UNAPEC UNIVERSIDAD APEC

Fundamentos de Base de Datos I

Ing. Francisco Jose Santana V.

II- Conceptos de un Sistema de Base de Datos (DBMS o SGBD) e Introducción a los Modelos de Datos.

El sistema de gestión de bases de datos es esencial para el adecuado funcionamiento y manipulación de los datos contenidos en la base de datos. Se puede definir como: "El Conjunto de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad".

Un poco de historia

La aparición de los SGBD fue fruto de la necesidad de cambiar el concepto de almacenamiento de datos. Antes de los SGBD (década de los setenta), la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo.

Éstos consistían en conjunto de programas que definían y trabajaban sus propios datos.

Este sistema presentaba diferentes inconvenientes:

- Redundancia e inconsistencia de los datos.
- Dificultad para tener acceso a los datos.
- Separación y aislamiento de los datos.
- Problemas en la seguridad de los datos.

Pero quizá el mayor problema que presentaba la gestión de archivos era la dependencia de la estructura del fichero con el programa. Puesto que la estructura del fichero dependía directamente del programa que lo gestionaba, en el momento que se cambiara esa estructura había que adaptar el propio programa y volver a compilar, lo que entonces llevaba largo tiempo.

Arquitectura

Fue en 1975, en el comité ANSI-SPARC (American National Standard Institute – Standards Planning and Requeriments Committee), cuando se propuso una arquitectura de tres niveles para los DBMS cuyo objetivo principal era separar la BD física de los programa de aplicación.

Los principales componentes del gestor de la base de datos son los siguientes:

- *Control de autorización*. Este módulo comprueba que el usuario tiene los permisos necesarios para llevar a cabo la operación que solicita.

- *Procesador de comandos.* Una vez que el sistema ha comprobado los permisos del usuario, se pasa el control al procesador de comandos.

- *Control de la integridad.* Cuando una operación cambia los datos de la base de datos, este módulo debe comprobar que la operación a realizar satisface todas las restricciones de integridad necesarias.

- *Optimizador de consultas.* Este módulo determina la estrategia óptima para la ejecución de las consultas.

- *Gestor de transacciones.* Este módulo realiza el procesamiento de las transacciones.

- *Planificador (scheduler).* Este módulo es el responsable de asegurar que las operaciones que se realizan concurrentemente sobre la base de datos tienen lugar sin conflictos.

- *Gestor de recuperación.* Este módulo garantiza que la base de datos permanece en un estado consistente en caso de que se produzca algún fallo.

- *Gestor de buffers.* Este módulo es el responsable de transferir los datos entre memoria principal y los dispositivos de almacenamiento secundario. A este módulo también se le denomina *gestor de datos*.

Tipos de Interfaces en DBMS

Interfaces basadas en menú
Interfaces gráficas
Interfaces basadas en formas
Interfaces de lenguaje natural
Interfaces para usuarios paramétricos
Interfaces para el DBA

Lenguajes e Interfaces de Bases de Datos:

Anteriormente discutimos la variedad de usuarios existentes en un DBMS, el manejador debe proveer lenguajes e interfaces apropiados para categoría de usuarios. En esta parte discutiremos los tipos de lenguajes e interfaces que provee un DBMS y las categorías de usuarios que tocan cada uno.

Lenguajes en DBMS

Una vez que el diseño de una base de datos es completado y un manejador es seleccionado para implementarla, la primera tarea es especificar esquemas internos y conceptuales para la base de datos y los mappings entre ambos. En los manejadores en los cuales no hay una separación estricta entre los niveles, un lenguaje llamado Data Definition Language (DDL), es utilizado por DBA y los diseñadores para definir ambos

esquemas. El manejador tendrá un compilador DDL cuya función es procesar las instrucciones DDL para identificar las descripciones de las construcciones de esquemas y almacenar las descripciones de esquema en el catálogo.

En los manejadores en los cuales existe una clara separación entre los niveles conceptuales e internos, el DDL es utilizado para especificar solamente el esquema conceptual. Otro lenguaje llamado **Storage Definition Language** (SDL) es utilizado para especificar el esquema interno.

Para una verdadera arquitectura de tres esquemas necesitamos un tercer lenguaje, **View Definition Language** (VDL), para especificar las imágenes o visiones del usuario y los mappings al esquema conceptual. La mayoría de los sistemas que soportan imágenes externas también proveen una variación del DDL o extienden el DDL con instrucciones para definición de imágenes. Las instrucciones de los lenguajes en DDL pueden estar contenidas en un lenguaje de programación de propósito general, o pueden ser compiladas por separado. En el caso anterior, instrucciones DDL deben ser identificadas como parte del programa, de manera que puedan ser extraídas y procesadas por le compilador DDL.

