TEMA 1: INTRODUCCIÓN Y DEFINICIONES INICIALES

Índice:

1.1 Concepto intuitivo de base de datos

Entre las desventajas de que los datos pertenezcan a las aplicaciones en lugar de a una base de datos, están:

- Redundancia: puede que cierta información esté almacenada en más de un sitio. Así se gasta más espacio; y si se actualizan los datos en un sitio, en el otro no están actualizados.
- Inconsistencia: datos actualizados en un sitio y obsoletos en el otro; se registran dos situaciones incompatibles.
- No hay reutilización: hay veces que merece la pena tener ciertos datos preguardados aunque perdamos espacio, para no tener que introducirlos cada vez que queremos utilizarlos.

Solucionar estos problemas utilizando un sistema de archivos tiene ciertas complejidades, como permitir el acceso desde distintas aplicaciones, con distintos lenguajes y sistemas operativos; proteger ciertos datos de usuarios no autorizados… La solución es utilizar un software especial que permita crear y usar almacenes de datos centralizados.

Base de datos: fondo común de información almacenada sin redundancia (en la medida de lo óptimo) en una computadora para que cualquier persona o programa autorizado (los usuarios de una base de datos normalmente son aplicaciones) pueda acceder a ella, independientemente del lugar de procedencia y del uso que se le vaya a dar.

Las operaciones de las bases de datos se recuerdan como CRUD (Create, Read, Update, Delete).

1.2 Sistemas de gestión de bases de datos

Sistema de gestión de bases de datos: conjunto de elementos software con 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

Los elementos involucrados en una base de datos son:

- Datos
    - Integrados (sin redundancia): se juntan todas las aplicaciones
    - Compartidos (útiles a varias aplicaciones)

- Hardware
    - BD centralizada
    - BD distribuida
- Software
    - Programas para definir las estructuras y gestionar la información de la base de datos
    - Programas de aplicación que se conectan con una base de datos para darle y leer datos
- Usuarios
    - Usuario final (terminal)
    - Programador de aplicaciones (el que desarrolla)
    - Administrador: implementación, configuración, organización y rendimiento

Dato operativo: pieza de información que necesita una organización para su funcionamiento

- Ítem básico: elemento sobre el que se puede pedir información
- Atributo: característica de los ítems básicos
- Relación: conexión lógica entre ítems

Objetivos de un SGBD

- Independencia de los datos
- Diseño y utilización orientada al usuario: los datos y aplicaciones deben ser accesibles a los usuarios lo más fácilmente posible
    - Modelo de datos teórico: sencillo de entender y trabajar (modelo de datos relacional)
    - Facilidades de definición
    - Lenguajes de acceso y modificación
- Centralización: los datos deben gestionarse de forma centralizada e independiente de las aplicaciones
- No redundancia: los datos no deben estar duplicados sin ninguna finalidad. Gestionar los accesos concurrentes
- Consistencia: los datos no deben tener fallos lógicos, deben respetar las reglas definidas en la organización, y cada operación debe llevar a la BD de un estado válido (consistente) a otro
- Fiabilidad: los datos deben estar protegidos contra fallos, por lo que debe haber mecanismos de recuperación y relanzamiento de transacciones. Cuanto más delicados son los datos, más sofisticada es la BD. Hay que estar preparados para tener los datos duplicados, de forma redundante pero transparente al usuario, en distintas localizaciones.
- Seguridad: no todos los datos deben ser accesibles a todos los usuarios, por lo que debe haber mecanismos de gestión de usuarios y privilegios; y mecanismos de gestión de datos.

1.3 Ventajas de utilizar un SGBD

Para el usuario:

- Usuario final: puede acceder a los datos
- Programador de aplicaciones: elimina problemas de diseño lógico y físico, depura errores y facilita el mantenimiento en general
- Administrador de BD: le permite adoptar decisiones a todos los niveles y ejecutarlas

Para el sistema:

- Control centralizado: fiabilidad, consistencia, seguridad
- Critérios de uniformización
- Generación de nuevas aplicaciones
- Equilibrio entre requerimientos conflictivos

1.4 Concepto de independencia

Independencia: los datos se organizan independientemente de las aplicaciones que los vayan a usar y de los archivos en los que vayan a almacenarse.

Independencia física: el diseño lógico de la BD debe ser independiente del almacenamiento físico de los datos. Esto permite realizar cambios en estructura física sin alterar la lógica de la aplicación (evita tener que reprogramar cada vez que se usa el programa); y liberar a las aplicaciones de gestionar los aspectos relativos al almacenamiento.

