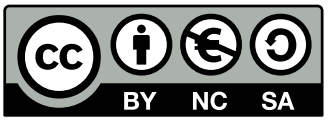
Fundamentos de Bases de Datos

LibreIM

Doble Grado de Informática y Matemáticas
Universidad de Granada

libreim.github.io/apuntesDGIIM



Fundamentos de Bases de Datos

LibreIM

Doble Grado de Informática y Matemáticas
Universidad de Granada

libreim.github.io/apuntesDGIIM

Índice

1 Tema 1. Introducción y definiciones iniciales

1.1 Concepto intuitivo de base de datos

Concepto intuitivo: 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 del uso que se haga.

Operaciones: insertar, obtener, modificar, borrar...

1.2 Bases de datos y sistemas gestores de bases de datos

Base de datos: conjunto de datos comunes a un proyecto almacenados sin redundancia para ser útiles a diferentes aplicaciones.

Sistemas de gestión de base de datos (SGDB o DBMS): conjunto de elementos software con capacidad para definir, mantener y utilizar una base de datos. Un SGDB debe permitir definir estructuras de almacenamiento, acceder a datos de forma eficiente y segura, organizar la actualización de los datos y el acceso multiusuario...

Elementos de una base de datos:

- **Datos**: integrados (sin redundancia) y compartidos (útiles a varias aplicaciones).
- **Hardware**: base de datos normal o distribuida.
- **Software**: SGBD, y programas para definir las estructuras y gestionar los datos de la base de datos.
- **Usuarios**: usuario final, programador de aplicaciones, administrador.

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

- **Ítem básico**: elemento acerca del que se puede pedir información (sustantivo).
- **Atributo**: características de los ítems básicos (adjetivos).
- **Relaciones**: conexiones lógicas entre ítems.

Cuando se determinan y clasifican de esta forma todos los datos operativos, se obtiene el **esquema lógico** de la base de datos.

Objetivos de un SGBD:

- **Independencia** de los datos.

- **Diseño y utilización** orientada al usuario: los datos y aplicaciones deben ser accesibles a los usuarios de la manera más amigable posible (soportar un modelo de datos teórico, soportar facilidades de definición y soportar lenguajes de acceso y modificación).
- **Centralización**: los datos deben gestionarse de forma centralizada e independiente de las aplicaciones (figura del ABD, utilidades de gestión...).
- **No redundancia**: los datos no deben estar duplicados (gestión de accesos concurrentes).
- **Consistencia**: los datos deben ser consistentes, sin fallos lógicos (mecanismos de mantenimiento de integridad).
- **Fiabilidad**: los datos deben estar protegidos contra fallos catastróficos (mecanismos de mantenimiento de recuperación y relanzamiento de transacciones).
- **Seguridad**: no todos los datos deben ser accesibles a todos los usuarios (mecanismos de gestión de usuarios y privilegios, mecanismos de protección de información).

1.3 Ventajas de utilizar un SGBD

Para el usuario:

- **Usuario final**: puede acceder a los datos.
- **Programador de aplicaciones**: eliminar problemas de diseño lógico y físico, depuración de errores y mantenimiento general.
- **Administrador de BD**: no existiría.

Para el sistema:

- **Control centralizado**: fiabilidad, consistencia, seguridad...
- **Criterios de uniformación**.
- **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 ficheros en los que vayan a almacenarse.

Independencia física: el diseño lógico de la BD, a todos los niveles, debe ser independiente del almacenamiento físico de los datos. Esto permite realizar libremente cambios en la estructura física y descargar a las aplicaciones de problemas físicos.

Independencia lógica: existen dos tipos de estructura lógica: esquema lógico general (vista global) y vistas de usuario (datos a los que se deja acceso a un usuario). Cada aplicación debe poder organizar los datos según sus propios esquemas

y acceder a los datos que le son necesarios y le conciernen. La independencia lógica persigue que los cambios en el esquema lógico general no afecten a las vistas de usuario de manera que las aplicaciones no necesiten ser modificadas. No siempre se puede conseguir.

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

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

2.1 Una arquitectura con tres niveles

£Por qué organizar en niveles? Los usuarios pueden acceder a los mismos datos desde distintas perspectivas. Si un usuario cambia la forma de ver los datos no influye al resto.

La organización global de los datos puede cambiarse sin afectar a los usuarios.

Los usuarios no tienen por qué gestionar aspectos relativos a la representación física de los datos. El administrador de la base de datos puede cambiar la forma de representar los datos sin influir en los usuarios.

La percepción de los datos en un SGBD puede hacerse siguiendo tres niveles de abstracción:

- **Nivel interno**: constituye la representación de la BD 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.
- **Nivel conceptual**: supone una abstracción global de la BD que integra y aglutina todas las percepciones que los usuarios tienen de ella. Tiene una visión global de los datos pero ningún detalle de almacenamiento.
- **Nivel externo**: a este nivel se definen todas las percepciones particulares de la BD por parte de los usuarios. Cada usuario puede tener su propia visión de la BD.

