Fundamentos de Bases de Datos

grado en ingeniería informática

Resumen del temario

Autor

Carlos Sánchez Páez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2017-2018

Índice

Índice de figuras

Índice de definiciones

1. Tema 1. Introducción y definiciones iniciales.

1.1. Concepto intuitivo de bases de datos

Prácticamente todas las empresas necesitan aplicaciones que gestionen información a la que se accederá desde distintos puntos. Si estos datos pertenecen a las aplicaciones, hay tres problemas principales:

- **Redundancia**. La información se repite en varios sitios a la vez.
- **Inconsistencia**. ¿Cuáles son los datos más actualizados?
- **No hay reutilización**.

Si utilizamos ficheros, podemos hacer que la información sea compartida, sin embargo:

- Tenemos que mantener una estructura determinada.
- Debemos proteger los archivos de ciertos usuarios.
- Debemos permitir el acceso con distintos lenguajes y sistemas operativos.

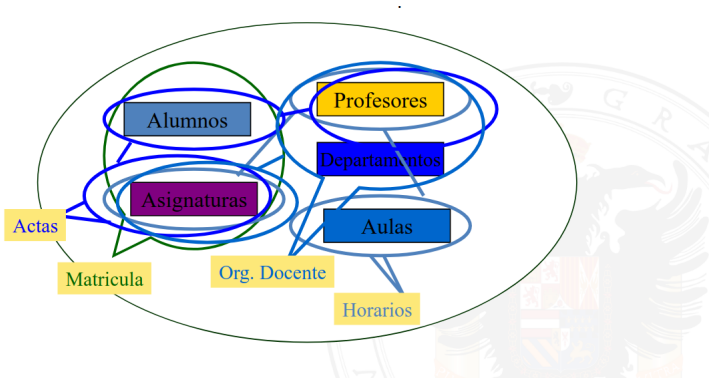

Figura 1: Acceso a archivos desde diferentes aplicaciones.

Por tanto, la solución final es **una base de datos**.

Definición 1.1.1 (Base de datos)**.** *Una* base de datos *es un conjunto de datos comunes a un proyecto almacenados sin redundancia para ser útiles a distintas aplicaciones.*

Definición 1.1.2 (Sistema gestor de bases de datos)**.** *Un* sistema gestor de bases de datos (SGBD) *es un conjunto de elementos software con la capacidad para definir, mantener y utilizar una base de datos.*

Un SGBD debe permitir:

- **Definir** estructuras de almacenamiento.
- **Acceder** a los datos de forma eficiente y segura.
- Organizar la **actualización** de los datos y el **acceso** multiusuario.
- etc.

Resumiendo, una base de datos es un fondo común de información almacenada en una computadora para que cualquier persona o programa autorizado pueda acceder a ella, independientemente del lugar de procedencia y el uso que haga de la misma.

Con un SGBD podemos gestionar datos y una estructura de datos de forma transparente (sin tener que programar un código específico):

- **Insertar** datos.
- **Modificar** datos existentes.
- **Borrar** datos existentes.
- **Obtener** datos previamente insertados.

Normalmente estas operaciones se denominan CRUD (**C**reate, **R**ead, **U**pdate y **D**elete).

1.2. Bases de datos y sistemas de gestión de bases de datos.

Una base de datos involucra:

- **Datos**
  - Integrados (sin redundancia).
  - Compartidos (útiles a varias aplicaciones).
- **Hardware**
  - Base de datos normal.
  - Base de datos distribuida.
- Software **DBMS** (DataBase Management System). Son programas par describir las estructuras y gestionar la información de la base de datos.
- **Usuarios**
  - Usuario final.
  - Programador de aplicaciones.
  - Administrador (DBA ó DBM).

Definición 1.2.1 (Dato operativo)**.** *Un* ***dato operativo*** *es una pieza de información básica que necesita una empresa, proyecto o aplicación para su funcionamiento.*

Un dato operativo puede ser:

- **Ítem básico**. Elementos acerca de los que se puede pedir información (sustantivos).
- **Atributos**. Características de los ítems básicos (adjetivos o propiedades de los ítems).
- **Relaciones**. Conexiones lógicas entre ítems.

Cuando se determinan y se clasifican así todos los datos operativos obtenemos el *esquema lógico* de la base de datos.

En el día a día, usaremos el término **campo** para referirnos a la representación de un dato o atributo en la base de datos.
Ejemplos de dato operativo:

- **Ítem básico**. Estudiante, asignatura, paciente, etc.
- **Atributos**. Nombre, apellidos, relación, etc.
- **Relaciones**. Estudiante **está matriculado en** asignatura, profesor **imparte** asignatura, etc.

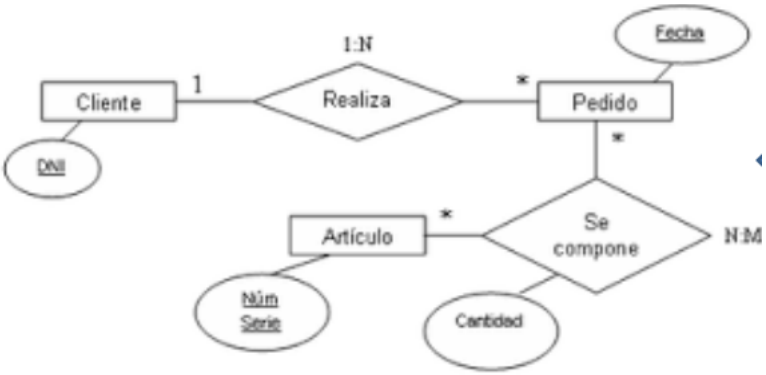

Figura 2: Ejemplo de diagrama entidad-relación.

1.3. Concepto de independencia

Definición 1.3.1 (Independencia)**.** *Los datos se organizan independientemente de las aplicaciones que los vayan a usar de los archivos en los que vayan a almacenarse.*

Definición 1.3.2 (Independencia física)**.** *El diseño lógico de la base de datos debe ser independiente del almacenamiento físico de los datos.*

Ésto permite:

- Realizar cambios en la estructura física sin alterar la lógica de la aplicación (representación de campos, organización en registros, mecanismos de acceso, etc.)
- Liberar a las aplicaciones de la misión de gestionara aspectos relativos al almacenamiento.

