2020

GRUPO STUDIUM ciclos

Tema 1: Temario

INTRODUCCIÓN A LAS BASES DE DATOS

STUDIUM

www.grupostudium.com
informacion@grupostudium.com
954 539 952

1.1 Introducción

Una **base de datos** o **banco de datos** (en ocasiones abreviada con la sigla *BD* o con la abreviatura *b. d.*) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la **informática** y la **electrónica**, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Existen programas denominados **Sistemas Gestores de Bases de Datos**, abreviado SGBD, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo, en España los datos personales se encuentran protegidos por la **Ley Orgánica de Protección de Datos de Carácter Personal** (LOPD).

1.2 Tipos de Base de Datos

Las **bases de datos** pueden clasificarse de varias maneras, de acuerdo con el contexto que se esté manejando, la utilidad de las mismas o las necesidades que satisfagan.

Según la variabilidad de los datos almacenados

Bases de datos estáticas

Son bases de datos de **sólo lectura**, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

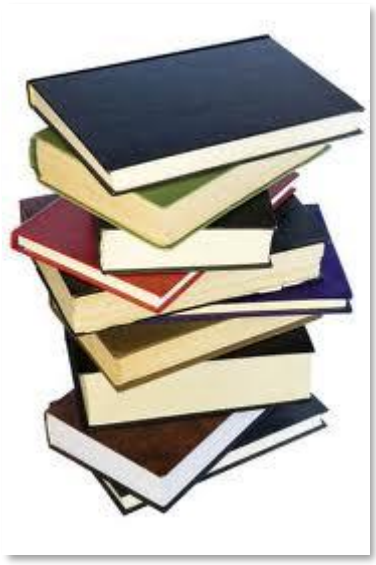
Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado, una farmacia, un videoclub o una empresa.

Según el contenido

Bases de datos bibliográficas

Sólo contienen un **subrogante** (representante) de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque si no, estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias —ver más abajo). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.



Bases de datos de texto completo

Almacenan las fuentes primarias, como, por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

Directorios

Un ejemplo son las guías telefónicas en formato electrónico.

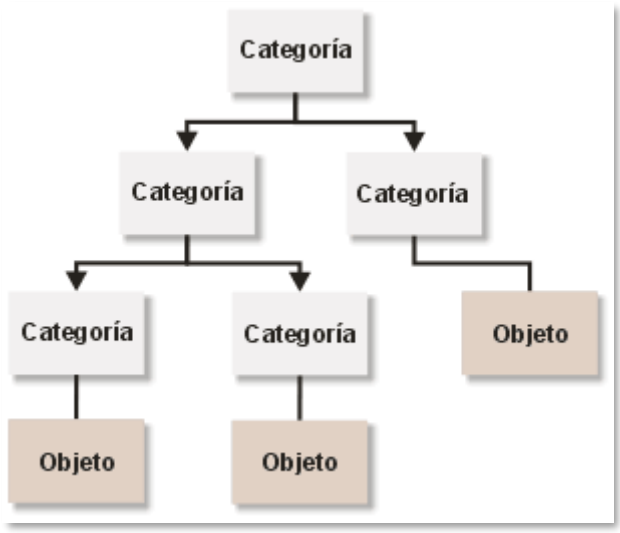
1.3 Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo con su **modelo** de administración de datos.

Un **modelo de datos** es básicamente una "descripción" de algo conocido como *contenedor de datos* (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de *base de datos*; por lo general se refieren a algoritmos, y conceptos matemáticos. Algunos modelos con frecuencia utilizados en las bases de datos:

Bases de datos jerárquicas

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*.