2.2 Correspondencia entre niveles

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

Transformación interna/conceptual: cómo se organizan las entidades lógicas del nivel conceptual en términos de registros y campos almacenados en el nivel interno. Independencia física:

- Varía el nivel interno
- Cambia la correspondencia
- No varía el nivel conceptual

Transformación externa/conceptual: describe un esquema externo en términos del esquema conceptual subyacente. Independencia lógica:

- Varía el nivel conceptual
- Cambia la Correspondencia
- No varía el nivel externo Esta transformación no siempre es posible

Transformación externa/externa: algunos SGBD permiten describir esquemas externos en términos de otros esquemas externos. Independencia lógica:

- Varía el esquema externo subyacente
- Cambia la correspondencia
- No varía el esquema externo dependiente

2.3 Lenguajes de una BD

Recomendación ANSI/SPARC: disponer de un lenguaje específico orientado a los datos (definición, control, manipulación). Sublenguaje de datos (DSL) implementado en el propio SGBD. Tiene distintas partes:

- **DDL**: Data Definition Language. Sublenguaje de definición de datos. Subconjunto del DSL destinado a la definición de estructuras de datos y esquemas en la BD.
- **DML**: Data Manipulation Lanquage. Sublenguaje de manipulación de datos. Subconjunto del DSL mediante el que podemos introducir datos en los esquemas, modificarlos, eliminarlos y consultarlos. También debe permitir consultar la estructura de los esquemas definidos en la BD.
- **DCL**: Sublenguaje de control de datos, que permite gestionar los requisitos de acceso a los datos y otras tareas administrativas sobre la BD.

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

En la práctica todos estos sublenguajes se presentan bajo una implementación única. Cada sentencia trabaja sobre uno o varios niveles. Un sistema de privilegios discrimina quién puede ejecutar qué.

La industria ha seguido un camino diferente: lenguajes de datos estándares.

El ejemplo más destacado es SQL: SQL89, SQL92 y SQL3.

Lenguaje anfitrión o de aplicación: desarrollo de aplicaciones en el SO que trabajen sobre la BD. El propósito general es C/C++, Java, C#. Las herramientas de desarrollo específicas son Developer de Oracle, Oracle Application Express (Oracle APEX), Sybase PowerBuilder, IBM Rational Application Developer... Además, proporciona procesamiento avanzado de datos y gestión de la interfaz de usuario. Hay que establecer un mecanismo para traducir: estructuras de datos y operaciones. Acoplamiento:

- **Débilmente acoplados** (si se pueden distinguir):
  - Lenguajes de propósito general
  - El programador puede distinguir:

    Sentencias propias del lenguaje anfitrión
    Sentencias dispuestas para acceder a la BD a través del DSL
- **Fuertemente acoplados** (si no se pueden distinguir):
  - Lenguajes y herramientas de propósito específico
  - Se parte del DSL como elemento central y se le incorporan características procedimentales para facilitar el desarrollo de aplicaciones.

Alternativas para implementar el **acoplamiento débil**:

- **APIs de acceso a BD**:
  - ODBC – Open Database Connectivity
  - JDBC – Java Database Connectivity
- **DSL inmerso en el código fuente del lenguaje anfitrión**: el programador escribe un código híbrido. Mezcla sentencias del lenguaje anfitrión con sentencias DSL. Hay un preprocesador que luego lo transforma.

Alternativas para implementar el **acoplamiento fuerte**: diversas propuestas (la mayoría propietarias): PL/SQL de Oracle (Extensión Procedural para SQL). Por ejemplo: ejecución de Java sobre una máquina virtual implantada en el propio SGBD.

También han aparecido numerosos entornos de desarrollo específicos para el desarrollo de aplicaciones de gestión: Diseñadores de informes, diseñadores de formularios...

2.4 Nivel externo

Parte de la BD que es relevante para cada usuario. Sólo aquellas entidades, relaciones y atributos que le son de interés. Representadas de la forma que le interesa al usuario, por ejemplo: nombre completo o nombre y apellidos; fecha o día, mes y año... También hay datos calculados a partir de los que hay: edad, ventas totales...

2.5 Nivel conceptual

Visión global de los datos. Estructura lógica de los datos: qué datos están almacenados y qué relaciones hay entre ellos. Este nivel representa:

- Todas las entidades, atributos y relaciones.

- Las restricciones que afectan a los datos.
- Información semántica sobre los datos.
- Información de seguridad y de integridad.

Además, da soporte a cada vista externa y no debe contener ningún detalle de almacenamiento.

2.6 Nivel interno

Representación física de la BD en el ordenador, es decir, cómo están almacenados los datos. El objetivo es buscar el rendimiento óptimo del sistema. Representa:

- Estructuras de datos.
- Organizaciones en ficheros.
- Comunicación con el SO para gestionar el uso de unidades de almacenamiento.
- Compresión de datos, encriptación...