Definición 1.3.3 (Independencia lógica)**.** *Cada aplicación debe poder organizar los datos según sus propios esquemas y acceder a los datos que le son necesarios y le conciernen (vistas de usuario).*

La independencia lógica provoca varias mejoras:

- Aumento de seguridad y fiabilidad.

- Menos problemas para las aplicaciones.
- Posibilidad de cambios en los esquemas por parte de desarrolladores de aplicaciones y administradores.

El **esquema lógico general** permite organizar la información global de toda la organización para optimizar accesos, evitar redundancia, etc.

La **vista de usuario** permite dar permiso a los programadores de las aplicaciones para acceder a los datos que pueden ver del esquema general, ocultando los datos a los que no se debe tener acceso.

1.4. Objetivos de un SGBD

Los objetivos de un sistema de gestión de bases de datos son:

- **Independencia de los datos**.
- **Utilización y diseño orientados al usuario**. Los datos y aplicaciones deben ser accesibles a los usuarios de la forma más amigable posible.
- **Centralización**. Los datos deben gestionarse de forma centralizada e independiente de las aplicaciones.
- **Eliminación de redundancia**. Los datos no pueden estar duplicados y se deben gestionar los accesos concurrentes.
- **Consistencia**. Los datos deben ser consistentes (sin fallos lógicos) y se deben implementar mecanismos para mantener la integridad.
- **Fiabilidad**. Los datos deben estar protegidos contra fallos, para lo que son necesarios mecanismos de mantenimiento, recuperación y realizamiento de transacciones.
- **Seguridad**. No todos los datos deben ser accesibles a todos los usuarios.

Hay varios tipos de usuario en una base de datos:

- **Usuario final**. Debe poder acceder a los datos.
- **Programador de aplicaciones**. Debe eliminar problemas de diseño, depuración y mantenimiento.
- **Administrador**. Su cometido surge con la aparición de la base de datos.

En cuanto al sistema:

- Control **centralizado**.
- Criterios de **uniformización**.
- Generación de **nuevas aplicaciones**.
- **Equilibrio** entre requerimientos conflictivos.

2. Tema 2. Arquitectura de un sistema gestor de bases de datos.

2.1. Una arquitectura con tres niveles

Un SGBD se organiza en niveles por varios motivos:

- Los usuarios pueden acceder a los mismos datos desde distintas perspectivas, por lo que si un usuario cambia la forma de ver los datos no influye al resto.
- La organización global de los datos puede cambiarse sin que afecte a los usuarios.
- Los usuarios no tienen por qué gestionar aspectos relativos a la representación física de los datos, por lo que el administrador de la base de datos puede cambiar la representación sin que ésto influya en los usuarios.

La división se realiza en tres niveles:

Definición 2.1.1 (Nivel interno)**.** *Constituye la representación de la base de datos más cercana a la estructura de almacenamiento físico. Por tanto, es la capa donde se establece la forma en que se implantan las estructuras de datos que organizan los niveles superiores.*

Definición 2.1.2 (Nivel conceptual)**.** *Supone una abstracción global de la base de datos que agrupa todas las percepciones que los usuarios tienen de ella.*

Definición 2.1.3 (Nivel externo)**.** *En este nivel se definen todas las percepciones particulares de la base de datos por parte de los usuarios. Cada usuario puede tener su propia visión de la BD.*

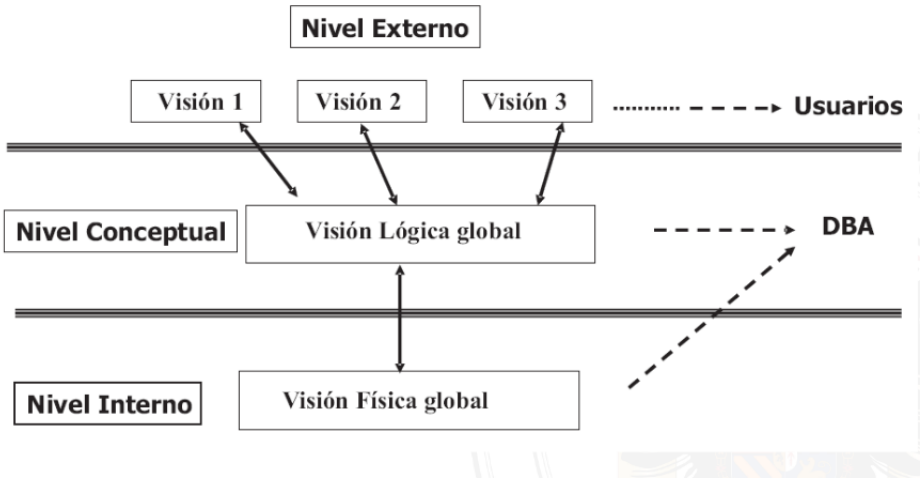

Figura 3: Organización en tres niveles.

2.1.1. Nivel externo

Es la parte de la base de datos que es relevante para el usuario, es decir, aquellas entidades, relaciones o atributos que le son de interés.
Estas entidades, relaciones y atributos son representadas de la forma que le interesa al usuario (nombre completo o nombre y apellidos, etc.) y se ofrecen datos calculados a partir de ellas (edad a partir de fecha de nacimiento, etc.)

2.1.2. Nivel conceptual

Ofrece una visión global de los datos, así como de su estructura lógica (los datos que están almacenados y las relaciones entre ellos).
Éste nivel representa:

- Todas las entidades, atributos y relaciones.
- Las restricciones que afectan a los datos.
- Información semántica sobre los datos.
- Información de seguridad e integridad.

También ofrece soporte a cada vista externa. No debe contener ningún detalle de almacenamiento.

