Utilizados con Bases de Datos

Así como existen muchos lenguajes de programación en los cuales podemos implementar aplicaciones que manejen archivos convencionales de datos, también existen lenguajes que se crearon con el fin de optimizar el acceso a los datos de una base de datos.

Antes de entrar a detallar en el tema de los lenguajes utilizados en bases de datos, debemos tener claro los conceptos de lenguajes procedimentales y lenguajes no procedimentales. Así que procedamos a introducir estos dos conceptos. Es de aclarar que la definición que se va a dar de esto dos conceptos estará enmarcado dentro de su utilidad en aplicaciones que accedan a datos desde archivos o bases de datos (o cualquier otra estructura existente).

Qué es un lenguaje procedimental?
Si desarrollamos una aplicación que acceda a datos de cualquier medio a través de un lenguaje procedimental, debemos ser conscientes que vamos a tener que expresar en las instrucciones del programa, no solamente qué datos vamos a acceder, sino también la manera como los vamos a acceder a ellos. Es decir, al escribir programas en este tipo de lenguajes, debemos estar atentos a especificar, además de los datos que necesitamos procesar, el algoritmo específico para recuperarlos del medio donde se encuentren.
Ejemplos típicos de este tipo de lenguajes son los lenguajes denominados de tercera generación: C, Visual Basic, Perl, entre otros.

Qué es un lenguaje no procedimental?
Con la definición anterior, es fácil deducir qué se entiende por un lenguaje no procedimental. A diferencia de los lenguajes procedimentales, en este tipo de lenguajes solo es necesario especificar dentro de las instrucciones que accedan a datos, qué datos vamos a recuperar. En realidad no nos importa el mecanismo de búsqueda que se vaya a implementar para recuperar los datos. Esto sería problema de "otra persona o entidad". Las instrucciones dentro de un lenguaje no procedimental son más concisas debido a su único requerimiento de especificación de datos a recuperar.
Los lenguajes de acceso a datos no procedimentales se conocen como lenguajes de cuarta generación y en realidad son lenguajes que muchas veces no tienen nombre propio. Esto debido a que son lenguajes que, la mayoría de las veces, vienen incorporados como módulos de los DBMS. (Recuerden que un DBMS consta de dos grandes partes y una de ellas es un conjunto de software que permite optimizar la manipulación de la base de datos. Dentro de este software están los lenguajes de cuarta generación). Hay motores como Oracle y SQL Server que tienen sus lenguajes de cuarta generación denominados PL/SQL y Transact SQL respectivamente. Hay otros motores como Informix que tienen su propio lenguaje de cuarta generación pero se le conoce como "el 4GL de Informix", es decir, no tiene un nombre específico.

Vemos, con base en la exposición anterior, que obviamente los lenguajes

procedimentales son mucho más complejos que los no procedimentales en cuanto al acceso de datos se refiere. Es decir, es mucho más fácil usar un lenguaje no procedimental que uno procedimental si de acceder datos de cualquier medio se trata.

Toda la contextualización anterior sirve para definir que el SQL, que es el lenguaje de consulta más utilizado en bases de datos, es un lenguaje no procedimental. Esto significa que el SQL es un lenguaje de cuarta generación. Lo que sucede es que, a diferencia de otros lenguajes, el SQL es un lenguaje que es aceptado como estándar mundial para acceso a bases de datos relacionales. Del tema de SQL como tal, hablaremos más adelante en este manual y estaremos ahondando en sus características generales y específicas.

Por lo tanto, nos damos cuenta que pedirle al DBMS que nos recupere los datos de una base de datos a través de SQL es una labor muy sencilla. Lo verdaderamente complejo son los mecanismos que debe implementar el DBMS, a través de sus algoritmos internos, para poder acceder a dichos datos de la manera más optima posible.

De lo anterior se desprende que cuando yo invoco una instrucción de consulta en SQL, es el DBMS el responsable de implementar el "como" se van a acceder a los datos requeridos por dicha consulta. Es el DBMS el que haría lo que debería hacer el Analista/Programador si escribiera la misma consulta en un lenguaje procedimental.

Debemos enunciar en este momento el hecho de que los lenguajes de cuarta generación tienen tres tipos de instrucciones:

a. Instrucciones DDL
b. Instrucciones DML
c. Instrucciones DCL

Veamos a continuación, de una manera genérica, a qué se refieren estos tipos de instrucciones.

Qué son instrucciones DDL?
La sigla DDL significa Data Definition Language, es decir, Lenguaje de Definición de Datos. Es a través de este tipo de instrucciones que se pueden definir las estructuras que se van a crear en la base de datos para contener los datos, es decir, son las instrucciones para crear lo que se denominan las tablas de la base de datos. Además, con este tipo de instrucciones se pueden definir reglas de integridad de datos, es decir, normas que deben cumplir los datos para considerarlos correctos dentro del contexto que maneja la base de datos.
Como conclusión podemos decir que el DDL permite definir el ESQUEMA de la base de datos.