Parte de las responsabilidades de este nivel las realiza el SO, se le suele llamar nivel físico. No existe una división clara, depende de cada SGBD y de cada SO.

2.7 Ejemplo

Ejemplo de Gestión Docente Universitaria:

- Item básico PROFESOR
- Identificado por: Número de registro personal (NRP).
- Caracterizado por: nombre y apellidos, sueldo y departamento al que pertenece
- **Visión conceptual**:

```
1  Profesor = registro de
2  NRP
3  campo alfanumérico de 10 caracteres,
4  Apellidos campo alfanumérico de 30 caracteres,
5  Nombre
6  campo alfanumérico de 20 caracteres,
7  Sueldo
8  campo decimal de 8+2 dígitos,
9  Departamento
10 campo alfanumérico de 30 caracteres
11 fin Profesor.
```

- **Visión externa 1**:

  - Gestión de personal.
  - Lenguaje A.

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

- **Visión externa 2**:
  - Ordenación académica.
  - Lenguaje B.

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

- **Visión interna**:

```
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
```

2.8 El administrador de la BD

El DBA es una figura de primordial relevancia en el contexto de los SGBD. Algunas de sus funciones son:

- **Elaboración del esquema conceptual**. Análisis de las necesidades de información de la empresa, identificación de los datos operativos, elaboración del esquema lógico, implantación del esquema conceptual.
- **Decidir la estructura del almacenamiento en nivel interno**. Esquema interno. Correspondencia conceptual/interna asociada.
- **Conexión con usuarios**. Análisis de requerimientos, diseño lógico, codificación del esquema externo, correspondencias externo/conceptual.

- **Definir restricciones de integridad**. Establecer reglas genéricas ye específicas. Si es posible, incluir integridad en el esquema conceptual.
- **Definir e implantar política de seguridad**. Gestión de usuarios, gestión de privilegios.
- **Definir e implantar la estrategia de recuperación frente a fallos**. Los SOs y los SGBDs suelen incorporar facilidades para afrontar los fallos: SGBDs redundantes y RAID (Redundant Array of Inexpensive Disks). El DBA puede y debe realizar copias de seguridad de la BD. También gestiona la política de gestión de transacciones (operaciones lógicas que ocurren en la BD).
- **Optimización del rendimiento**. Liberar espacio no utilizado, reorganizar las operaciones para que se ejecuten de forma más rápida, determinar la necesidad de nuevos recursos hardware, establecer prioridades en el uso de los recursos.
- **Monitorizar el SGBD**. Seguimiento continuo de la actividad del sistema. Auditar el acceso a los usuarios a los diversos recursos de la BD. Comprobar los niveles de uso de los sistemas de almacenamiento. Evaluar la eficiencia con que se realizan las operaciones.

2.9 Tipos de arquitecturas de implantación

El concepto de SBD ha evolucionado paralelamente al desarrollo del informática en la forma de gestionar la información, de ejecutar los programas y de interactuar con el usuario.

Inicialmente se utilizaba un sistema **centralizado** donde toda la carga de gestión y procesamiento de información recaía en servidores centrales. El usuario accedía mediante terminales. En el ordenador principal se encontraban el SGBD y los programas de aplicación.

El principal problema de este sistema es el elevado coste de los ordenadores principales con la aparición del PC. La solución fue desplazar la ejecución de los programas de usuario y las interacciones hasta los PCs (reducción de costes en hardware). Surge la primera aproximación **cliente/servidor**. El servidor alojaba la base de datos y un servicio de escucha de peticiones, mientras que el cliente, los PCs conectados al servidor, tenían los programas de aplicación y el servicio de enlace cliente que interactúa con el servicio de escucha instalado en el servidor. Con el desarrollo de las comunicaciones este enfoque deriva en un enfoque distribuido.

El principal problema es el alto coste del mantenimiento de los PCs (instalación, configuración y actualización), que se soluciona separando en las aplicaciones: la parte que interactúa con el usuario (interfaz) de la parte de ejecución lógica del programa.

Actualmente se utiliza una arquitectura articulada entre 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 del servidor web que proporcionan programas de aplicación a clientes ligeros, que disponen de entornos de ejecución de aplicaciones: usando estándares, protocolos de red TCP/IP, protocolo HTTP, despliegue de Applets Java a ejecutar en navegadores con soporte de máquina virtual Java, Servlets, JSP, ASP...
- **Nivel de cliente**. PCs ligeros dotados de configuraciones basadas en estándares abiertos. Basados en el carácter portátil con que se distribuyen las aplicaciones desde los servidores de aplicaciones. Menos dependencia del hardware y SO a la hora de abordar la ejecución de las aplicaciones.