Una vez el esquema es compilado y la base de datos poblada con información, los usuarios disponen varias formas de manejar la base de datos. Las operaciones de manejo típicas incluyen acceso, inserción, eliminación de los datos. El DBMS provee un Data Manipulation Language (DML) para estos propósitos.

- **Compilador de DLL (Data Definition Language):**

  - Es un lenguaje de definición de datos.
  - Al estar la base de datos divida en niveles el lenguaje DLL nos debe de proporcionar acceso a nivel físico y a nivel conceptual.
  - Dentro del DLL encontraremos el DDSL que nos permite definir y almacenar datos.
  - El DLL nos permite definir las entidades y las relaciones entre las mismas.

- **Diccionario de datos:**

  - Contiene información de cómo se definen o almacenan los datos. Se dice que contiene metadatos.
  - El diccionario lo podemos generar con DLL. Ejemplo: Utilización del DLL.

Desde el punto de vista del modelo relacional:

SQL:
```
create table cliente
(nombre var char(15),
ciudad var char(10),
numero_cuenta number);
```

Existen dos clases principales de DMLs. Un DML de **Alto Nivel** o **No-procedimental** puede ser utilizado para especificar operaciones complejas de bases de datos de una manera concisa. Un DML de **Bajo Nivel o Procedimental** debe estar unido a un lenguaje de programación de propósito general. Este tipo de DML típicamente acceder a records individuales de la BD y procesa cada record por separado. Por lo tanto necesita hacer uso de estructuras propias de un lenguaje de programación, como bucles, para acceder y procesar cada registro individual de un conjunto. Los DMLs de bajo nivel son llamados además **registro-a-la-vez**, por esta propiedad. Los DMLs de alto nivel pueden accesar y especificar varios registros en una simple instrucción de DML, y son usualmente llamados **conjunto-a-la-vez u orientado-a-conjuntos**. Una consulta (query) en un DML de alto nivel comúnmente específica cuales datos deben ser accesados en vez de como deben accesados, por esta razón los lenguajes anteriores son llamados **declarativos**.

Siempre que los comandos de DML, sean estos de bajo o alto nivel, están contenidos en un lenguaje de programación de propósito general, ese lenguaje es llamado **Host Language** (Lenguaje Anfitrion) y el DML llamado Sub-lenguaje de Datos. Por otro lado, un DML de alto nivel utilizado por sí solo en forma interactiva es llamado (lenguaje de consulta).

Clasificación de los DBMS

El criterio principal utilizado para clasificar los DBMSs es el modelo de datos en el cual esta basado el manejador. Los modelos de datos más usados comúnmente en bases de datos comerciales son el relacional, jerárquico y de red. Algunas bases de datos recientes están basadas en modelos conceptuales u orientados a objetos.

Segundo criterio usado para clasificar los DBMSs es el número de usuarios soportado por el manejador. Los sistemas de un sólo usuario, utilizados en su mayoría en computadoras personales y sistemas multiusuarios.

Un **Tercer criterio** es el número de sitios en los que está distribuida la base de datos. Casi todos los SGBD son *centralizados*: sus datos se almacenan en un solo computador. Los SGBD centralizados pueden atender a varios usuarios, pero el SGBD y la base de datos en sí residen por completo en una sola máquina. En los SGBD *distribuidos* la base de datos real y el propio software del SGBD pueden estar distribuidos en varios sitios conectados por una red. Los SGBD *distribuidos homogéneos* utilizan el mismo SGBD en múltiples sitios. Una tendencia reciente consiste en crear software para tener acceso a varias bases de datos autónomas preexistentes almacenadas en SGBD *distribuidos heterogéneos*. Esto da lugar a los SGBD *federados* o *sistemas multibase de datos* en los que los SGBD participantes tienen cierto grado de autonomía local. Muchos SGBD distribuidos emplean una arquitectura cliente-servidor.

Un cuarto criterio es el costo, la mayoría de los paquetes de DBMS cuestan entre US$10,000 y US $100,000. Los sistemas para un usuario entre US $100 y US $3000. Por otro lado paquetes muy elaborados entre US $100,000 y US $300,000.

Por último, los SGBD pueden ser de *propósito general* o de *propósito específico*. Cuando el rendimiento es fundamental, se puede diseñar y construir un SGBD de

propósito especial para una aplicación específica, y este sistema no sirve para otras aplicaciones. Muchos sistemas de reservas de líneas aéreas son SGBD de propósito especial y pertenecen a la categoría de *sistemas de procesamiento de transacciones en línea* (OLTP), que deben atender un gran número de transacciones concurrentes sin imponer excesivos retrasos.