Y qué significan instrucciones DML?
DML como sigla significa Data Manipulation Language. Es a través de este tipo de instrucciones que podemos manipular los datos que hay contenidos o grabados en la base de datos. En otras palabras, a través de instrucciones DML se pueden realizar las cuatro operaciones básicas sobre los datos: inserción de nuevos datos, actualización de datos ya existentes, borrado de datos y consulta de datos.

Y el DCL?
La sigla DCL viene de Data Control Language, en otras palabras, Lenguaje de Control de Datos. Que se puede realizar con este tipo de instrucciones? A través de las instrucciones DCL podemos controlar el acceso a los datos, es decir, podemos definir perfiles de usuarios con sus respectivos permisos. Es a través de este tipo de lenguaje que podemos administrar lo que se denomina CONTROL DE ACCESO en el mundo de las bases de datos.

Modelos de Datos y su relación con las Bases de Datos

Hablar de los modelos de datos es hablar de la historia de las bases de datos. Es decir, la explicación puntual de los diferentes modelos de datos que han existido nos da pie para poder explicar, con cierto detalle, como han evolucionado las bases de datos hasta lo que conocemos hoy como bases de datos relacionales.

Antes de entrar a explicar cual es la relación de los modelos de datos con el mundo de las bases de datos, empecemos por definir el concepto "modelo de datos".

Qué es un modelo de datos?

Podemos empezar diciendo que sinónimos de modelo son patrón, ejemplo, formato, "esqueleto", representación, entre otros sinónimos. Es este último sinónimo el que mejor nos describe lo que es un modelo. Es la representación de algo que se quiere mostrar o ejemplificar.
Pues bien, un modelo de datos no es mas que una representación gráfica de 4 elementos que están inmersos en cualquier problema de la vida real que se desee sistematizar informáticamente:

a. Los datos a manejar dentro del problema
b. Las relaciones entre estos datos.
c. La semántica de estos datos, es decir, su significado.
d. Las reglas de integridad, si existieran. Es decir, las normas que deben cumplir estos datos para que sean correctos dentro del problema a solucionar.

Se debe recalcar que todo modelo de datos es gráfico, es decir, el resultado final es un dibujo que describe, en el mejor de los casos, los 4 elementos enunciados anteriormente.

Existen múltiples modelos de datos para diferentes finalidades. La siguiente es una categorización estructural de los modelos de datos, desde el punto de vista de su

construcción. Existen 3 categorías de modelos de datos:

a. Modelos de datos basados en objetos
b. Modelos de datos basados en registros
c. Modelos de datos físicos.

A continuación detallaremos cada uno de los tipos de modelos, explicando su función y detallando qué modelos de datos específicos corresponden a cada categoría.

Modelos de Datos Basados en Objetos

El término objeto en este contexto no es el mismo del término de la programación orientada a objetos. Acá, en este contexto, no hay que ponerle mucho misterio al asunto.

Todos sabemos que sinónimos de objeto son "cosa", "elemento", "ítem", entre otros. Y es precisamente por eso que estos modelos se llaman así. Su construcción está basada en cosas o elementos concretos tales como rectángulos, rombos, elipses, flechas, cuadrados, líneas punteadas, entre otras cosas, dependiendo del modelo del cual estemos hablando. En otras palabras, el dibujo que resulta de hacer un modelo de esta categoría está conformado por un subconjunto de esos objetos mencionados anteriormente.

Dentro de los modelos basados en objetos, se destacan los siguientes:

a. Modelo Entidad / Relación
b. Modelo Orientado a Objetos
c. Modelo Semántico
d. Modelo Funcional

En el ámbito de las bases de datos es muy importante el Modelo Entidad/Relación. Los otros 3 modelos se utilizan en otros contextos. Por eso, a nivel de este manual, nos vamos a concentrar en el primer modelo y para eso nos concentraremos, en el módulo 2, en el tema del Modelo Entidad/Relación.

Modelos de Datos Basados en Registros

Siempre hemos dibujado un registro de la siguiente manera:

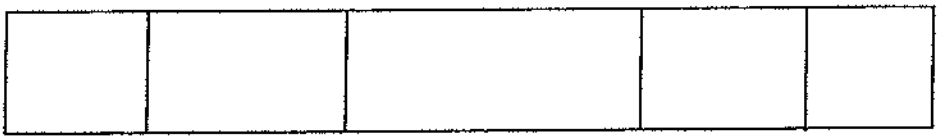

Figura 1. Gráfica Informal de un Registro

Pues bien, los modelos basados en registros se denominan de esa forma porque se dibujan con registros, tal y como se dibujó anteriormente.