Ventajas: reducción significativa en cuanto al mantenimiento de los clientes. Mayor facilidad y flexibilidad para el usuario.

Inconvenientes: mayor complejidad en la configuración y administración de los servidores de aplicaciones. El desarrollo de las aplicaciones conforme a este modelo distribuido es más costoso.

3 Tema 3. Modelos de datos

3.1 Cocepto de modelo de datos

Modelo de datos es el mecanismo formal para representar y manipular información de manera general y sistemática. Debe costar de:

1. Notación para describir datos.
2. Notación para describir operaciones.
3. Notación para describir reglas de integridad.

Objetivo: describir modelos que representen los datos y los describan de una forma entendible y manipulable. En relación a la arquitectura ANSI/SPARC:

- Nivel externo: modelo de datos externo.
- Nivel conceptual: modelo de datos conceptual.
- Nivel interno: modelo de datos interno.

Clasificación:

- Basados en registros.
- Basados en objetos.
- Físicos.

Utilización:

- Basados en registros y objetos: nivel externo y conceptual.
- Físicos: nivel interno.

3.2 Modelos de datos basados en registros

3.2.1 Modelo jerárquico

Fue el primero en implementarse físicamente. Carecía de lenguaje de consulta. La estructura de datos básica es el árbol. La base de datos es una colección instanciada de árboles. Esta estructura plasma de forma muy directa relaciones muchos a uno y relaciones uno a uno, pero necesita duplicar información para las relaciones muchos a muchos.

Inconvenientes:

- Almacenar árboles en un fichero es complejo. Varios tipos de registros y punteros que hay que mantener.

- DML difícil de implementar y usar.
- Dependencia existencial obligatoria de los registros de tipo secundario con respecto a los de tipo raíz.
- Redundancia necesaria para plasmar relaciones muchos a muchos.

3.2.2 Modelo en red

La estructura de datos consiste en grafos cuya topología depende de las conexiones existentes 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.

La base de datos es una colección de instancias de grafos. La estructura es muy genérica. Permite plasmar todo tipo de relaciones e implementar directamente las relaciones muchos a muchos.

Ventajas:

- Estructura más homogénea.
- Permite insertar nuevas entidades en un conjunto de forma independiente.

Problemas:

- La existencia de enlaces entre los registros hace que las operaciones DDL y el DML sigan siendo complejas de implementar y utilizar.

3.2.3 Modelo relacional

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.

Definiciones:

Esquema de una base de datos relacional: colección de esquemas de relaciones junto con restricciones de integridad.

Instancia o estado de una base de datos: colección de instancias de relaciones que verifican las restricciones de integridad.

Base de datos relacional: instancia de una base de datos junto con su esquema.

Claves:

- **Superclave**: cualquier conjunto de atributos que identifica unívocamente a cada tupla de la relación.
- **Clave candidata**: superclave minimal.
- De entre las candidatas (si hubiera más de una) hay que elegir una como principal que se denomina **clave primaria**.

Para describir una relación, se subrayan los atributos que forman su clave primaria.

4 Tema 4. El modelo de datos relacional

4.1 La estructura de datos relacional

Introducido por E. F. Codd en 1970. El modelo de datos relacional abarca tres ámbitos distintos:

- Las estructuras para almacenarlos. El usuario recibe la información de la base de datos estructurada en tablas.
- La 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, el sistema es libre de almacenar los datos en el formato más adecuado(archivo secuencial, archivo indexado, lista con apuntadores...).

Atributo: cualquier elemento de información susceptible de tomar valores Dominio: rango de valores donde toma sus datos un atributo. Relación: dados los atributos $A_i, i = 1, 2, ..., n$, con dominios $Di, i = 1, 2, ..., n$ no necesariamente distintos, definimos la relación asociada a $A_1, ..., A_n$, y lo notaremos como $R(A_1, ..., A_n)$ a cualquier subconjunto de productos cartesianos $D_1 x ... x D_n$.

Tupla: cada una de las filas de la relación. Cardinalidad de una relación: número de tuplas que contiene. Es variable en el tiempo. Grado de una relación: número de atributos de su esquema. Invariable en el tiempo. Esquema de una relación R: Atributos A1: D1,..., A:Dn Instancia de una relación: conjunto de tuplas que la componen en cada momento.

4.2 Propiedades de la estructura de datos relacional:

Condición de normalización. Todos los valores de los atributos de una relación son atómicos. Valor atómico es un valor no estructurado. 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 o tipo tabla. Problema: todas las representaciones son extensivas. Consecuencias de la definición: no hay tuplas duplicadas, no hay orden en las filas ni en los atributos, varias instancias representan la misma relación.

Esquema de una base de datos relacional: colección de esquemas de relaciones junto con restricciones de integridad.

Instancia o estado de una base de datos: colección de instancias de relaciones que verifican las restricciones de integridad.

Base de datos relacional: instancia de una base de datos junto con su esquema.