Finalmente discutiremos de forma breve el criterio principal para clasificación que es el modelo de datos. El modelo relacional representa la base de datos como una colección de tablas, que lucen como archivos. La mayoría de las base de datos relacionales tienen lenguajes de consulta de alto nivel y soportan una limitada forma de visiones de usuarios. Usualmente el esquema conceptual e interno no son diferenciales, y un DDL sencillo es utilizado para describir todos los aspectos de la estructura de la BD.

SEGURIDAD E INTEGRIDAD DE LOS DATOS

La información almacenada en la base de datos debe estar protegida contra accesos no autorizados, destrucción o alteración con fines indebidos y la introducción accidental de inconsistencia. Anteriormente vimos cómo preservar la integridad en sistemas distribuidos. Hasta ahora hemos considerado solamente cómo prevenir la pérdida accidental por falta de integridad de la información. En lo adelante examinaremos las formas en las que se produce un mal uso de la información o que ésta se vuelva inconsistente de forma intencionada.

VIOLACIONES DE LA SEGURIDAD E INTEGRIDAD

El mal uso que se haga de la base de datos puede ser intencionado (con fines indebidos) o accidental. La pérdida accidental de la consistencia de los datos puede deberse a:

- Caídas durante el procesamiento de las transacciones.
- Anomalías por acceso concurrente a la base de datos.
- Anomalías que resultan de la distribución de los datos entre varios computadores.
- Un error lógico que viola la suposición de que la transacciones respetan las protecciones de consistencia de la base de datos.

Es más fácil prevenir la pérdida accidental de la consistencia de los datos que prevenir el acceso mal intencionado a la base de datos. Algunas de las formas de acceso indebido son las siguientes:

- Lectura de datos sin autorización (robo de información).
- Modificación de datos sin autorización.
- Destrucción no autorizada de los datos.

No es posible proteger de manera absoluta a la base de datos contra un manejo indebido, pero puede hacerse que el coste para el autor sea tan alto que frene prácticamente todos los intentos de acceder a la base de datos sin la autorización debida. El término *seguridad de la base de datos* normalmente se refiere a la protección contra el acceso al intencionado, mientras que *integridad* se refiere a la

protección contra una pérdida accidental de consistencia. En la práctica, la línea que separa la seguridad de la integridad no siempre está bien definida. Aquí sí utilizaremos el término *seguridad* para referirnos tanto a la *seguridad* como a la *integridad* en los casos que la distinción entre estos dos conceptos no sea esencial.

Para proteger a la base de datos es necesario adoptar medidas de seguridad en varios niveles:

- **Físico**: La localidad o localidades que contienen a los sistemas de computadores deben protegerse físicamente contra la penetración armada o clandestina de intrusos.

- **Humano**: Debe tenerse mucho cuidado al conceder autorización a los usuarios para reducir la probabilidad de que un usuario autorizado permita el acceso a un intruso de sobornos u otros favores.

- **Sistema operativo**: Aunque el sistema de base de datos esté bien protegido, si no se protege de forma adecuada el sistema operativo éste puede servir para obtener acceso sin autorización a la base de datos. Dado que casi todos los sistemas de base de datos permiten acceso remoto a través de terminales o redes, la seguridad a nivel software dentro del sistema operativo es tan importante como la seguridad física.

- **Sistemas de base de datos:** Puede darse el caso de que algunos usuarios estén a autorizados sólo para tener acceso a una porción limitada de la base de datos. Es posible también que a algunos usuarios se les permita hacer consultas, pero se les prohíbe modificar la base de datos. El sistema de base de datos tiene la responsabilidad de garantizar que no se violen estas restricciones.

  La seguridad en todos los niveles anteriores debe mantenerse para garantizar la seguridad de la base de datos. Un punto débil en un nivel bajo de seguridad a alto nivel (base de batos).

  En muchas aplicaciones vale la pena dedicar un esfuerzo considerable a la conversación de la seguridad e integridad de la base de datos.

Las bases de datos grandes que contienen información de nóminas u otro tipo de datos financieros, son un blanco tentador para los ladrones. Las bases de datos que contienen información referente a las operaciones de una compañía pueden ser de interés para competidores sin escrúpulos. Además, la pérdida de este tipo de información, ya sea accidental o fraudulenta, puede afectar seriamente la capacidad de funcionamiento de una compañía.