Es en esta categorización de los modelos donde podemos abarcar un poco acerca de la historia de las bases de datos y su evolución hasta lo que encontramos hoy con las bases de datos relacionales.

Existen 3 modelos basados en registros y que dan buena cuenta de lo que han sido las bases de datos hasta hoy. Dichos modelos son los siguientes:

a. Modelo Jerárquico
b. Modelo en Red
c. Modelo Relacional

Para poder entender mejor cual es la diferencia entre estos tres modelos y su relación con la historia de las bases de datos, vamos a desarrollar un mismo ejemplo para los 3 modelos.

Cabe anotar que los modelos fueron surgiendo con base en problemas o desventajas que tenían los modelos antecesores. Es decir, el modelo en red nació por la necesidad de solucionar algo del modelo jerárquico; y el modelo relacional, por la necesidad de solucionar una desventaja del modelo en red.

Contextualizando el ejemplo a desarrollar, diremos que vamos a modelar los datos de los estudiantes de una universidad con las materias que está cursando cada estudiante dentro del semestre en curso. Es decir, tenemos dos grandes entes en el ejemplo: ESTUDIANTE y MATERIA.

Modelo Jerárquico

El diagrama siguiente muestra lo que sería el modelo jerárquico para el ejemplo de 5 estudiantes específicos de la universidad.

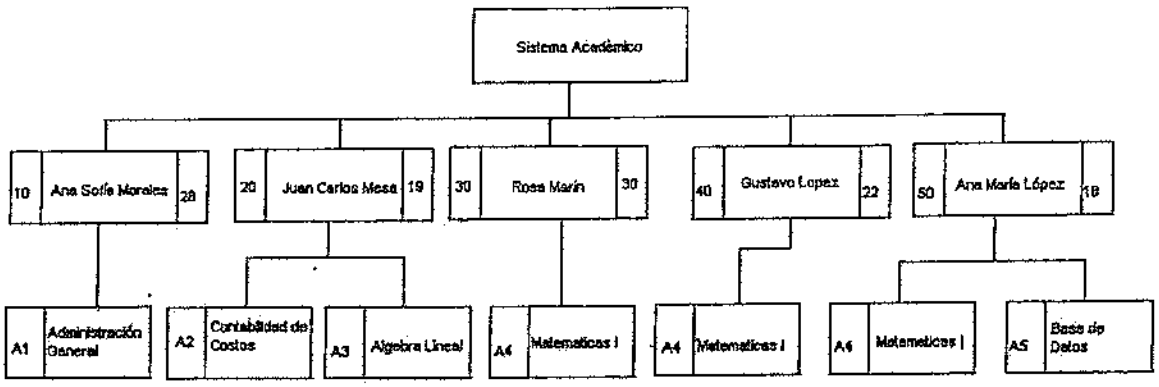

Figura 2. Modelo Jerárquico - Ejemplo Estudiantes - Materias

Por qué se llama modelo jerárquico? Como se puede ver, el modelo está construido en niveles de jerarquía. El esquema con el que se dibuja este modelo es un forma de árbol, es decir, con ramas y hojas. Existe un nodo superior que se denomina nodo raíz y es donde se coloca el nombre de la futura base de datos.
En dicho modelo, un nodo hijo no puede tener más de un nodo padre.
Visualmente es un modelo fácil de analizar. Por ejemplo, si se hiciera la pregunta de cuales son las materias que está cursando actualmente el estudiante Juan Carlos Mesa, es fácil deducir que son Contabilidad de Costos y Álgebra Lineal.
Visualmente es fácil seguir el "camino" de las materias de un estudiante especifico.
Pero así como el modelo tiene sus ventajas, también tiene una gran desventaja.
Cual es? Si UNA MATERIA la están cursando varios estudiantes, el cual es un caso muy común, esta materia se debe dibujar tantas veces como estudiantes la cursen.
En el ejemplo graficado, se ve que el nodo "Matemáticas I" está dibujado 3 veces ya que dicha materia está siendo cursada por Ana María López, Rosa Marín y Gustavo López.
Por lo tanto, la principal desventaja del modelo jerárquico es que produce REDUNDANCIA DE INFORMACIÓN.
En resumen, las principales características del modelo jerárquico son:

- Se guía por jerarquías entre sus nodos
- Existe un nodo principal llamado nodo raíz.
- Visualmente es fácil de interpretar.
- Un nodo hijo no puede tener más de un nodo padre.
- Puede generar, en muchos casos y dependiendo de la información manejada, redundancia de datos.

Las primeras bases de datos que surgieron fueron las bases de datos jerárquicas y su implementación física se basó en las características principales del modelo jerárquico. Este tipo de bases de datos nació iniciando la década de 1960 a 1970.