Algunas veces no se conoce el valor de un atributo de determinada tupla. En esos casos a ese atributo de esa tupla se le asigna un valor nulo. Un valor nulo puede ser un valor desconocido o un atributo no aplicable. En cualquier caso, ese valor es un valor común a todos los dominios de la base de datos.

4.3 Restricciones de integridad

En una relación puede haber más de un conjunto de atributos que puedan ser elegidos como clave. Estos conjuntos se llaman claves candidatas. Una clave candidata es un atributo o conjunto de atributos que identifican a cada tupla en la relación y que, además, no existe un subconjunto de ellos que también identifiquen a la tupla en la relación. Una clave primaria es una clave candidata elegida por el diseñador. Si se verifica la unicidad pero no la minimalidad se denomina superclave.

Condiciones de integridad: normas que mantienen la corrección semántica de una base de datos. Son metarreglas (generan reglas de integridad aplicadas a una base de datos concreta). Existen la integridad de entidad y la referencial.

Integridad de entidad: no se debe permitir que una entidad sea representada en la base de datos si no se tienen una información completa de los atributos que son claves de la entidad. Clave primaria o parte de ella no puede ser valor nulo.

Clave externa: un conjunto de atributos en una relación que es una clave en otra (o en la misma) relación. Podemos ver una clave externa como un conjunto de atributos de una relación cuyos valores en las tuplas deben coincidir con los valores de la clave primaria de las tuplas de otra relación. Si PK es la clave primaria de R y FK la clave externa de S con respecto a R verifica que el dominio activo de FK debe estar incluido en el dominio activo de PC para cualquier instancia de la base de datos.

Integridad referencial: una base de datos en la que todos los valores no nulos de una clave externa referencian valores reales de la clave referenciada en la otra relación cumple la regla de integridad referencial. Si una relación incluye una clave externa conectada a una clave primaria, el valor de la clave externa debe ser, bien igual a un valor ya existente en el dominio activo de la clave primaria o bien completamente nulo (si la semántica lo permite). La integridad referencial mantiene las conexiones entre las bases de datos relacionales. Puede haber más de una clave externa en una relación, y más de una clave externa a la clave primaria de la propia relación.

El SGBD debe encargarse de mantener las siguientes restricciones:

- La unicidad de la clave primaria y las claves candidatas.
- La restricción de integridad de identidad.

- La integridad referencial:
  - En inserción: rechazar tuplas insertadas si el valor de la clave externa no concuerda en la relación referenciada para alguna tupla en el valor de su clave primaria. Si el valor para la clave externa es nulo y el diseño no lo permite habrá que rechazar también esa inserción.
  - En actualización: si se actualiza la clave externa, rechazar la modificación si se producen alguna de las circunstancias descritas en el punto anterior. Si se actualiza la clave primaria de la relación referenciada se deben actualizar en cadena las claves externas que la referencian.
  - En borrado: si se borran la clave primara de la relación referenciada, se debe borrar en cadena todas las tuplas que la referencian o poner valor nulo en la clave externa de todas esas tuplas.

5 Tema 5. Nivel Interno

5.1 Introducción

Sabemos que una BD almacena grandes cantidades de datos y se pretende gestionarlos de forma eficiente, tanto ellos como su almacenamiento. Esto afecta a su organización lógica y física.

El *nivel interno* expresa las operaciones sobre los datos en términos de actuación sobre unidades mínimas de almacenamiento, las páginas o bloques de base de datos. Está implementado en el SGBD y provee al administrador mecanismos para optimizar el acceso y almacenamiento de datos.

El *nivel físico* está implementado en el SO y da al SGBD abstracción sobre el hardware. Este nivel accede al almacenamiento mediante llamadas a los servicios del sistema de archivos proporcionado por el SO.

5.2 Dispositivos de almacenamiento

Existen diferentes dispositivos de almacenamiento con diferentes características. En orden creciente de coste, capacidad y tiempo de acceso los más importantes son: Registros, Caché, Memoria Principal, Discos Magnéticos, Discos ópticos.

- Memoria principal: Hace trabajos de caché de la porción de la BD que ha sido usada más recientemente. Ubica de forma temporal los datos afectados por las operaciones. El nivel interno debe optimizar su uso y garantizar que haya respaldo de la información si cae el sistema.
- Disco duro: El más usado en BD. Son conjuntos de discos magnéticos de dos caras donde cada cara tiene un conjunto de pistas concéntricas que se dividen en sectores con la misma capacidad de almacenamiento.