En lo que resta, hablaremos de seguridad a nivel de la base de datos. Aunque la seguridad en el nivel físico y humano es muy importante, queda más allá del propósito de este libro. La seguridad dentro del sistema operativo se implanta en varios niveles que van desde palabras clave para tener acceso al sistema hasta el aislamiento de procesos concurrentes que se ejecutan en el sistema. El sistema de archivos proporciona también cierto grado de protección. El presente análisis de la seguridad lo

haremos en términos del modelo relacional, aunque los conceptos de esta teoría se pueden aplicar por igual a todos los modelos de datos.

Metodología de diseño de bases de datos

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

El diseño conceptual parte de las especificaciones de requisitos de usuarios y su resultado es el esquema conceptual de la base de datos. Un ***esquema conceptual*** es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para manipularla. Un ***modelo conceptual*** es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un ***esquema lógico*** es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un ***modelo lógico*** es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un *esquema físico* es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos.

Modelos de Datos

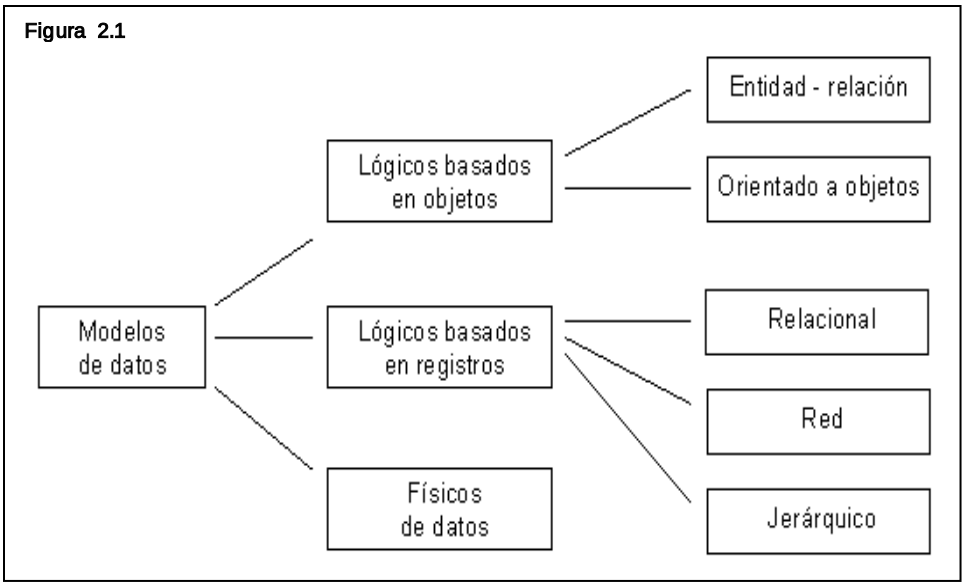
Es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciónes de consistencia. Los diversos modelos de datos que se han propuesto se dividen en tres grupos: modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos de datos.

Otra definición de Modelo de Datos es un conjunto de conceptos que pueden ser utilizados para describir la estructura de una base de datos.

Los principales objetivos del proceso de modelamiento es saber identificar cuál es el problema y encontrar la forma de representarlo en un sistema. Esto significa saber de los datos, saber quienes van a usarlos y como van a ser usados.

La siguiente figura muestra los tipos de modelos de datos.

Figura 2.1



Modelos lógicos basados en objetos:

Los modelos lógicos basados en objetos se usan para describir datos en los niveles conceptuales y de visión. Se caracterizan por el hecho de que proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explicitamente. Hay muchos modelos diferentes entre ellos:

- El modelo entidad-relación
- El modelo Orientado a objetos.

Modelos lógicos basados en registros:

Los modelos lógicos basados en registros se utilizan para describir datos en los modelos conceptual y físico. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los modelos basados en registros se llaman asi porque la base de datos está estructurada o compuesta por tablas y estas a su vez por registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos y cada campo normalmente es una longitud fija.

Los modelos de datos más ampliamente aceptados son:

- El modelo Relacional
- El modelo de Red
- El modelo jerárquico

El **modelo relacional** será nuestra base de estudio en la materia y los demás modelos los veremos en capitulos mas adelante.

Las bases de datos relacionales son el tipo de bases de datos actualmente más difundido. Los motivos de este éxito son fundamentalmente dos:

1. ofrecen sistemas simples y eficaces para representar y manipular los datos
2. se basan en un modelo, el relacional, con sólidas bases teóricas

El modelo relacional fue propuesto originariamente por E.F. Codd en un ya famoso artículo de 1970. Gracias a su coherencia y facilidad de uso, el modelo se ha convertido en los años 80 en el más usado para la producción de DBMS.