Modelo en Red

Para solucionar el problema de la redundancia del modelo jerárquico, surgió el

modelo en red. A continuación se presenta el ejemplo de los estudiantes y sus materias, representado por el modelo en red:

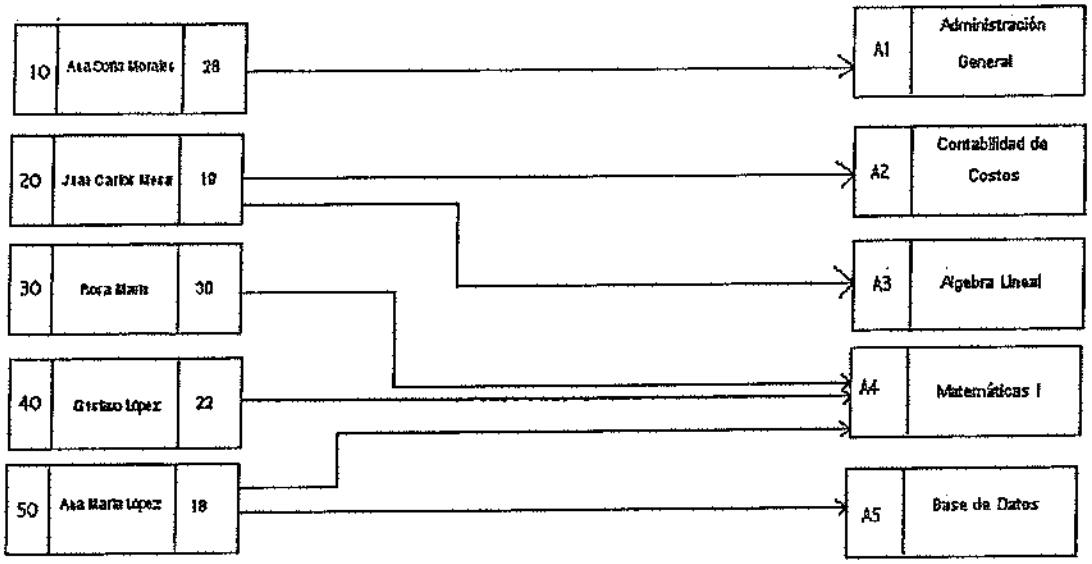

Figura 3. Modelo en Red - Ejemplo Estudiantes - Materias

Como se puede apreciar, el problema de la redundancia de datos quedó superado con este modelo. A pesar de que la materia "Matemáticas I" está siendo vista por 3 estudiantes diferentes, solo aparece un nodo para dicha materia.

Por que este modelo se llama modelo en red? Como se puede apreciar en el ejemplo, existe un conjunto de apuntadores que permiten unir a los estudiantes con las materias que está cursando. Estos apuntadores conforman una red (entiéndase como red el cruce "indiscriminado" de apuntadores en el dibujo).

Si notamos bien, en realidad, visualmente es más fácil realizar una consulta en el modelo jerárquico que en el modelo en red, debido a que si esta red de apuntadores crece, es difícil seguirle el rastro a las materias cursadas por un estudiante.

En resumen, el modelo en red tiene las siguientes características:

- La relación entre registros se da con apuntadores.
- Todos los registros tienen la misma jerarquía.
- El manejo de la red de apuntadores es engorroso, si este tiende a crecer.

El tipo de bases de datos que nacieron con la concepción del modelo en red se llamaron bases de datos en red. El auge de este tipo de bases de datos se dio finalizando la década entre 1960 y 1970.

Modelo Relacional

Con el modelo relacional, los problemas de los modelos jerárquicos y en red desaparecen. Se supera el problema de la redundancia de datos y además se suprime el manejo de apuntadores para la relación entre los datos.

El diagrama que se muestra a continuación es una demostración de lo dicho anteriormente.

Código	Nombre	Edad
10	Ana Sofía Morales	28
20	Juan Carlos Mesa	19
30	Rosa Marín	30
40	Gustavo López	22
50	Ana María López	18

Tabla Estudiante

Código Estudiante	Código Materia
10	A1
20	A2
20	A3
30	A4
40	A4
50	A4
50	A5

Tabla Relación Materia / Estudiante

Código	Nombre de Materia
A1	Administración General
A2	Contabilidad de Costos
A3	Algebra Lineal
A4	Matemáticas I
A5	Bases de Datos

Tabla Materia

Figura 4. Modelo Relacional - Ejemplo Estudiantes - Materias

Como se puede apreciar, cada objeto dentro del problema se esquematiza con una estructura llamada tabla (mas adelante se formalizará este concepto). Y lo que es más importante es que la relación entre los diferentes datos están dados por columnas comunes entre las tablas.
Es cierto que a nivel visual es mucho más fácil encontrar la información en el modelo jerárquico, pero el modelo relacional tiene, para su implementación, una profunda concepción matemática que lo vuelve un modelo muy estable.