Cada dispositivo tiene un tiempo medio de acceso (ta), tiempo medio de búsqueda (tb) y tiempo de latencia rotacional (tl). También tiene un tiempo medio entre fallos ($MTBF$). Así, se cumple que:

$$ta = tb + tl$$

5.3 Método de acceso a la BD almacenada

Si tenemos un bloque de la BD en disco, para obtenerlo se siguen una serie de pasos:

1. El SGBD solicita la página al gestor de archivos.
2. El gestor de archivos solicita los bloques de SO al gestor de disco.
3. El gestor de disco hace una operación de E/S a disco.
4. El disco devuelve de la base de datos almacenada los sectores solicitados.
5. El gestor de disco devuelve al gestor de archivos los bloques solicitados.
6. El gestor de archivos manda al SGBD la página almacenada solicitada.

En este proceso, para que el gestor de almacenamiento pueda localizar la página de BD, se utiliza el *Record Identifier*. Cada registro tiene una cabecera y los datos. Los bloques de la BD tendrán un tamaño que es múltiplo de las páginas del SO, de forma que para recuperar un registro almacenado hay que ver en qué página de la BD está. Así, la estructura de almacenamiento debe estar organizada y deben minimizarse las operaciones de E/S a disco.

5.3.1 El gestor de disco del SO

Normalmente, el SGBD interactúa con la BD almacenada mediante el *gestor de disco del SO*. Este, organiza los datos en conjuntos de bloques o archivos de SO y gestiona el espacio libre en disco. Una BD puede obtenerse de uno o varios de estos archivos. Sus funciones fundamentales son: crear y eliminar archivos de SO, añadir y eliminar nuevos bloques o reemplazar estos bloques.

5.3.2 El gestor de archivos del SGBD

Es un componente encargado de:

- Transformar entre campos, registros y archivos almacenados a bloques y conjuntos de bloques para que el gestor de disco lo entienda.
- Organizar los datos para minimizar el tiempo de recuperación y las E/S a disco.

Puede crear archivos almacenados, eliminarlos, recuperar registros de archivos almacenados (buscando en qué página está) o añadir registros a archivos almacenados (buscando en qué pagina es más adecuado, y si no se puede se solicita

una nueva página). También puede eliminar registros de archivos (marcando el espacio como disponible) y actualizar registros en archivos almacenados.

5.4 Representación de la BD en el Nivel Interno

La BD se representa de formas distintas en los distintos niveles de arquitectura del SGBD. Esta representación no tiene por qué coincidir con la representación en el nivel conceptual, y un conjunto de registros del mismo tipo no tiene por qué ser un fichero. El nivel interno debe traducir las estructuras del nivel conceptual a otras intermedias cercanas al almacenamiento real de los datos.

Si la BD en el nivel interno tiene al conjunto de páginas en las que se ubican los registros, tenemos:

- Agrupamiento Intra-Archivo. Se ubican en la misma página registros del mismo tipo. Es el más frecuente.
- Agrupamiento Inter-Archivo. Se ubican en la misma página registros de distinto tipo. Debe existir relación entre ellos (entidades fuerte-débil).

En la vida real, cada SGBD comercial utiliza su organización concreta. No existe una relación directa entre los ficheros almacenados y los ficheros físicos pues los conjuntos de páginas están almacenados en uno o varios ficheros.

5.5 Organización y métodos de acceso

Cuando queremos acceder a los datos de una BD, pretendemos minimizar el número de acceso a disco. Por ello, necesitamos minimizar la cantidad da páginas de una BD involucradas en una de estas operaciones. Existen varias organizaciones usadas para las BD. La calidad de estas se mide en el tiempo de acceso y el porcentaje de memoria ocupada por los datos.

5.5.1 Organización secuencial

Existe un fichero de acceso secuencial donde los registros están almacenados consecutivamente y ordenados por una clave y para acceder a un registro debemos pasar por los registros que le preceden.

- Insertar un registro: implica buscar el bloque que le corresponde, colocarlo si hay sitio y si no o se crea uno nuevo o se crea uno se crea un bloque de desbordamiento. (Es recomendable dejar espacio entre bloques para evitar problemas de reorganización).
- Borrar un registro: implica buscar el registro y borrarlo. Además, puede implicar una reorganización local de los registros de un bloque.

Ambas suponen escribir en el bloque de registro, crear o liberar bloques de datos en el fichero secuencia, crear o liberar bloques de desbordamiento y reorganizar registros entre bloques contiguos.

Es por ello que esta forma tiene grandes inconvenientes que son subsanables mediante el uso de estructuras adicionales que nos permitan localizar los datos de manera más rápida y disminuir el número de bloques de disco transferidos. Las técnicas más populares son la indexación y el acceso directo.

Indexación Pretende disminuir el tiempo de acceso a los datos utilizando una clave de búsqueda.

Fichero secuencial indexado Partiendo de un fichero secuencial, añadimos una estructura llamada fichero índice, cuyos ficheros tienen una clave de búsqueda y un campo de referencia con los RID de los registros. Son registros más pequeños que los del fichero de datos aunque hay el mismo número de registros en ambos.

- Índice primario: la clave de búsqueda es el mismo campo clave por el que se ordena el fichero secuencial de datos.
- Índices secundarios: se construyen sobre otros campos que no sean la clave física del fichero de datos.