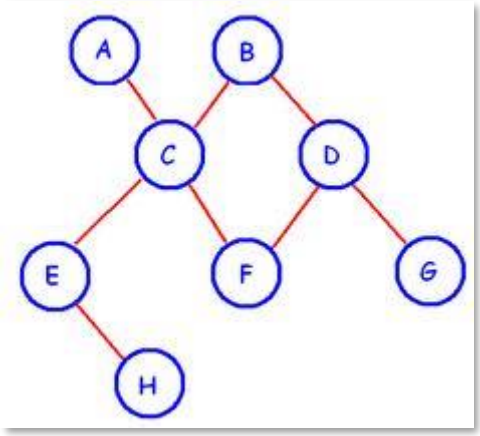
Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

Base de datos en red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de *nodo*: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

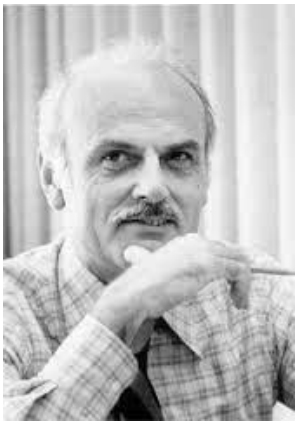
Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.



Bases de datos transaccionales

Son bases de datos cuyo único fin es el **envío y recepción** de datos a grandes velocidades, estas bases son muy poco comunes y están dirigidas por lo general al entorno de **análisis de calidad, datos de producción e industrial**. Es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto, la redundancia y duplicación de información no es un problema como con las demás bases de datos, por lo general para poderlas aprovechar al máximo permiten algún tipo de conectividad a bases de datos relacionales. Un ejemplo habitual de transacción es el traspaso de una cantidad de dinero entre cuentas bancarias. Normalmente se realiza mediante dos operaciones distintas, una en la que se decrementa el saldo de la cuenta origen y otra en la que incrementamos el saldo de la cuenta destino. Para garantizar la atomicidad del sistema (es decir, para que no aparezca o desaparezca dinero), las dos operaciones deben ser atómicas, es decir, el sistema debe garantizar que, bajo cualquier circunstancia (incluso una caída del sistema), el resultado final es que, o bien se han realizado las dos operaciones, o bien no se ha realizado ninguna.

Bases de datos relacionales



Éste es el modelo utilizado en la **actualidad** para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por **Edgar Frank Codd**, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "**relaciones**". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "**tuplas**". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una **tabla** que está compuesta por ***registros*** (las filas de una tabla), que representarían las tuplas, y ***campos*** (las columnas de una tabla).

Empleados Consulta

IDEmpleadc	FechaIngres	Nombre	Apellido	Departamer	Salario
12345	01/02/2012	Moises	Ortiz	Informática	$45,000.00
12346	02/02/2012	Darío	Ventura	Finanzas	$35,000.00
12347	03/02/2012	Alejandra	Medina	Marketing	$29,000.00
12348	06/02/2012	Teresa	Hernández	Producción	$25,000.00
12349	07/02/2012	Karen	Alvarez	Finanzas	$58,000.00
12350	08/02/2012	Gustavo	Romero	Informática	$39,000.00
12351	09/02/2012	Manuel	Calderón	Finanzas	$49,000.00
12352	10/03/2012	José	Márquez	Marketing	$42,000.00
12353	13/02/2012	Carlos	Menchaca	Producción	$39,000.00
12354	14/02/2012	Vanesa	Díaz	Finanzas	$26,000.00

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es **SQL**, *Structured Query Language* o *Lenguaje Estructurado de Consultas*, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como **normalización de una base de datos**.

www.grupostudium.com
informacion@grupostudium.com
954 539 952

Durante los años 80 la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

Bases de datos multidimensionales

Son bases de datos ideadas para desarrollar aplicaciones muy concretas, como creación de **Cubos OLAP**. Básicamente no se diferencian demasiado de las bases de datos relacionales (una tabla en una base de datos relacional podría serlo también en una base de datos multidimensional), la diferencia está más bien a nivel conceptual; en las bases de datos multidimensionales los campos o atributos de una tabla pueden ser de dos tipos, o bien representan dimensiones de la tabla, o bien representan métricas que se desean estudiar.