2.1.3. Nivel interno

Es la representación física de la base de datos en el ordenador, es decir, la forma en la que se almacenan los datos. Busca obtener el máximo rendimiento del sistema. Representa:

- Estructuras de datos.
- Organizaciones en ficheros.
- Comunicación con el sistema operativo para gestionar el uso de las unidades de almacenamiento.
- Compresión de datos, cifrado, etc.

Parte de las responsabilidades de este nivel son realizadas por el sistema operativo (el *nivel físico*). Sin embargo, como depende de cada SGBD y de cada SO, no hay una división clara.

```
Profesor = registro de
    NRP          campo alfanumérico de 10 caracteres,
    Apellidos    campo alfanumérico de 30 caracteres,
    Nombre       campo alfanumérico de 20 caracteres,
    Sueldo       campo decimal de 8+2 dígitos,
    Departamento campo alfanumérico de 30 caracteres
fin Profesor.
```

(a) Visión conceptual.

– Gestión de personal
– Lenguaje A

```
TYPE Profesor IS RECORD (
  NRP VARCHAR2(10),
  Apellidos VARCHAR2(30),
  Nombre VARCHAR2(20),
  Sueldo NUMBER(8,2)
);
```

(b) Visión externa 1.

– Ordenación académica
– Lenguaje B

```
TYPE Profesor = RECORD
  NRP : STRING[10];
  Apellidos : STRING[30];
  Nombre : STRING[20];
  Departamento : STRING[30];
END;
```

(c) Visión externa 2.

```
Profesor_interno BYTES=74

    NRP TYPE=BYTES(10),OFFSET=0
    Apellidos TYPE=BYTES(30),OFFSET=10
    Nombre TYPE=BYTES(20),OFFSET=40
    Sueldo TYPE=WORD(2),OFFSET=60
    Departamento TYPE=BYTES(10),OFFSET=64.
```

(d) Visión interna.

Figura 4: Ejemplo de los tres niveles de visión.

2.2. Correspondencias entre niveles

Definición 2.2.1 (Transformación o correspondencia entre niveles)**.** *Es un conjunto de normas que establece cómo se definen los datos de un nivel en términos de otro.*

Es un mecanismo fundamental para establecer la independencia tanto física como lógica.

Definición 2.2.2 (Transformación conceptual/interna)**.** *Define cómo se organizan las entidades lógicas del nivel conceptual en términos de registros y campos almacenados en el nivel interno.*

Definición 2.2.3 (Transformación externa/conceptual)**.** *Describe un esquema externo en términos del esquema conceptual subyacente.*

Algunos SGBDs permiten la descripción de esquemas externos en términos de otros esquemas externos (transformación externa/externa).

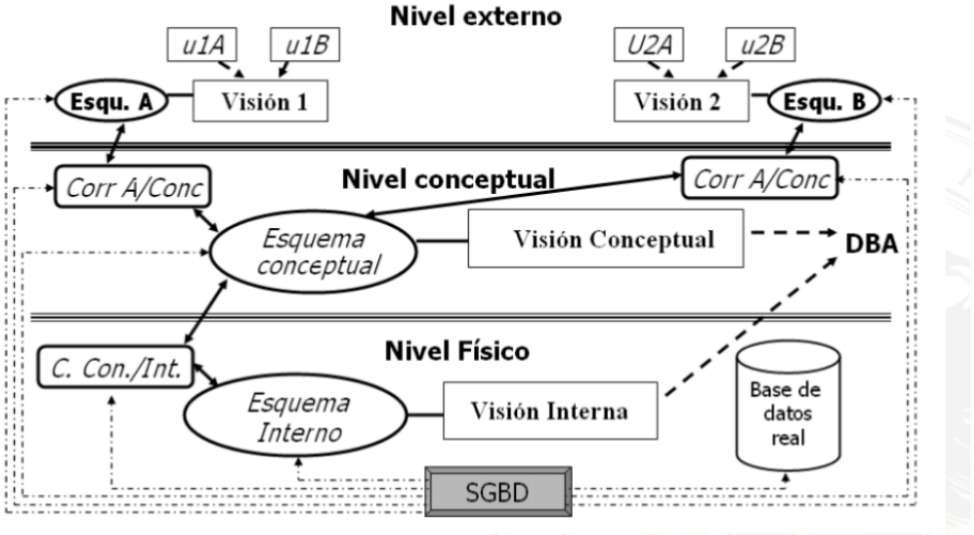

Figura 5: Correspondencia entre niveles.

2.3. Lenguajes de una base de datos

ANSI/SPARC recomienda disponer de un lenguaje específico orientado a los datos que permita definirlos, controlarlos y manipularlos.

Definición 2.3.1 (DSL)**.** *Sublenguaje de datos implementado en el propio SGBD. Consta de tres partes: DDL, DCL y DML.*

Definición 2.3.2 (DDL)**.** *(Data Definition Languaje) o sublenguaje de definición de datos. Es el subconjunto del DSL destinado a la definición de estructuras de datos y esquemas en la base de datos.*

Definición 2.3.3 (DML)**.** *(Data Manipulation Languaje) o sublenguaje de manipulación de datos. Es el subconjunto del DSL que permite introducir datos en los esquemas, modificarlos, eliminarlos y consultarlos. También permite consultar la estructura de los esquemas definidos en la base de datos.*

Definición 2.3.4 (DCL)**.** *(Data Control Languaje) o sublenguaje de control de datos. Permite gestionar los requisitos de acceso a los datos y otras tareas administrativas de la base de datos.*

ANSI/SPARC propone disponer de un *DDL*, un *DML* y un *DCL* para cada nivel de la arquitectura.

En la práctica, estos lenguajes se presentan bajo una **única implementación** en la que cada sentencia trabajará sobre uno o varios niveles. Un sistema de privilegios será el

encargado de discriminar quién puede ejecutar qué y en qué nivel.

Como la industria ha seguido diferentes caminos, han surgido diferentes lenguajes de datos diferentes a la par que intentos por estandarizarlos (como SQL y sus estandarizaciones SQL89,SQL92, etc.).