Para consultar un valor, podemos consultar:

- Por valor de la clave: sobre el índice localizamos la clave, obtenemos el RID del registro y vamos a disco para recuperar el bloque donde está el registro señalado por el RID.
- Por rango de valores: se busca en el índice por valor de la clave de la cota inferior y se recorren las entradas que están en el intervalo recuperando los registros correspondientes gracias a su RID.

Para insertar un registro se hace lo mismo que en el fichero secuencial, y hay que actualizar el índice.

Para borrar un registro se borra en el fichero de datos y se borra una entrada en el índice.

Como conclusión, estos índices aceleran el acceso pero hay que mantener estos índices por lo que se ralentizan otras operaciones.

Índices no densos Los índices suelen ser muy grandes pues contienen todos los registros del fichero que indexan. Por tanto, aparecen los índices no densos que se componen por la clave de búsqueda y la dirección de comienzo del bloque donde está el registro deseado. El número de registros se reduce al número de bloques del fichero de datos.

La búsqueda ahora es diferente, pues una vez se encuentra el bloque donde podría estar el registro hay que cargarlo en memoria y hacer búsqueda secuencial en ese bloque. Además, no se garantiza encontrar el registro hasta consultar el bloque completo.

Los índices no densos sólo se pueden definir sobre la clave física. Su mantenimiento es menos costoso: la inserción y borrado son menos frecuentes pues solo ocurren cuando la operación afecta al valor que representa al bloque completo.

Índices jerárquicos Estos índices quieren disminuir el tiempo necesario para encontrar un registro. Para ello, crearán índices multinivel, formados por índices construidos sucesivamente unos sobre otros. El tamaño de bloque se establece para optimizar las operaciones de acceso al disco. Así se reduce el acceso a disco para encontrar un registro pero es más difícil mantener los índices.

Los **Árboles B+** son una generalización de los árboles binarios balanceados en la que los nodos pueden tener más de 2 hijos. Los valores de la clave se encuentran almacenados en los nodos hoja. Un árbol B+ de orden M (máximo número de hijos que puede tener cada nodo) tiene en cada nodo como máximo M hijos y como mínimo (M+1)/2 hijos; todos los nodos hoja están al mismo nivel y las claves contenidas en cada nodo nos llevan al nodo del nivel inferior.

Se pone como restricción que los valores de clave C_i están ordenados dentro del nodo y que los valores x del subárbol apuntado por P_i cumplen:

$$C_{i-1} \leq x < C_i$$

Excepto para $i = 1$ ($x < C_1$) e $i = n$ ($x \geq C_n$).

Los nodos hoja tienen punteros al siguiente nodo hoja. La lista concatenada de nodos hoja es útil para hacer consultas por intervalos. Las claves aparecerán ordenadas en cada nodo y todas deben ser menores que las del siguiente nodo. Los nodos deberán estar rellenos como mínimo hasta la mitad.

- Consulta: Para localizar un registro, se bajan niveles desde la raíz, buscando el registro en el nodo hoja y recuperando el registro del fichero de datos por el RID. Para consultar en un rango se localiza el nodo hoja con el valor inferior y se recorren los demás nodos hoja hasta encontrar el superior.
- Inserción y borrado: Los algoritmos utilizados deben garantizar que el árbol siga siendo equilibrado.

Los árboles B+ en BD son variaciones del árbol B+ de orden elevado en la que se procura que cada nodo tenga casi el mismo almacenamiento que un bloque de datos, reduciendo los acceso a disco. En los nodos *intermedios* solo están los rangos de valores de la clave y los punteros a los nodos hijo. En los nodos *hoja* , que están enlazados, están los valores de clave ordenados junto con los RIDs que apuntan a las tuplas que contienen ese valor de clave.

También existen las Tablas Organizadas por Índice. En este caso, las hojas contienen las tuplas en lugar del RID. Así, una IOT solo puede estar organizada de esta forma mediante una clave aunque se pueden definir índices adicionales basados en otras claves.

Tabla normal |

IOT	
Identificador único ROWID (RID)	
Identificado por clave primaria	
ROWID implícito Soporta varios índices	No
ROWID Sólo un índice IOT, varios B-tree	
Una recuperación completa devuelve las tuplas desordenadas	Una
recuperación completa devuelve las las tuplas ordenadas según CP	

Índices Clave invertida Los índices por clave invertida invierten los datos del valor de la clave. Es adecuado para búsquedas basadas en predicados y mejora el rendimiento de la inserción de tuplas si se insertan de forma ascendente para valores de la clave. También mejora accesos concurrentes.

Índices BITMAP En los índices BITMAP, para cada valor que toma la clave almacena una secuencia de bits (tantos como tuplas contenga la tabla), en los que hay un 1 si el valor está presente en la tupla y un 0 si no lo está.