El modelo relacional no trabaja con apuntadores ni jerarquías. Solamente lo hace con estructuras más simples de manejar como son las tablas, las filas y las columnas de esas tablas.

En resumen, el modelo relacional sugiere:

- El manejo de tablas, filas y columnas
- Supresión de apuntadores, los cuales hacen engorroso la captura y almacenamiento de datos.
- Relacionar los datos a través de columnas o campos en común.
- Una formalización de su manejo a través de una concepción matemática bien fundamentada.

El modelo relacional, en la década de los años 70, dio pie al nacimiento de las bases de datos relacionales, las cuales cumplen con las características anteriormente anotadas.

En este punto de la exposición del tema, no se ha preguntado usted por qué las bases de datos relacionales han persistido tanto en el tiempo, mientras que las bases de datos jerárquicas y en red solo duraron unos pocos años?

La concepción matemática del modelo relacional hace que sus bases sean muy sólidas y formales, lo cual sugiere que cualquier otra metodología de almacenamiento de datos que desee desplazar al modelo relacional, debe pasar por encima de esta concepción matemática, lo cual es muy difícil. En el módulo referente al modelo relacional, se detallará mucho más en cuales son los conceptos matemáticos involucrados en él.

Modelos de Datos Físicos

Los modelos de datos físicos son aquellos que permiten esquematizar la implementación física del almacenamiento de datos. Es decir, permite mostrar qué estructuras físicas de almacenamiento de datos se van a utilizar. Es un tipo de modelo muy utilizado por el DBA.
Dentro de los modelos de datos físicos se encuentran los siguientes:

a. Modelo de Unificación
b. Modelo de Memoria por Marcos.

La descripción detallada de este tipo de modelos escapa del ámbito del tema de este curso.

Usuarios de un Ambiente de Bases de Datos

En la bibliografía sobre bases de datos, existen múltiples clasificaciones de los usuarios que utilizan esta tecnología. Algunos autores hablan de usuarios sofisticados, usuarios especializados, usuarios normales (o rasos), usuarios finales, analistas, programadores, DBA, entre otros. Y algunas de estas categorías se solapan unas a otras, utilizando denominaciones diferentes.

El siguiente es un esquema en el cual, personalmente, categorizo a los usuarios de una base de datos:

a. Usuarios Finales
b. Analistas
c. DBA

Como ya se expresó en apartados anteriores, estos tres tipos de usuarios son los que definen los niveles de abstracción de datos.

Cabe recordar que el usuario final se incluye dentro del nivel de mayor abstracción, es decir, el nivel de vistas. El Analista se ubica en el nivel de abstracción intermedio

llamado el nivel conceptual (o lógico). Y el DBA es el que se ubica en el nivel físico que es el de menor abstracción de datos.

Lo que es importante dentro de esta categorización es distinguir entre los diferentes "perfiles" que existen de usuarios finales de una base de datos. Y dentro de esta categoría, identificándome con lo expresado por Henry Korth en sus libros de bases de datos, se pueden distinguir tres clases de usuarios finales:

a. Usuario Normal: Es aquel usuario cuya principal característica es que es un usuario neto de la base de datos, es decir, es aquella persona que en todo momento está utilizando la base de datos a través de una aplicación desarrollada exclusivamente por el departamento de sistemas de la empresa. Es decir, es un tipo de usuario con pocos (o a veces ningún) conocimientos técnicos, pero experto en el problema cotidiano que está solucionando la aplicación.
Un ejemplo típico de este tipo de usuario es el operador de datos, el despachador de pedidos, el cajero que factura en un sistema POS, entre otros. (Cabe aclarar que no estoy estereotipando a estos cargos. Simplemente quiero expresar ejemplos de casos vividos en mi experiencia laboral, lo cual no puede tomarse como una generalización).

b. Usuario Especializado: Dentro de esta categoría, encontramos a aquellos usuarios finales que, sin ser técnicos en el área de la informática, tienen unos conocimientos técnicos previos, adquiridos por cursos tomados o inclusive por experiencias anteriores. Este conocimiento adquirido hace que sean usuarios más autónomos e independientes del departamento de sistemas.
Por su conocimiento, son usuarios a los cuales se les podría instalar el lenguaje SQL en sus computadores para ellos mismos hacer consultas sobre la base de datos. En caso de consultas de mayor complejidad, sí recurrirían al departamento de sistemas.
Inclusive, dentro de esta categoría, se pueden incluir aquellos usuarios que han desarrollado pequeñas aplicaciones informáticas para su uso "personal" dentro de sus labores en la empresa.