Bases de datos orientadas a objetos

Este modelo, bastante reciente, y propio de los **modelos informáticos orientados a objetos**, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulación** - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia** - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo** - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación

(llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

Bases de datos documentales

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes. **Tesaurus** es un sistema de índices optimizado para este tipo de bases de datos.

Bases de datos deductivas

Un sistema de base de datos **deductiva** es un sistema de base de datos, pero con la diferencia de que permite hacer deducciones a través de **inferencias**. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. Las bases de datos deductivas son también llamadas **bases de datos lógicas**, a raíz de que se basan en lógica matemática. Este tipo de base de datos surge debido a las limitaciones de la Base de Datos Relacional de responder a consultas recursivas y de deducir relaciones indirectas de los datos almacenados en la base de datos.

Ventajas:

- Uso de reglas lógicas para expresar las consultas
- Permite responder consultas recursivas
- Cuenta con negaciones estratificadas
- Capacidad de obtener nueva información a través de la ya almacenada en la base de datos mediante inferencia
- Uso de algoritmos de optimización de consultas
- Soporta objetos y conjuntos complejos

Desventajas:

- Crear procedimientos eficaces de deducción para evitar caer en bucles infinitos
- Encontrar criterios que decidan la utilización de una ley como regla de deducción
- Replantear las convenciones habituales de la base de datos

Bases de datos NoSQL

Estas bases de datos son la últimas en aparecer y han irrumpido de la mano del Big Data y del abaratamiento tanto del almacenamiento como del procesamiento de datos.

Se denominan NoSQL pues no siguen el protocolo más usado hoy en día que es el SQL o bases de datos relacionales.

En estas bases de datos lo que prima es la rapidez en obtener los datos, a pesar de tener cierto grado de error en los resultados, pero que se considera admisible.

Por ejemplo, si hacemos una búsqueda en el catálogo de Amazon, queremos obtener datos de forma rápida, aunque salgan algunos resultados que no tengan nada que ver con lo que he buscado.

Existe mucha **redundancia**, pero siempre tolerable pues los sistemas de almacenamiento cada vez son más grandes, más económicos y también más eficientes.

1.4 Gestión de bases de datos distribuidas

La base de datos y el software SGBD pueden estar distribuidos en múltiples sitios conectados por una red. Hay de dos tipos:

1. Distribuidos homogéneos: utilizan el mismo SGBD en múltiples sitios.

2. Distribuidos heterogéneos: Da lugar a los SGBD federados o sistemas multibase de datos en los que los SGBD participantes tienen cierto grado de autonomía local y tienen acceso a varias bases de datos autónomas preexistentes almacenados en los SGBD, muchos de estos emplean una arquitectura cliente-servidor.