B-tree |

BITMAP	
Adecuado para columnas que tomen muchos valores	Adecuado para
columnas que tomen pocos valores	
Actualización de claves poco costosa	Actualización de
claves muy costosa	
Ineficiente para consultas usando predicados OR	Eficiente para
consultas usando predicados OR	

5.5.2 Acceso directo

El acceso directo es una forma de acceder a un registro almacenado. En este caso, no tenemos estructura adicional sino que usamos un algoritmo para identificar la posición del registro deseado. Para ello, debemos tener un campo que identifique unívocamente al registro.

Lo usual es que no se pueda establecer una clave física totalmente correlativa y única, por lo que nuestro algoritmo deberá tener una entrada de un campo clave y proporcionará una salida que será un valor entero positivo transformable en RID.

Estos algoritmos de direccionamiento no suelen mantener el orden de la clave. Los registros no están almacenados según el orden de su clave física. Es por ello por lo que tendremos problemas a la hora de recuperar intervalos de datos.

Los algoritmos usados son variados. Si la clave es alfanumérica, se transforma a un valor numérico. Algunos comunes son los de los cuadrados centrales,

congruencias, desplazamiento o conversión de base.

Estos algoritmos tienen varios problemas:

- Es muy difícil encontrar una transformación que dé un valor entero positivo en un rango de valores limitado tal que dos claves diferentes den siempre valores distintos.
- Producen huecos , zonas vacías del rango de salida, no asignadas por el algoritmo, que generan huecos en el fichero de datos.
- Para gestionar colisiones y huecos tenemos que combinar acceso directo y listas de colisión, que mantienen los registros con claves que producen colisión en dichas lista. Si estas listas crecen el acceso directo no resulta adecuado pues hay que mantener las listas y la zona de desbordamiento es casi como el fichero original.

Hashing básico Aparece para solucionar el problema del acceso directo. Ya que los valores de las claves no estaban uniformemente distribuidos en un intervalo, sino que se acumulan en una parte de él, lo que se hace es asignar más espacio a esa parte.

Así, se divide el fichero en *buckets* (cubos) y el algoritmo asignará cubos, no direcciones concretos. En cada bucket habrá más de un registro y ciertos rangos de valores tendrán asignados más buckets que otros. Complementamos esto con el uso de cubos de desbordamiento.

Tendremos como parámetros por tanto:

- Número de cubos
- Tamaño de los cubos
- La transformada clave/dirección, que tiene en cuenta la distribución de la clave para que los cubos queden equitativamente rellenos.

Para *insertar* un registro, transformamos la clave, localizamos su cubo, se inserta si hay sitio y si no se sitúa en el cubo de desbordamiento conectando el cubo a donde realmente le corresponde el registro.

Para *buscar* un registro, transformamos la clave, localizamos su cubo y dentro del cubo buscamos secuencialmente y luego en los cubos de desbordamiento.

Hashing dinámico El hashing básico tiene el problema de que hay que conocer la distribución previa de las claves para asignar los buckets, de lo contrario sigue habiendo huecos y colisiones. Además, al aumentar el número de registro,s hay más registros en las páginas de desbordamiento y a veces hay que reorganizar los datos.

El Hashing dinámico trabaja partiendo de una configuración uniforme y de pocos cubos y generando los restantes cuando los necesite, asignando a los rangos conforme la afluencia de registros lo pide.

£Cómo se hace?

El valor transformado del campo clave da una entrada de una tabla índice que está en memoria. Allí está la dirección del cubo donde están los registros que tienen asociado este valor transformado (puede que varias entradas de la tabla lleven al mismo cubo). Cuando se insertan más registros, se generan nuevos cubos y cambian las salidas de la tabla índice.

Partimos de:

- k una clave física para direccionar.
- $k' = h(k)$ un entero en un intervalo.
- n un número de bits que tiene k' en binario.
- d, los primeros d dígitos de k', que seleccionan el cubo donde está el registro(pseudoclave).
- $b < d <= n$, pues inicialmente el archivo tiene 2^b cubos distintos y como máximo tendrá 2^d.

Entonces, si tenemos una tabla índice con 2^d filas, en la primera columna se sitúan las posibles soluciones de d dígitos binarios (d es llamada la profundidad global de la tabla). Entonces, las entradas cuyos b primeros dígitos son iguales apuntan al mismo cubo. Todos los cubos suelen tener profundidad local igual a b. Por último, al llenar un cubo se divide en dos, poniendo en uno los registros con el dígito $b + 1$ de k' a 0 y en otro los que tienen el dígito $b + 1$ igual a 1. Aumenta entonces la profundidad local en uno.

De este modo, solventamos los problemas del acceso directo, aunque tenemos inconvenientes pues tenemos que usar una tabla índice adicional(y por tanto acceder más a disco si no cabe en memoria) , y el tamaño de la tabla depende de d (primeros dígitos de $k' = h(k)$.