c. Usuario Sofisticado: En esta categoría, incluimos a las personas que no necesariamente tienen conocimientos técnicos en informática. Lo que los vuelve "sofisticados" es el hecho de que deben utilizar herramientas especiales para poder lograr obtener la información requerida de la base de datos. Por ejemplo, a un alto directivo de una empresa no le basta con que le entreguen, para su labor de toma de decisiones, un listado que muestre las ventas del semestre pasado, discriminado por producto. Debemos anotar que un listado de este tipo, en la mayoría de los casos, tendría un tamaño de 100 hojas o más, lo cual no sería muy práctico para este directivo. Él lo que necesita generalmente es un resumen de las ventas, con proyecciones que le permitan visualizar qué estrategia desarrollar. Para lograr esto, se podría necesitar en algún momento la técnica de la minería de datos, para poder extraer la información requerida de la base de datos y analizarla desde el punto de vista de comportamiento, patrones, tendencias que siguen los datos.
En resumen, el usuario sofisticado es aquel que requiere de un procesamiento

adicional a la información que entrega la base de datos, para qué ésta le pueda ser útil.

El Administrador de la Base de Datos – DBA

Los que hemos trabajado con Bases de Datos, hemos visto que esta tecnología permite que el usuario técnico, en este caso el analista, se desentienda de una serie de cosas que automáticamente realiza el DBMS. Es así como, por ejemplo, cuando el DBMS está en pleno funcionamiento con 50 usuarios accediendo la base de datos a la vez, y de repente se presenta una falla (corte de luz, bloqueo del servidor, problema lógico de una transacción, etc.). En este caso, y como lo veremos en módulos posteriores, el usuario no se tiene que preocupar por la consistencia de la base de datos. Después de que se soluciona el problema que dio origen a la falla, es el DBMS quien automáticamente restaura la base de datos y la deja en un estado consistente, libre de problemas.

Pero para que suceda lo anterior, debe existir una persona que configure el DBMS para que haga esta labor de la mejor manera posible. Esa persona es el DBA.

Qué significa DBA?

DBA viene de la sigla en inglés Data Base Administrator, es decir, Administrador de la Base de Datos. Es conveniente aclarar qué se tiene la creencia con mucha frecuencia de que el DBA es un software que permite controlar el funcionamiento del DBMS. Debo decir que el DBA no es un software, es una persona de carne y hueso como usted señor lector o como yo que tiene una funciones muy definidas dentro del departamento de sistemas de una compañía.

Funciones del DBA

En términos generales, la responsabilidad del DBA es optimizar el funcionamiento y rendimiento del DBMS, para que los usuarios finales del mismo se sientan satisfechos con los tiempos de respuesta.
Dentro de las funciones específicas del DBA, se encuentran las siguientes:

- Sintonizar la Base de Datos: Es talvez la labor más importante del DBA, sin querer decir que las demás que voy a enunciar no sean importantes. Lo que sucede es que a través de esta actividad, en gran medida, el DBA estará logrando el rendimiento óptimo del DBMS. El término sintonizar en inglés se traduce como "tuning" y por eso es muy frecuente oír decir que al DBA le toca hacerle "tuning" a la base de datos.

  Pero qué es sintonizar una base de datos? Seguramente usted, cuando oye la palabra "sintonizar", lo primero que se le viene a la mente es la imagen de un radio transistor. Y es obvio que suceda esto ya que cuando usamos un radio, lo que estamos haciendo es sintonizando emisoras. En este caso,

sintonizar una emisora significa mover el dial hasta el punto en que el sonido que proviene de la emisora se escuche de la mejor manera posible, es decir, qué el sonido sea claro, "nítido", sin interferencias. Por lo tanto, sintonizar una emisora significa mover el dial del radio hasta que el sonido de la emisora sea lo más óptimo posible.
Exactamente lo mismo sucede en una base de datos. Para poder sintonizar una base de datos, el DBA configura una serie de parámetros que son los que dictaminan como debe trabajar el DBMS. Estos parámetros vienen grabados en un archivo que se llama el archivo de configuración del DBMS. Cada vez que el DBMS empieza su ejecución, se guía por los parámetros grabados en este archivo para poder saber como debe empezar a funcionar. Para poder poner un caso, existe un parámetro que le dice al DBMS cuantos usuarios en total va a soportar en forma concurrente, es decir, accediendo a las bases de datos al mismo tiempo. Supongamos que este parámetro tiene el valor de 100 y el DBA se da cuenta que, con el hardware disponible en un momento dado, el comportamiento de la base de datos es muy lento, podría pensar en tomar la decisión de disminuir el valor para este parámetro, ya que teniendo 100 usuarios concurrentemente, el rendimiento de la base de datos se disminuirá.