Los SGBD deben tener un lenguaje anfitrión en el que están programados. Pueden ser de dos tipos:

- De **propósito general**: C, C++, Java, C#, etc.
- Herramientas de desarrollo **específicas**: Developer de Oracle, Oracle Application Express (Oracle APEX), Sysbase PowerBuilder, etc.

Estos lenguajes proporcionan un procesamiento avanzado de datos y la gestión de la interfaz de usuario.

Es necesario establecer un mecanismo para trasladar las estructuras de datos y las operaciones desde la base de datos al entorno de procesamiento de la aplicación. Surgen así dos tipos de acoplamiento:

Débil Se utilizan lenguajes de propósito general. En ellos el programador puede distinguir sentencias propias del lenguaje anfitrión y sentencias que acceden a la BD a partir de DSL.

Las principales alternativas para implementarlos son las APIs de acceso a BD (ODBC, JDCB) y el DSL inmerso en el código fuente del lenguaje anfitrión (el programador programa un código híbrido entre lenguaje anfitrión y sentencias DLS y este código pasa por un preprocesador).

Fuerte Se utilizan las herramientas de propósito especifico. Consiste en incorporar características al DSL para facilitar el desarrollo de aplicaciones.

Las principales alternativas para su implementación son PL de Oracle o la ejecución de Java sobre una máquina virtual implantada en el SGBD.

2.4. Enfoques para la arquitectura de un SGBD

Inicialmente, el esquema de la base de datos era centralizado:

- Toda la carga de gestión y procesamiento de la información recaía en servidores centrales.
- El acceso se realizaba mediante terminales.
- En el ordenador principal se alojaban el SGBD y los programas de aplicación.

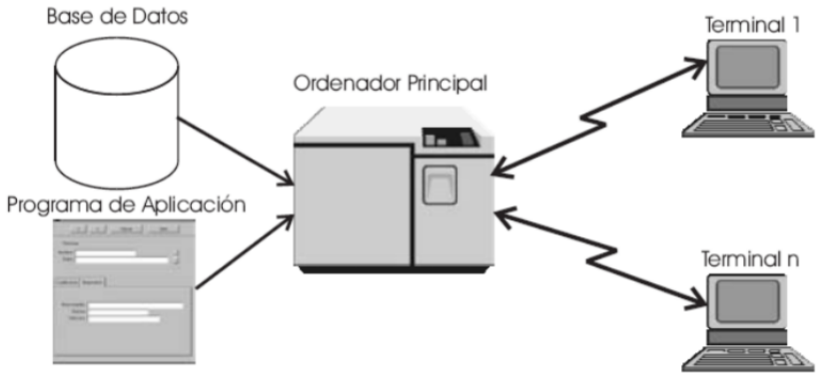

Figura 6: Arquitectura centralizada.

Esta arquitectura tenía un problema principal: el elevado coste de los ordenadores principales.
La solución propuesta fue desplazar la ejecución de los programas de usuario y la interacción hasta los emergentes PCs, reduciendo los costes en hardware. De esta forma, se conseguía una aproximación al modelo cliente/servidor.

- **Servidor**. Es el servidor de la base de datos y de escucha de peticiones.
- **PCs conectados en red con el servidor**. Programas de aplicación y el servicio de enlace cliente que interactúa con el servicio de escucha instalado en el servidor.

El desarrollo de las redes de comunicaciones conllevó un enfoque distribuido para los servidores.

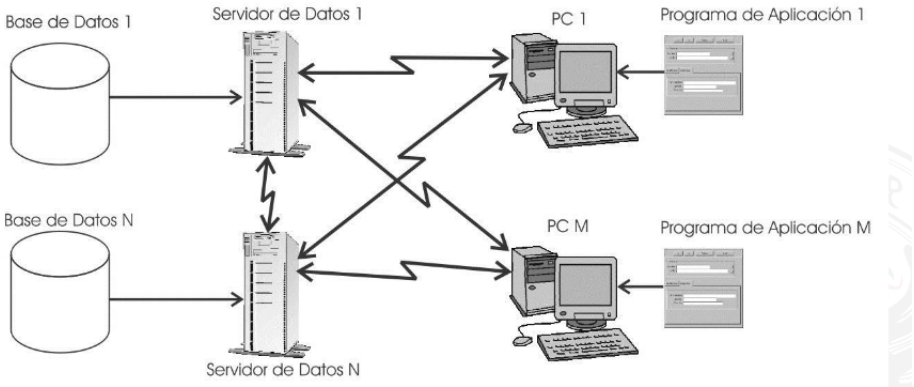

Figura 7: Arquitectura distribuida.

Sin embargo, esta arquitectura también ofrecía un gran problema: el alto coste de mantenimiento de los PCs (instalación, configuración y actualización). La solución fue separar el sistema en aplicaciones: una parte que interactúa con el usuario (*interfaz de usuario*) y otra parte que se encarga de la ejecución lógica del programa.

Actualmente las bases de datos se estructuran en tres niveles de procesamiento:

- **Nivel de servidor de datos**.
  - Posiblemente distribuido.
  - El SGBD permite organizar la información de la empresa como una BD global.
  - Las peticiones de datos formuladas desde una sede se traducen de forma transparente a peticiones en las sedes donde se encuentran esos datos.
- **Nivel de servidor de aplicaciones**. Son evoluciones de servidores web que proporcionan los programas de aplicación a clientes ligeros.
- **Nivel de cliente**. PCs ligeros dotados de configuraciones basadas en estándares abiertos. En muchos casos se pueden ejecutar las aplicaciones en un navegador web que soporte *javascript* y *html* avanzado.