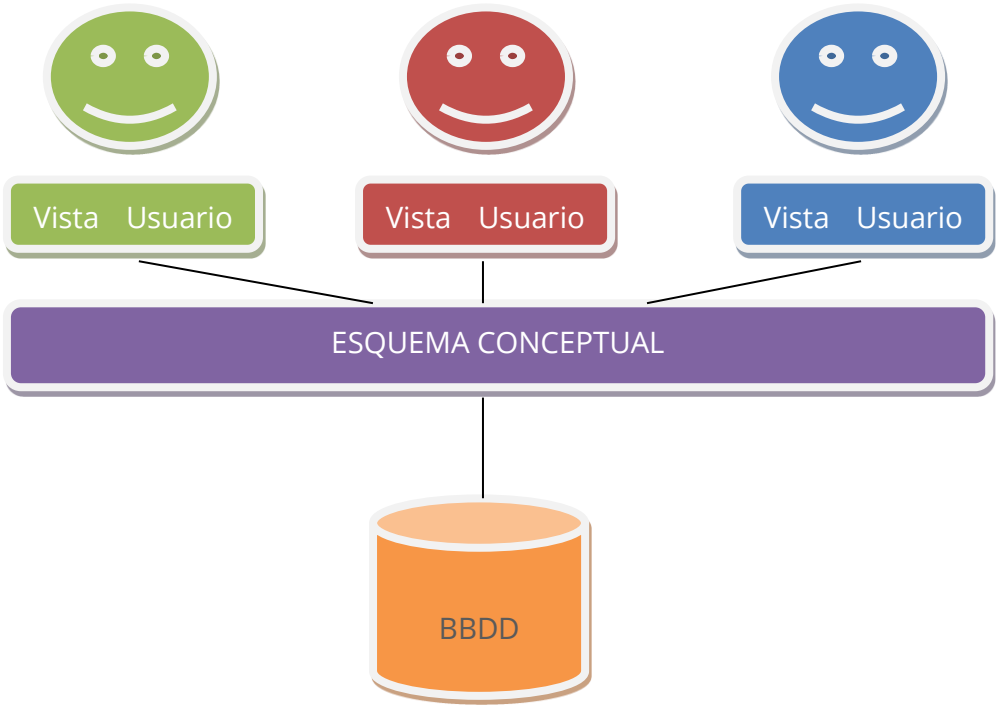
Estas surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades, sucursales de tiendas, etcétera.

1.5 Arquitectura de un SGBD

Todos los gestores mencionados hasta ahora cumplen una arquitectura estándar denominada **ANSI-SPARC**, acrónimo de American National Standards Institute – Standards Planning and Requirement Committee. Aunque este estándar nunca se formalizó, contiene un modelo tan abstracto que se puede aplicar a cualquier gestor de bases de datos corporativas. Este modelo separa la visión que tiene el usuario de una base de datos de cómo está implementada a nivel software. De este modo, se aísla la complejidad del sistema de almacenamiento de datos y el usuario puede entender la forma de trabajar con los datos de forma más intuitiva.

Para aislar esta complejidad, el sistema se divide en 3 capas:

- **La capa interna o física**: Describe cómo los datos son almacenados en los dispositivos hardware del ordenador. Tiene como objetivos disminuir los tiempos de respuesta, el espacio de almacenamiento y optimizar el consumo de recursos, entre otros.
- **La capa conceptual**: Representa una forma de describir la información que se almacena en la base de datos y sus relaciones. En esta capa no se especifica cómo se almacenan en el ordenador. Es independiente de los usuarios y de las aplicaciones y no se considera la eficiencia del ordenador.
- **La capa externa**: Esta capa constituye una colección de vistas de usuarios. Una vista de usuario muestra la información relevante para un usuario en concreto.



www.grupostudium.com
informacion@grupostudium.com
954 539 952

1.6 Modelado de una base de datos

Al enfrentarnos al **diseño de una base de datos** siempre partimos de una **descripción** más o menos exhaustiva. En caso de que no exista, debemos mantener tantas **reuniones** con el cliente como sean necesarias para que todo quede perfectamente **especificado, delimitado y claro**. Este es el caso ideal. En la realidad, siempre habrá modificaciones, actualizaciones y correcciones sobre la marcha que aparecerán a la hora de diseñar, programar o implementar las bases de datos.

Una vez tengamos los requisitos, pasamos al análisis de éstos, de forma que repetiremos la siguiente secuencia de pasos tantas veces como sea necesario para cumplir con todos los requisitos:

1 – Identificar las **entidades**. Las entidades son los elementos del universo en el que nos movemos de las que queremos guardar información relevante. Pueden ser los *Clientes*, los *Empleados*, las *Facturas*, los *Vehículos*, los *Tipos de Vehículos*, etc.