- Controlar el uso de recursos de hardware: Es deber del DBA estar al cuidado de que, con los parámetros de configuración establecidos en la labor de sintonización de la base de datos, los recursos del hardware estén siendo óptimamente utilizados, es decir, que la memoria RAM pueda alojar temporalmente los diferentes datos de los usuarios, que el procesador pueda realizar su función con todas las transacciones concurrentes que existan, que la fragmentación del disco duro no sea muy alta para que el acceso a los datos sea rápido. Generalmente los DBMS, dentro de sus módulos, incluyen uno que le permite al DBA realizar esta labor, es decir, permite visualizar, generalmente de una manera gráfica, cuanta memoria RAM se está utilizando en un momento dado, cómo están distribuidos los datos dentro del disco duro, en qué porcentaje de capacidad total está trabajando el procesador, etc.

- Realizar backups de la base de datos: Cuando, por alguna circunstancia, se pierde información de la base de datos, es responsabilidad del DBA poder restaurar los datos perdidos. Es decir, poder dejar la base de datos tal y como estaba inmediatamente antes de que se perdieran los datos.
  Para lograr esto, el DBA debe realizar periódicamente copias de seguridad de los datos de la base de datos, copias que se conocen como backups.
  Cabe mencionar que dentro de una base de datos existen 2 tipos de backups: completos e incrementales (o diferenciales). En el siguiente apartado de este módulo se explicará en detalle que son estos backups y cuales son sus diferencias.

- Manejar el control de acceso a la base de datos: El término "control de acceso" en una base de datos es muy importante ya que se refiere a los

perfiles de los diferentes usuarios, es decir, a los permisos que tiene cada usuario para acceder a la base de datos.
Cabe anotar que es responsabilidad del DBA velar porque estos controles de acceso no sean violados. También es su responsabilidad la de configurar el control de acceso dentro de la base de datos, lo cual se logra a través de unas instrucciones SQL que se estudiarán más adelante. Pero cabe explicar que NO es responsabilidad del DBA definir este control de acceso. Es el analista, que es la persona que está desarrollando la aplicación y que conoce a los diferentes usuarios que van a acceder a la base de datos, el que define como debe el DBA configurar físicamente el control de acceso sobre la base de datos.
En resumen, si en algún momento un usuario hace algo indebido sobre la base de datos, es decir, algo sobre lo cual no tenía permiso, es responsabilidad del DBA solucionar esta situación.

Las anteriores son solamente algunas de las principales funciones del DBA, lo cual da una clara muestra de que el nivel de responsabilidad de esta persona dentro del departamento de sistemas es alto.

Quiero aclarar que no comparto el criterio de muchos autores sobre base de datos cuando exponen que una de las funciones del DBA es definir el esquema de la base de datos. Quiero explicar el por qué de mi desacuerdo.
Ya dijimos que el esquema de la base de datos es la estructura de la misma, es decir, es el conjunto de tablas que van a conformar la base de datos, junto con la definición de cada uno de sus campos (tipo de datos, longitud, nulo, etc.). Y es el analista, precisamente con su análisis realizado, el que va a diseñar el esquema de la base de datos. Al fin y al cabo es la persona que ha estado en pleno contacto con el usuario de la base de datos y sus necesidades. Recuerden que el analista está en el nivel conceptual de abstracción de datos, mientras que el DBA está en el nivel físico de abstracción de datos. Cual de estas dos personas, dentro del esquema de niveles de abstracción de datos, está mas cercano al usuario final? Ya sabemos que la respuesta es el analista.
Por lo tanto, considero que NO es responsabilidad del DBA definir el esquema de la base de datos. Sí es su responsabilidad que el esquema definido y creado por el analista, funcione óptimamente dentro del hardware y software con el que se cuenta para ello.

Backups de una Base de Datos

Como ya lo expresé anteriormente, una de las funciones del DBA es realizar los backups de la base de datos y mantenerlos al día.

Para poder mantener al día las copias de seguridad de los datos, el DBA cuenta, en general, con 2 tipos de backups:

- Backup Completo

- Backup Incremental (también llamados diferenciales).

A continuación se explica en detalle qué características tiene cada uno de estos backups y cuales son sus relaciones y diferencias.

Backup Completo

Como su nombre lo indica, el backup completo realiza una copia de TODOS los datos de una base de datos. Es decir, le realiza una copia al EJEMPLAR de la base de datos existente en el momento de hacer la copia (esta última frase ya la debe entender usted, señor lector!).

Backup Incremental

También se le llama diferencial. A diferencia del completo, el backup incremental le hace una copia solo a los movimientos hechos sobre la base de datos, desde el momento en que se le realizó el último backup completo o desde el momento en que se realizó el último backup incremental..
Lo anterior quiere decir que todo backup incremental debe tener asociado un backup completo hecho previamente.