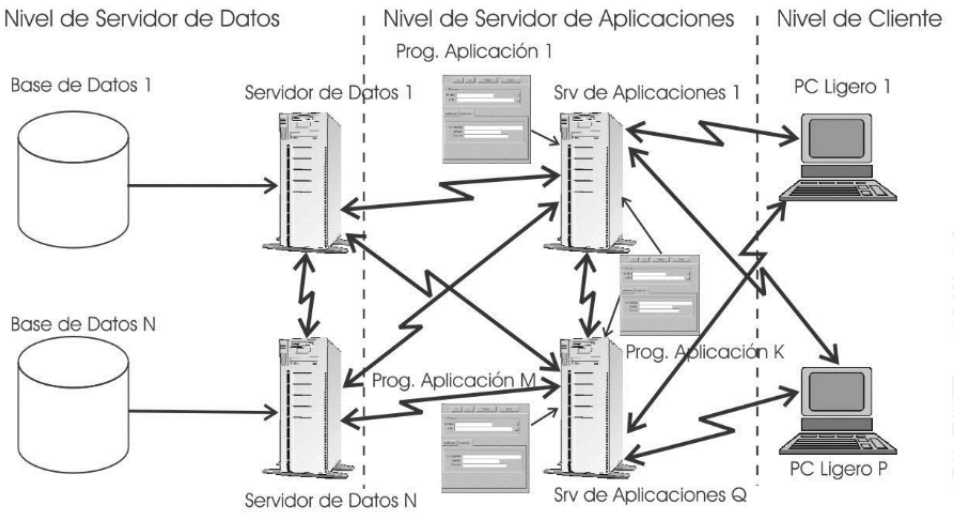

Figura 8: Arquitectura de tres niveles.

Ventajas de la arquitectura de tres niveles

- Reducción de costes en cuanto al mantenimiento de los clientes, ya que se realiza en el servidor, no en cada cliente.
- Mayor facilidad y flexibilidad para el usuario, ya que puede acceder casi desde cualquier puesto o dispositivo.

Su principales inconvenientes son que aumenta la complejidad a la hora de configurar y administrar servidores de aplicaciones y que es necesario desarrollar aplicaciones adaptadas a este modelo distribuido.

2.5. El administrador de la base de datos (DBA)

El DBA es una figura muy importancia en el contexto de los SGBD, ya que:

- Elabora el esquema conceptual, analizando las necesidades de información de la empresa, identificando los datos operativos, elaborando el esquema lógico e implantando el esquema conceptual.
- Decide la estructura de almacenamiento en el nivel interno.
- Conecta con los usuarios mediante el análisis de requisitos, el diseño lógico, etc.
- Define las restricciones de integridad, como las reglas.
- Define e implanta la política de seguridad (gestión de usuarios y privilegios).
- Define e implanta la estrategia de recuperación frente a fallos (SGBDs redundantes o RAID).
- Optimiza el rendimiento de la BD liberando espacio no utilizado, reorganizando las operaciones para que se realicen de forma más rápida, estableciendo prioridades. etc.
- Monitoriza el SGBD.

3. Tema 3. Modelos de datos.

Definición 3.0.1 (Modelo de datos)**.** *Un modelo de datos es un mecanismo formal para representar y manipular información de manera general y sistemática. Debe constar de una notación para describir datos, otra para describir operaciones y otra para describir reglas de integridad.*

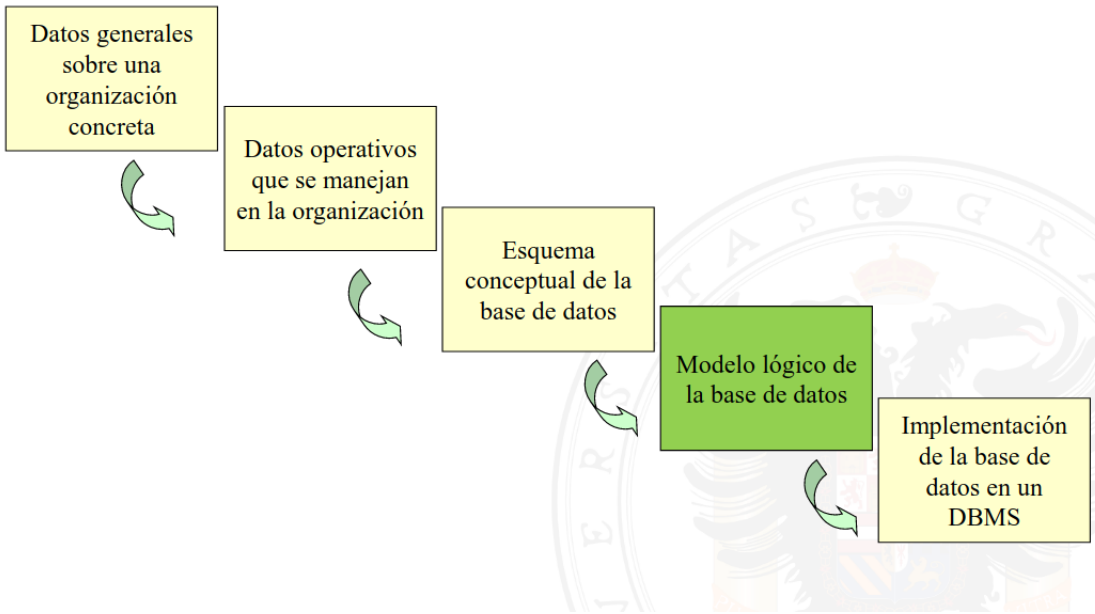

Figura 9: Pasos a seguir para crear una base de datos.

Una vez llevado a cabo el proceso de análisis de datos y obtenido el esquema conceptual o *lógico* de nuestra base de datos, necesitamos implantarla en un sistema a partir de un proceso de **diseño** que nos permitirá trasladar la estructura actual a un modelo de datos implementable.

El **modelado lógico** consiste en trasladar a un esquema lógico en función de una estructura implementable. Por último implementamos esta estructura en un sistema comercial.

Los modelos de datos son necesarios por varias razones:

- Cada esquema se describe utilizando un lenguaje de definición de datos.
- Este lenguaje es de muy bajo nivel, muy ligado al SGBD.
- Hacen falta otros mecanismos de más alto nivel para poder describir los datos de una forma no ambigua y entendible por los usuarios implicados en cada paso del proceso de implantación.