2 – Para cada entidad, establecer las **propiedades** o **atributos** que queremos guardar de cada una de ellas. Por ejemplo, para cada entidad Clientes, nos puede interesar guardar su Nombre, sus Apellidos, su DNI, etc. Depende de lo que estemos modelando, para una misma entidad pueden ser necesarios diferentes atributos. Existe un atributo que no puede faltar en ninguna entidad. Es el llamado campo clave, clave primaria o clave principal. De todos los campos detectados para una entidad, dicho campo es el que nos permitirá diferenciar a una entidad Clientes de otra entidad Clientes. En este caso podría ser el DNI, que es único para cada cliente y por tanto nos sirve para diferenciarlos entre sí.

Este campo clave NUNCA podrá ser **NULO**, ni se podrá repetir. Existe en la mayoría de los sistemas gestores de bases de datos, un tipo de atributo llamado **Autonumérico** que de forma automática da valores diferentes siempre.

3 – Establecer las **relaciones** entre las entidades antes descritas. Como ya se ha comentado anteriormente, una de las ventajas de usar bases de datos relacionales es la de evitar la redundancia. Esto se consigue de la siguiente forma: Imaginemos que queremos guardar los datos de nuestras facturas. En cada nueva factura aparece su número y su fecha, pero también aparecen los datos de los productos que se venden y los datos del cliente que compra. Si tuviéramos que poner todos los datos de cada cliente cada vez que le hacemos una factura, su nombre, DNI, etc. se meterían en nuestro sistema muchas veces. Si relacionamos de alguna manera sencilla las facturas con los clientes, evitaríamos la redundancia. De esta forma, al crear un registro de la entidad *Facturas*, dentro de ella tendremos un campo llamado *idClienteFK* que identificará el campo clave de la entidad *Clientes*. Y así se crea la **relación** entre estas dos entidades.

Entidad **Facturas** SIN relaciones:

idFactura	fechaFactura	nombreCliente	apellidosCliente	dniCliente	telefonoCliente
1	12/10/2011	Juan	Pérez Pérez	12345678Z	654654654
2	12/10/2011	Pedro	Mármol Ruiz	87654321X	654778899
3	12/10/2011	Juan	Pérez Pérez	12345678Z	654654654
4	13/10/2011	Pablo	Márquez Luz	12233445V	652223344
5	13/10/2011	José	López Ruiz	54435465G	687878787
6	14/10/2011	Juan	Pérez Pérez	12345678Z	654654654

Como podemos ver, los datos del cliente Juan Pérez Pérez se repiten en cada factura. Si esto lo extrapolamos a muchas facturas de este cliente y muchos clientes como él, con muchas facturas, la cantidad de información redundante es excesiva.

Para evitarlo, sacamos dos entidades de la anterior, una para los Clientes y otra para las Facturas:

Entidad **Clientes** SIN repeticiones:

dniCliente	nombreCliente	apellidosCliente	telefonoCliente
12345678Z	Juan	Pérez Pérez	654654654
87654321X	Pedro	Mármol Ruiz	654778899
12233445V	Pablo	Márquez Luz	652223344
54435465G	José	López Ruiz	687878787

Entidad **Facturas** SIN repeticiones:

idFactura	fechaFactura	dniClienteFK
1	12/10/2011	12345678Z
2	12/10/2011	87654321X
3	12/10/2011	12345678Z
4	13/10/2011	12233445V
5	13/10/2011	54435465G
6	14/10/2011	12345678Z

Para este caso, la relación se ha establecido entre el atributo *dniCliente* de **Clientes** y *dniClienteFK* de Facturas. Como vemos es el mismo atributo, pero a este último se le ha añadido en su nombre las letras FK. Estas siglas vienen de **Foreign Key** o **Clave Foránea** o **Clave Ajena**, que indica el campo por el que se relacionan las entidades teniendo en cuenta que dicha relación será SIEMPRE entre una **entidad** y el **atributo clave** o **clave primaria** de la otra entidad.

22/08/2022

www.grupostudium.com
informacion@grupostudium.com
954 539 952