Independencia lógica: poder hacer cambios en el modelo lógico sin tener que modificar la aplicación. No siempre se puede conseguir ya que algunos cambios requieren cambios en las vistas de usuario. Cada aplicación debe poder organizar los datos según sus esquemas y acceder a los datos que le son necesarios y le conciernen (vistas de usuario).

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

TEMA 2: ARQUITECTURA DE UN SGDB

Índice:

2.1 Niveles generales de la arquitectura

Hay tres capas de organización de una base de datos:

- Nivel Interno: representación de la base de datos más cercana a la estructura de almacenamiento físico. Estructuras de datos sobre las que se sustentan los niveles superiores. Este nivel lo ven los organizadores, programadores…
- Nivel Conceptual: abstracción global de la base de datos. Integra todas las percepciones que los distintos usuarios tienen. Este nivel lo ven los organizadores, programadores…
- Nivel Externo: se definen las distintas percepciones de la base de datos por parte de los usuarios. Hay distintas visiones, adaptadas a cada usuario. Este nivel lo ve el usuario

Transformación o correspondencia entre niveles: conjunto de normas que establece cómo se definen los datos de un nivel en términos de otro. Es fundamental para establecer la independencia lógica y física.

- Transformación conceptual/interna: dice cómo se construye el nivel conceptual a partir del nivel interno. Tiene que ver con la independencia física: hay un cambio en el nivel interno, se va la correspondencia y no varía el nivel conceptual
- Transformación externa/conceptual: describe cómo se construye el nivel externo a partir del nivel conceptual. Tiene que ver con la independencia lógica: hay un cambio en el nivel conceptual, se cambia la correspondencia y no varía el nivel externo. No siempre es posible esta independencia.
- Transformación externa/externa: algunos SGBD permiten describir esquemas externos en términos de otros esquemas externos. Tiene que ver con la independencia lógica: hay un cambio en el esquema externo subyacente, se cambia la correspondencia y no varía el esquema externo dependiente. No siempre es posible esta independencia.

Lenguajes de una base de datos:
Según ANSI/SPARC, es recomendable tener un lenguaje específico orientado a los datos, para poder definirlos, controlarlos y modificarlos. DSL (Domain Specific Language) es un lenguaje en el propio SGBD y tiene tres partes distintas: DDL (Data Definition Language), DML (Data Manipulation Language) y DCL (Data Control Language). Se recomienda disponer de cada uno de estos tres lenguajes para cada nivel de la arquitectura.

Lenguaje anfitrión o de aplicación (necesita utilizar el DSL): sirve para el desarrollo de aplicaciones en el SO que trabajan sobre la BD. Proporciona procesamiento avanzado de datos y gestión de la interfaz de usuario. Es un lenguaje adicional que hace falta.
Hay que establecer un mecanismo para trasladar las estructuras de datos y las operaciones de la BD al entorno de procesamiento de la aplicación.

Acoplamiento:
- Débilmente acoplados: lenguajes de propósito general. Se pueden distinguir las sentencias propias de DSL y las del lenguaje anfitrión.
- Fuertemente acoplados: lenguajes y herramientas de propósito específico. No se pueden distinguir las sentencias propias de DSL y las del lenguaje anfitrión. El DSL es el elemento central y se incorporan características para facilitar el desarrollo de aplicaciones.

Alternativas para implementar el acoplamiento débil:
- APIs de acceso a BD (lo más habitual)
- DSL inmerso en el código fuente del lenguaje anfitrión (hacen falta estrategias de preprocesamiento)

Alternativas para implementar el acoplamiento fuerte:
- Ejecución de Java sobre una máquina virtual implantada en el propio SGBD

Existen entornos de desarrollo específicos para facilitar el desarrollo de aplicaciones de gestión. Con ellos se resuelve la creación de bases de datos y se da una herramienta para crear las aplicaciones.

2.2 El nivel externo
Es la parte de la BD que es relevante para cada usuario, pues cada usuario o aplicación percibe la BD de una manera diferente. Sólo ve aquellas entidades, relaciones y atributos que le son de interés. Está conformada por datos calculados a partir de los que hay.

2.3 El nivel conceptual
Hay una visión global de todos los datos, toda la BD desde el punto lógico. La estructura lógica de los datos dice qué datos están almacenado y la relación que hay entre ellos.
Este nivel representa todas las entidades, atributos y relaciones; las restricciones que afectan a los datos (de carácter genérico o específicas); información semántica de los datos; e información de seguridad y de integridad. Da soporte a cada vista externa y no debe contener detalles de almacenamiento.