El objetivo de un modelo de datos es **describir** modelos que representen los datos y que los describan de una forma entendible y manipulable. En relación a la arquitectura ANSI/SPARC existen tres modelos: externo, conceptual e interno.

Los modelos de datos pueden ser:

- Basados en registros.
  - Modelo de datos **jerárquico**.
  - Modelo de datos **en red**.
  - Modelo **relacional**.
- Basados en objetos.
- Físicos.

Los dos primeros se utilizan para los niveles externo y conceptual y el físico para el nivel interno.

3.1. Modelo jerárquico

Fue el primero en implementarse físicamente. En el nivel externo se utilizaban aplicaciones en *Cobol* y no había interactividad (no había lenguaje de consulta).

Se basa en una estructura de datos básica: un árbol con registros maestro y secundarios. La base de datos es una colección de instancias de árboles.

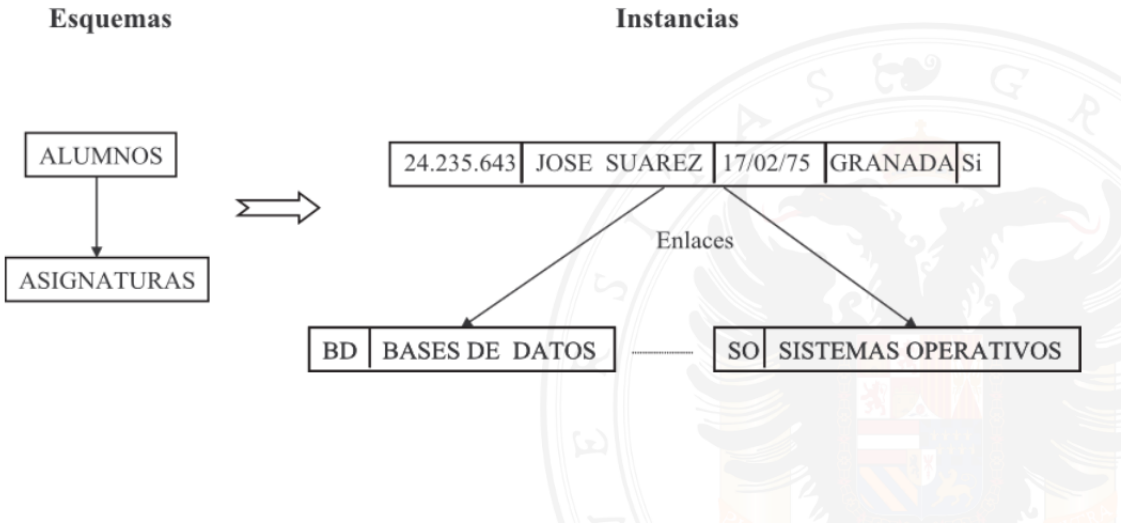

Figura 10: Modelo jerárquico.

Plasma de forma muy directa las relaciones muchos a uno y uno a uno. Sin embargo, tenemos que duplicar la información de las entidades involucradas en relaciones muchos a muchos.

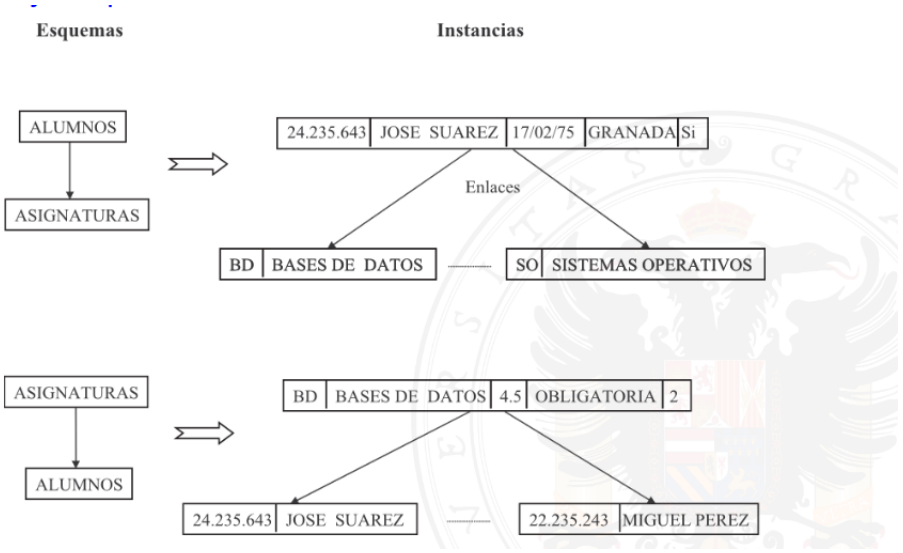

Figura 11: Relaciones muchos a muchos en el modelo jerárquico.

Sus principales inconvenientes son:

- Almacenar árboles es complejo (mantenimiento de punteros y variedad de tipos de registros).
- El DML es difícil de usar e implementar.
- Dependencia existencial obligatoria de los registros secundarios con respecto a los de tipo raíz (no se puede insertar un registro secundario sin uno raíz donde *engancharlo*).
- Redundancia al plasmar relaciones muchos a muchos.

3.2. Modelo en red

Su estructura de datos son grafos cuya topología depende de las conexiones entre las entidades.

- Nodos → registros.
- Arcos → enlaces entre registros (punteros).
- Relaciones entre conjuntos de entidades → conectores (registros especiales). Cada ocurrencia de un conector representa una asociación distinta.
- Cualquier registro puede relacionarse con cualquier registro.

La base de datos se estructura como una colección de instancias de grafos.

La estructura es muy genérica: permite plasmar todo tipo de relaciones e implementa directamente las relaciones muchos a muchos.