Como funcionan, en la vida real, estos dos tipos de backups?

La responsabilidad del DBA ante la pérdida de datos es poder restaurar la base de datos al estado más próximo al que estaba en el momento de la pérdida.
Para poder explicar lo anterior en el marco de los 2 tipos de backups expuestos, se expone un ejemplo concreto, factible de suceder en la cotidianidad de un departamento de sistemas.

Suponga que el miércoles 6 de septiembre de 2006, Juan Peláez que es el DBA de la empresa XYZ, realiza a las 12:00 del medio día un backup completo de la base de datos. Siendo las 6:00 p.m. del mismo día, sucede un daño y se pierden los datos de la base de datos. Qué puede hacer Juan Peláez? En este caso, él como DBA, debe "montar" el backup completo realizado. Pero y como es posible recuperar las 6 horas de trabajo perdidas, para que lo hecho se refleje en lo recuperado por Juan Peláez? Existe un componente muy importante de toda base de datos que se conoce como el log de transacciones, que es simplemente en archivo físico del sistema operativo en el cual se van grabando todas las acciones que se van realizando sobre la base de datos (en módulos posteriores, se estará explicando este concepto con mayor nivel de detalle). Por lo tanto, después de haber montado el backup completo, Juan Peláez debe aplicarle el log de transacciones hechas después de las 12:00 del medio día. Con esta operación, la base de datos quedará como si nunca se hubieran perdido los datos.

Sigamos con el caso anterior. Después de restaurar la base de datos, Juan Peláez vuelve y realiza un backup completo a las 6:30 p.m. Al finalizar el día siguiente, 6:00 p.m. del 7 de septiembre de 2006, el DBA hace un backup incremental. Y al

empezar el día siguiente, 8:00 a.m. del 8 de septiembre de 2006, Juan Peláez se encuentra con que la base de datos ha sufrido un daño por un mal arranque del servidor. Como recupera Juan la base de datos en este caso? Lo que se debe hacer es restaurar el último backup completo realizado, es decir, el del 6 de septiembre de 2006 a las 6:30 p.m. En este caso, la base de datos quedará con un día entero de atraso. Pero como existe un backup incremental hecho el día anterior al finalizar el día, es factible montar sobre el backup completo recién restaurado, el último backup incremental realizado para así dejar la base de datos como estaba en el momento del fallo. (En estos ejemplos estoy suponiendo que Juan Peláez trabaja en una empresa que no labora en las horas de la noche sobre la base de datos).

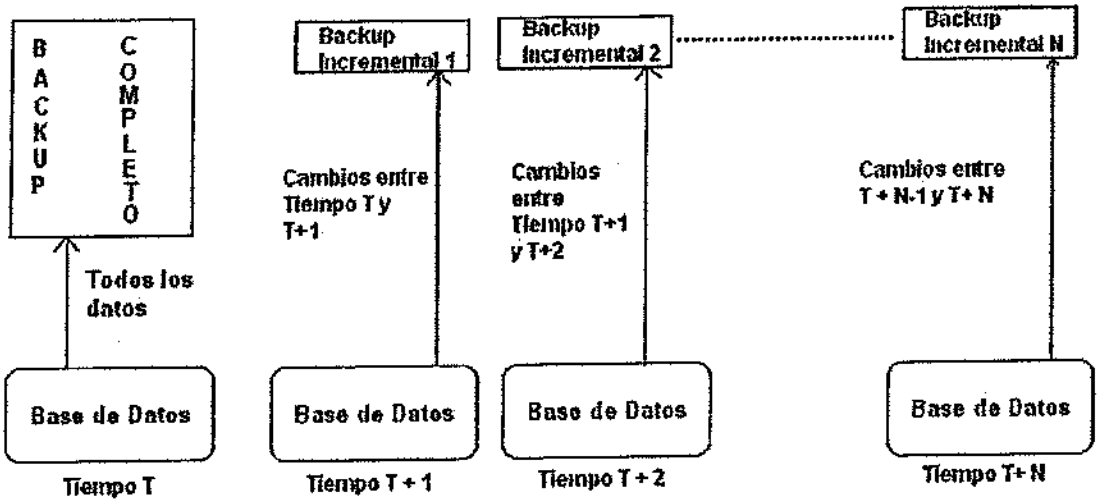

Figura 5. Relación entre Backup Completo e Incremental

De lo anterior podemos deducir que es una política común hacer un backup completo de la base de datos cada semana y backups incrementales de la misma todos los días al finalizar la jornada. Esta política, como se ilustra en la figura No. 5, permite que el DBA cumpla con su función plenamente.