2.4 Nivel interno
Es la representación física de la BD en el ordenador, cómo están almacenados los datos. Busca el rendimiento óptimo del sistema (se tienen herramientas para ello).
Este nivel representa las estructuras de datos; organizaciones en ficheros; comunicación con el SO para gestionar el almacenamiento (el SO tiene que almacenar en sus dispositivo de almacenamiento secundario, los datos de la BD); compresión y cifrado de los datos…
Parte de las responsabilidades de este nivel las realiza el SO (se encarga de gestionar el nivel físico).

Ej:

Visión conceptual:

```
Profesor = registro de
        NRP             campo alfanumérico de 10 caracteres
        Sueldo          campo decimal de 8+2 dígitos
        Departamento    campo alfanumérico de 30 caracteres
fin Profesor
```

Visión externa 1:

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

Visión externa 2:

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

Visión interna:

```
Profesor_interno BYTES = 74
        NRP TYPE=BYTES(10), OFFSET=0
        Sueldo TYPE=WORD(2), OFFSET=60
        Departamento TYPE=BYTES(10), OFFSET=64
```

2.5 El administrador de la base de datos (DBA)

El DBA es una figura muy relevante en el contexto de los SGBDs. Se encarga de:

- Elaboración del esquema conceptual
- Decidir la estructura de almacenamiento en el nivel interno
- Conexión con usuarios
- Definir las restricciones de integridad
- Definir e implantar la política de seguridad (IMPORTANTE: siempre es función del DBA)
- Definir e implantar la estrategia de recuperación frente a fallos (IMPORTANTE: siempre es función del DBA)
- Optimización del rendimiento (de lo más importante del nivel interno)
- Monitorizar el SGBD

2.6 Tipos de arquitecturas de implantación

La forma de gestionar la información, de ejecutar los programas y de interactuar con el usuario han ido evolucionando.

Inicialmente había un esquema centralizado, donde toda la carga de gestión y procesamiento recaía en servidores centrales y el usuario accedía mediante terminales. El ordenador principal es un superordenador donde se instala todo: el SGBD y los programas de aplicación. Las desventajas es que este superordenador es caro y difícil de configurar. Los terminales piden al ordenador principal que ejecute programas de aplicación.

La solución es una arquitectura cliente-servidor. El servidor tiene la BD y el servicio de escucha de peticiones. Los PCs conectados en red con el servidor tienen los programas de aplicación y el servicio de enlace cliente que interactúa con el de escucha instalado en el servidor. La desventaja es que cuando se actualizan los programas de aplicación, hay que ir actualizándolos en todos los servidores en los que están instalados.

Actualmente se usa una arquitectura articulada en tres niveles de procesamiento:

- Nivel de servidor de datos: 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: proporcionan los programas de aplicación a clientes ligeros. Disponen de entornos de ejecución de aplicaciones
- Nivel de cliente: PCs ligeros dotados de configuraciones basadas en estándares abiertos. En muchos casos se pueden ejecutar las aplicaciones desplegadas en un navegador web con el soporte de ejecución necesario, por lo que no hace falta instalar otras aplicaciones, sino que se accede directamente a ellas. Tienen menos dependencia del hardware y del SO al abordar la ejecución de las aplicaciones

Ventajas: se reduce significativamente el mantenimiento de los clientes (instalación, configuración y actualización de las aplicaciones realizada en el servicio, no en cada cliente), y hay mayor facilidad y flexibilidad para el usuario (puede acceder casi desde cualquier puesto y dispositivo).
Desventajas: mayor complejidad en la configuración y administración de los servidores de aplicaciones y en el desarrollo de las aplicaciones.

TEMA 3: MODELO DE DATOS RELACIONAL

Índice:

3.1 Definición de modelo de datos

Los datos pasan por un proceso de transformación:

- Datos generales sobre una organización concreta
- Datos operativos que se manejan en la organización
- Esquema conceptual de la base de datos
- Modelo lógico de la base de datos: trasladamos a un esquema lógico en función de una estructura implementable (tablas)
- Implementación de la base de datos en un DBMS: lo implementamos en un SGBD

Un modelo de datos es un mecanismo formal para representar y manipular información de manera general y sistemática. Debe consar de notación para: escribir datos; describir operaciones; y describir restricciones de integridad.
La necesidad de usar modelos de datos, viene de que cada esquema se describe utilizando un lenguaje de definición de datos. Este lenguaje es de muy bajo nivel y está muy ligado al SGBD. Por ello, hacen falta otros mecanismos de más alto nivel que permitan describir 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 describan de una forma entendible y manipulable.

Los modelos de datos se clasifican en: basados en registros, basados en objetos y físicos. Los dos primeros se usan en los niveles externo y conceptual; y el último en el nivel interno.