Ejemplo del modelo relacional:

Empleado

Codigo	Nombre	Dept	Telefono
0001	Pedro Santana	55	555-5555
0002	Manuel Montero	44	222-2222
0003	Jose Paulino	33	111-1111

Departamento

Dept	Nombre	Fecha C.
44	Administracion	01/01/2001
55	Computos	01/01/2002
33	Contabilidad	01/01/2001

Los modelos de alto nivel utilizan conceptos tales como, entidad, atributos y relaciones.

El principal concepto del modelo ER es la **entidad,** que es una "cosa" en el mundo real con existencia independiente. Una entidad puede ser un objeto físico (una persona, un auto, una casa o un empleado) o un objeto conceptual (una compañía, un puesto de trabajo o un curso universitario). En nuestro ejemplo de la sección anterior podemos definir dos entidades: Empleado y Departamento. Las entidades las podemos clasificar en:

Regulares: aquellas que existen por sí mismas y que la existencia de un ejemplar en la entidad no depende de la existencia de otros ejemplares en otra entidad. Por ejemplo "EMPLEADO", "PROFESOR". La representación gráfica dentro del diagrama es la siguiente:

Débiles: son aquellas entidades en las que se hace necesaria la existencia de ejemplares de otras entidades distintas para que puedan existir ejemplares en esta entidad. Un ejemplo sería la entidad "ALBARÁN" que sólo existe si previamente existe el correspondiente pedido.

Como complemento al diagrama de entidades del modelo de datos, podemos utilizar la siguiente plantilla para definir las diferentes entidades:

Nombre	**PROFESOR**
Objeto	Almacenar la información relativa de los profesores de la organización.
Alcance	Se entiende como profesor a aquella persona que, contratada por la organización, imparte, al menos, un curso dentro de la misma.
Número de ejemplares	10 profesores
Crecimiento previsto	2 profesores / año
Confidencialidad	1. Nombre y apellidos: Acceso público. 2. Datos personales: Acceso restringido a secretaría y dirección. 3. Salario: Acceso restringido a dirección.
Derechos de Acceso	Para garantizar la total confidencialidad de esta entidad, el sistema de bases de datos deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	Los ejemplares dados de baja no serán eliminados de la base de datos; pasarán a tener una marca de eliminado y no serán visualizados desde la aplicación.

Cada entidad tiene propiedades específicas, llamadas *atributos*, que la describen; podemos decir tambien **Un Atributo** es una propiedad que describe algun aspecto de un objeto. Por ejemplo, una sala de clases tiene un nombre (1-28S), una ubicación, un cupo máximo, etc. En nuestro ejemplo, la entidad "Empleado" posee los atributos nombre y No_empleado. Una entidad particular tiene un valor para cada uno de sus atributos.

Cada uno de los atributos de una entidad posee un dominio, el que corresponde al tipo del atributo. Por ejemplo, "No_empleado" tiene como dominio al conjunto de los enteros positivos y "nombre" tiene como dominio al conjunto de caracteres.

Para todo conjunto de valores de una entidad, debe existir un atributo o combinación de atributos, que identifique a cada entidad en forma única. Este atributo o combinación de atributos se denomina llave (pimaria). Por ejemplo, el número de matricula es una buena llave para la entidad alumno, no así el nombre, porque pueden existir dos personas con el mismo nombre.

La descripción de la bases de datos es llamada **Esquema de la Base de Datos**. **Un esquema** es especificado durante el diseño de la base de datos y no esta supuesto a cambiar frecuentemente. La representación de un esquema es llamado diagrama de esquema, el cual representa la estructura de cada archivo, pero no los registros que contiene. Cada objeto en el esquema, como estudiante o asignatura en el ejemplo nuestro, es llamado construcción de esquema.

La coleccion de información almacenada en las bases de datos en un momento determinado es llamada **instancia.** Varias instancias pueden corresponder a un esquema particular. Distinguir entre un esquema y una instancia en base de datos es sumamente importante. Cando definimos una nueva base de datos solo le

especificamos su esquema al manejador, en este punto la instancia corresponde a una "instancia vacia" si datos.

Ejemplo: Diagrama de esquema para una estructura en una Base de Datos:

Matricula	Nombre	Apellido	Fecha_nac	Sexo	Direccion	Telefono

Ejemplo de Una Instancia

Matricula	Nombre	Apellido	Fecha_nac	Sexo	Direccion	Telefono
56253	Manuel	Almonte	01/01/2003	M	C/2 #2	222-2222