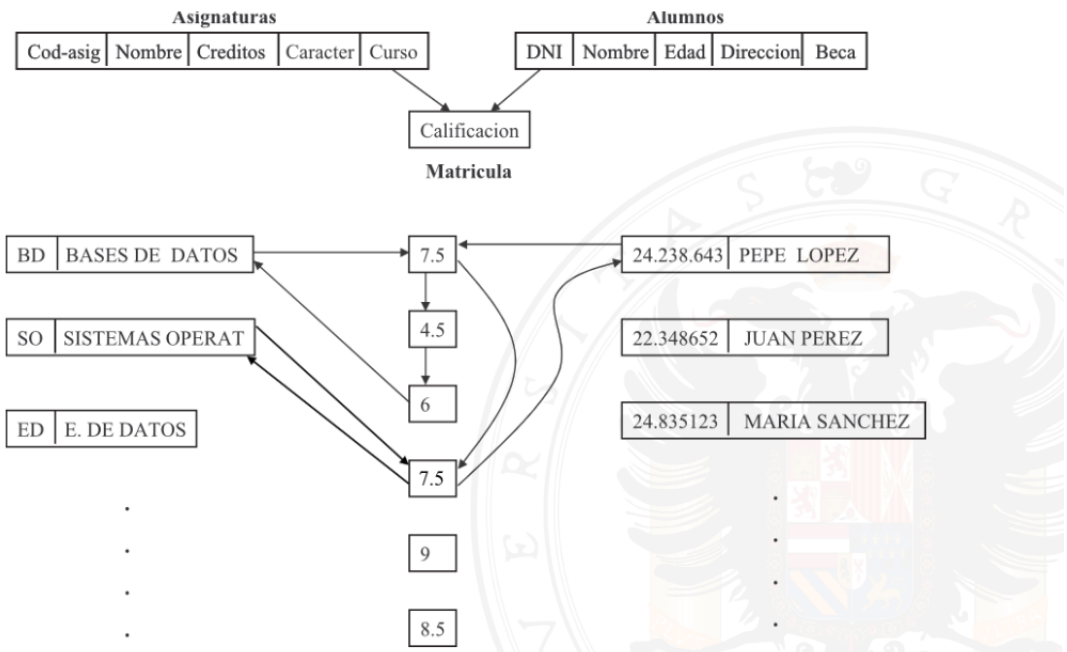

Figura 12: Modelo en red.

Sus principales ventajas son que la estructura es más homogénea y que permite insertar nuevas entidades de forma independiente.
Sin embargo, la existencia de enlaces entre registros hace que las operaciones del DDL y el DML sean complejas de implementar y utilizar.

3.3. Modelo relacional

Organiza y representa los datos en forma de tablas o relaciones. Una base de datos relacional es una colección de tablas cada una de las cuales tiene un nombre único.

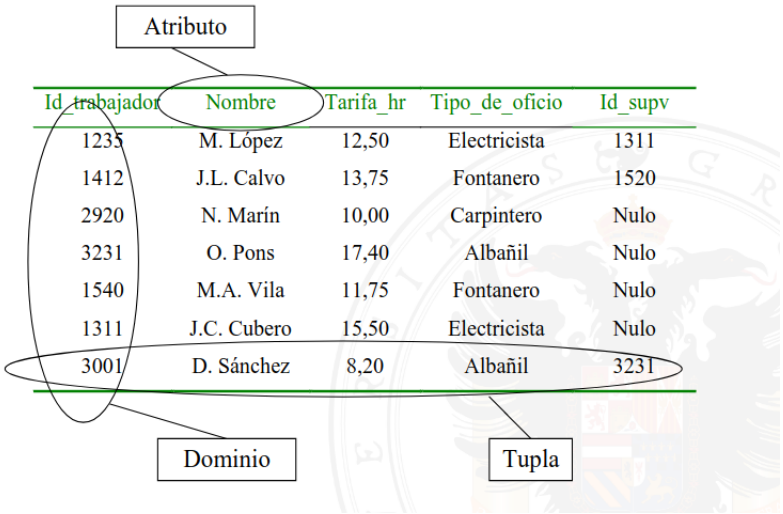

Figura 13: Modelo relacional.

Definición 3.3.1 (Esquema de una base de datos relacional)**.** *Colección de esquemas de relaciones junto con restricciones de integridad.*

Definición 3.3.2 (Instancia o estado de una base de datos)**.** *Colección de instancias de relaciones que verifican las restricciones de integridad.*

Definición 3.3.3 (Base de datos relacional)**.** *Instancia de una base de datos junto con su esquema.*

Claves

Definición 3.3.4 (Superclave)**.** *Cualquier conjunto de atributos que identifica unívocamente cada* tupla *de una relación.*

Definición 3.3.5 (Clave candidata)**.** *Es una superclave minimal.*

De entre las candidatas tenemos que elegir una como principal, denominada **clave primaria**.

	Relacional	**Basado en grafos**
Representación	Un sólo elemento para la representación Conexiones lógicas Representación de relaciones n:m simétrica. Identidad por valor	Dos elementos para la representación Conexiones en el modelo físico subyacente. Representación de relaciones n:m imposible en modelos jerárquicos, difícil en modelos de red. Identidad por posición
Consulta	Consultas simétricas en jerarquías. Obtención de la consulta como resultado global Lenguajes declarativos	Consultas no simétricas en jerarquías. Mecanismo de navegación por punteros. Lenguajes procedimentales

Figura 14: Comparativa de modelos.

4. Tema 4. El modelo relacional.

4.1. La estructura de datos relacional

El modelo relacional abarca tres ámbitos distintos de los datos:

- **Las estructuras para almacenarlos**. El usuario percibe la información de la base de datos estructurada en tablas.
- **La integridad**. Las tablas deben satisfacer ciertas condiciones que preservan la integridad y coherencia de la información que contienen.
- **Consulta y manipulación**. Los operadores empleados por el modelo se aplican sobre tablas y devuelven tablas.

La tabla es la estructura lógica de un sistema relacional. A nivel físico, el sistema es libre de almacenar los datos en el formato más adecuado (archivo secuencial, indexado, etc.).