- Basado en registros: modelo de datos jerárquico, modelo de datos en red, modelo de datos relacional (1969)

3.2.1 El modelo de datos relacional: Estructura de datos

El modelo de datos 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. Este modelo fue propuesto por E.F. Codd en 1970. Abarca tres ámbitos distintos de los datos:

- Estructuras de almacenamiento: el usuario percibe la información de la base de datos estructurada en tablas
- Integridad: las tablas deben satisfacer ciertas condiciones que preservan la integridad y la 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/interno, el sistema es libre de almacenar los datos en el formato más adecuado.
El dominio es el rango de valores donde toma sus datos un atributo. Se considera finito.
Una tupla es cada una de las filas de una relación.
La cardinalidad de una relación es el número de tuplas que contiene. Varía en el tiempo.
El esquema de una relación R son los atributos de la relación junto con su dominio.
El grado de una relación es el número de atributos de su esquema y no varía con el tiempo.
Una instancia de una relación es el conjunto de tuplas que la componen en cada momento.
Una BD relacional es una instancia de una base de datos junto con su esquema.

Propiedades de un modelo de datos relacional:

- Condición de normalización: todos los valores de los atributos de una relación son atómicos, es decir, valores no estructurados. Cuando una relación cumple la primera condición de normalización, se dice que está en Primera Forma Normal.
    - Consecuencias: no hay valores tipo conjunto, tipo registro, ni tipo tablas.
    - Problema: todas las representaciones son extensivas, es decir, no se pueden representar directamente información del tipo "el valor del atributo asignaturas de un alumno es: (FBD, ALG, LD)"
- Consecuencias de la definición: no hay tuplas duplicadas y no hay orden en las filas ni en los atributos, por lo que el acceso es por nombre de atributo y valor.

- Representación física: archivo, registros y campos
- Representación intuitiva: tabla, filas y columnas
- Modelo matemático: relación, tuplas y atributos

Si no se conoce el valor de un atributo para una determinada tupla, se le asigna un valor nulo a dicho atributo. Un valor nulo puede ser un valor desconocido, o un atributo no aplicable, pero dicho valor está en los dominios de la base de datos.

3.2.2 El modelo de datos relacional: Restricciones de integridad

Las restricciones o reglas de integridad son condiciones para preservar la corrección semántica de una base de datos. Las hay específicas del problema (ej: 0<= edad <= 100), o propias del papel de los atributos en el esquema (ej: imparte.NRP € profesor.NRP (un profesor inexistente no puede impartir una asignatura)).

Una superclave es un conjunto cualquiera de atributos que identifica unívocamente a cada tupla de una relación. En una relación puede que más de un conjunto de atributos cumplan las condiciones para ser clave. Estos son las claves candidatas, de entre las claves hay q elegir una como principal, la clave primaria.
Una clave candidata es un atributo o el conjunto más pequeño posible de ellos que identifica a cada tupla en la relación.
Una clave externa (puede haber más de una en una misma relación) es un atributo de una relación, que coincide con la clave primaria de otra relación. Establecen relaciones de inclusión entre los dominios activos (valores del dominio de un atributo de una relación que está presente en la instancia de la relación) de la clave externa y la clave referenciada.
Una clave de una tabla es una restricción, pues no se pueden insertar dos entradas con la misma clave.

Condiciones generales:

- Condiciones de integridad: normas que mantienen la corrección de una base de datos
- Integridad genérica: generan reglas de integridad aplicadas a una BD concreta
    - Integridad de entidad: no se debe permitir que una entidad sea representada en la base de datos si no se tiene una información completa de los atributos que son clave primaria de la entidad. Las claves no pueden estar repetidas. Las claves primarias no pueden ser nulas porque identifican la tupla. Las candidatas si pueden ser nulas
    - Integridad referencial:
        - Rechazar tupla insertada si el valor de la clave externa no coincide en la relación referenciada para alguna tupla en el valor de su clave primaria. También si el valor de la clave externa es NULO y el diseño no lo permite. Si ocurre esto en modificaciones, se rechaza la modificación.
        - Actualizar en cadenas las claves externas que la referencien si se actualiza la clave primaria de la relación referenciada.
        - Si se borra la clave primaria en la relación referenciada, se borran todas las tuplas que la referencian o se pone el valor de la clave externa a nulo en todas esas tuplas.

El SGBD se encarga de mantener la unicidad de la clave primaria y de las claves candidatas en operaciones de inserción o actualización; y la integridad de identidad (se deben rechazar modificaciones que asignen un valor nulo a algún atributo de la clave primaria).

3.3 Otros modelos de datos

Modelo jerárquico: fue el primero en implementarse físicamente. No había integridad ya que carecía de un lenguaje de consulta. La BD es una colección de instancias de árboles. Plasma de manera directa las relaciones muchos a muchos (hay que duplicar la información sobre las entidades involucradas) y uno a uno.

- Inconvenientes: los árboles son difíciles de almacenar en ficheros, ya que hay que mantener los ficheros. Existe una dependencia obligatoria, pues un registro de tipo secundario no se puede insertar mientras no exista uno del tipo raíz con el que enlazarlo. Redundancia necesaria para plasmar relaciones muchos a muchos.

Modelo en red: utiliza grafos cuya topología depende de las conexiones existentes entre las entidades. Los nodos son los registros; los arcos los enlaces; y las relaciones entre conjuntos de entidades son conectores (atributos propios de la relación). Cualquier registro puede relacionarse con cualquier otro. La BD es una colección de instancias de grafos. La estructura es muy genérica, puede plasmar todo tipo de relaciones e implementa directamente las relaciones muchos a muchos.

- Ventajas: estructura algo más homogénea que permite insertar nuevas entidades de un conjunto de forma independiente.
- Inconvenientes: la existencia de enlaces entre los registros hace que las operaciones del DDL y el DML sigan siendo complejas de implementar y utilizar.

Comparativa:

- Representación:
    - Relacional: un sólo elemento para la representación, conexiones lógicas, representación de relaciones n:m simétrica, identidad por valor
    - Basado en grafos: dos elementos para la representación, conexiones en el modelo físico subyacente, representación de conexiones n:m imposible en modelos jerárquicos y difícil en modelos en red, identidad por posición
- Consulta:
    - Relacional: obtención de la consulta como resultado global, lenguajes declarativos y procedimentales
    - Basado en grafos: mecanismo de navegación por punteros, lenguajes procedimentales

Modelo orientado a objetos: abstrae un modelo de la realidad en forma de conjunto de objetos que interaccionan entre sí por medio de mensajes. Tiene una gran capacidad de modelado, sin embargo, es difícil de gestionar, lo que supone un rendimiento menor.
Los SGBD relacionales tienen algunos inconvenientes, como la pobre representación de las entidades del mundo real, sobrecarga semántica de las relaciones, la estructura relacional que es muy estricta…

Modelo objeto-relacional: tiene solidez de SGBD relacionales y capacidad de modelado de la orientación a objetos.

Cubos de datos - sistemas multidimensionales: las bases de datos operacionales, utilizan un modelo de datos relacional o de objetos, y OLTP. Las bases de datos analíticas (DW), utilizan un modelo multidimensional y OLAP.

Una BD operacional está orientada a DBAs y programadores de apps. Soporta el funcionamiento diario de la organización. Se encarga de procesar transacciones utilizando datos actuales, aislados y relacionales, de los que hace un uso repetitivo y sencillo. Se accede a los datos a través de la lectura/escritura y de transacciones simples. Es crucial la gestión de transacciones y la consistencia de los datos.

OLTP (Online transaction processing):

- Aplicaciones OLTP: realizan operaciones diarias, estructuradas y repetitivas
- Requieren datos detallados y actualizados al día
- Las operaciones afectan a pocos registros a los que se accede a través de la clave primaria
- Es muy importante su consistencia y fiabilidad
- Su objetivo se basa en optimizar la gestión de transacciones

Un DW (Data Warehouse - Almacén de datos) está orientado a ejecutivos. Soporta la toma de decisiones. Hace un procesamiento analítico de datos históricos, resumidos y multidimensionales, de los que hace un uso espontáneo. El acceso se hace mediante lectura y consultas complejas. Es crucial la gestión de consultas y la oportunidad de los datos.

OLAP (Online Analytical Processing):

- Aplicaciones OLAP: realizan decisiones que aparecen en el momento, no son tan previsibles
- Requieren datos consolidados, resumidos y de tipo histórico
- Necesitan resolver consultas complejas, tanto predefinidas como espontáneas, y que pueden involucrar a millones de registros
- Necesitan estructuras de datos diferentes
- El procesamiento de consultas y el tiempo de respuesta son más importantes que el control de transacciones

Modelado dimensional: los datos se organizan como un conjunto de medidas descritas por aspectos comunes del negocio. Es útil para agregar/desagregar datos y reordenarlos para el análisis. Está enfocado para trabajar sobre datos numéricos. Es más facil de entender y usar que otros modelos propios de sistemas operacionales (objeto-relacional) y más atractivo visualmente.