4.2. Definiciones iniciales

Definición 4.2.1 (Atributo)**.** *Cualquier elemento de información susceptible de tomar valores. Notación:* A_i

Definición 4.2.2 (Dominio)**.** *Rango de valores donde toma sus datos un atributo. Se considera finito. Notación:* D_i

Definición 4.2.3 (Relación)**.** *Dados los atributos* $A_i \forall i \in \mathbb{N}$ *con dominios* $D_i \forall i \in \mathbb{N}$*, no necesariamente distintos, definimos relación asociada a* $A_1, ..., A_n$ *y lo notaremos por* $R(A_1, ..., A_n)$ *a cualquier subconjunto de* $D_1 \times D_2 \times, ..., \times D_n$*.*

Definición 4.2.4 (Tupla)**.** *Cada una de las filas de una relación.*

Definición 4.2.5 (Cardinalidad de una relación)**.** *Número de tuplas que contiene. Es variable en el tiempo.*

Definición 4.2.6 (Esquema de una relación R)**.** *Atributos de la relación junto con su dominio,* $A_1 : D_1, ..., A_n : D_n$*.*

Definición 4.2.7 (Grado de una relación)**.** *Número de atributos de su esquema. Es invariable en el tiempo.*

Definición 4.2.8 (Instancia de una relación)**.** *Conjunto de tuplas* $(x_1, x_2, ..., x_n) \subseteq D_1 \times D_2 \times, ..., \times D_n$ *que componen una relación en cada momento.*

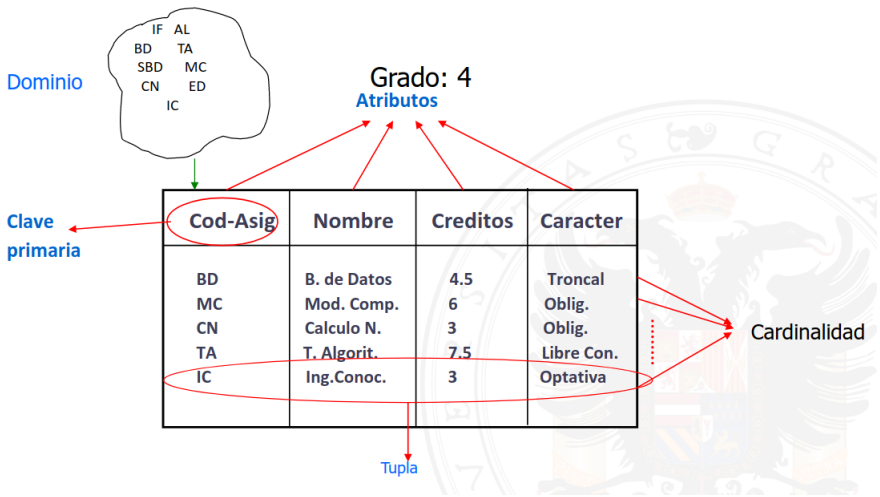

Figura 15: Ejemplo de elementos del modelo relacional.

4.3. Propiedades de la estructura de datos relacional

Definición 4.3.1 (Condición de normalización)**.** *Todos los valores de los atributos de una relación son no atómicos (no estructurados). Cuando una relación cumple esta condición, se dice que está en* primera forma normal.

La condición de normalización trae varias consecuencias:

- No hay valores de tipo conjunto, registro ni tablas.
- No hay tuplas duplicadas.
- No hay orden en las filas ni en los atributos (se accede por nombre de atributo y valor).
- Varias instancias representan la misma relación.

Sin embargo, trae un problema: todas las representaciones son extensivas, por lo que no se puede representar información del tipo .[el] valor de un atributo en una tupla es igual al de otro atributo en otra tupla".

4.4. Notación

- **Relación**. R,S,T,...
- **Atributos**. A,B,...
- **Esquema de relación**. $R[A_1, A_2, ..., A_n]$.
- **Instancia de relación R**. r.
- **Tuplas de una instancia**. $x_1, x_2, ..., x_n \in r$.
- **Valor de un atributo A_i en una tupla x_j**. $x_j[A_j]$ ó A_{ij}.
- Si no conocemos el valor de un atributo en una tupla, se le asigna un valor **nulo**.

4.5. Restricciones o reglas de integridad

Hay varios tipos:

- **Asociada a tablas**. Ejemplos: $0 \leq edad \leq 100$, $creditos > 0$,...
- **Asociada a bases de datos**. Ejemplos: $imparte.NRP \in profesor.NRP$, $cod_asig \neq nulo$.

Cuando hay más de una clave candidata en una relación, tenemos que elegir una, que será denominada **clave primaria**.

Definición 4.5.1 (Integridad de entidad)**.** *No se debe permitir que una entidad sea definida en la base de datos si no tiene una información completa de sus atributos clave, es decir, la clave primaria (o una parte de ella) no puede ser nula.*

Definición 4.5.2 (Clave externa)**.** *Conjunto de atributos de una relación que es clave en otra (o incluso en la misma) relación. Es decir, es un conjunto de atributos de una relación cuyos valores coinciden con valores de la clave primaria de las tuplas de otra relación.*

Definición 4.5.3 (Integridad referencial)**.** *Todos los valores no nulos de una clave externa deben referenciar valores reales de la clave referenciada en la otra relación.*

El SGBD debe encargarse de mantener las siguientes restricciones:

1. **Unicidad de clave primaria y claves candidatas**. Ante inserciones y actualizaciones, el SGBD debe rechazar los valores que ya existan en la base de datos en las claves primarias y candidatas.
2. **Restricción de integridad de identidad**. Ante inserciones y actualizaciones, se deben rechazar las modificaciones que vulneren la unicidad de la clave primaria o que le asignen algún valor nulo.
3. **Integridad referencial**
   a) En la *inserción* se debe rechazar la tupla insertada si el valor de su clave externa no concuerda en la relación referenciada o si el valor es nulo y el diseño no lo permite.
   b) En la *actualización*, además de verificar todo lo anterior, se debe actualizar la clave primaria de la relación referenciada.
   c) Si se *borra* la clave primaria en la relación referenciada se deben borrar en cadena todas las tuplas que la referencian o poner su clave externa a valor nulo.