BASES DE DATOS

Tema 1. Sistemas de bases de datos

Objetivos:

- Conocer los objetivos básicos, funciones, modelos, componentes, aplicaciones e impacto social de los sistemas de bases de datos.
- Identificar y comparar las características que aportan los sistemas de bases de datos frente a los tradicionales de procesamiento de ficheros.
- Distinguir los modelos de datos según los conceptos que ofrecen para describir la estructura de la base de datos: modelos conceptuales, de representación y físicos.
- Comprender el concepto de independencia de datos y apreciar su importancia en un sistema de bases de datos.
- Identificar los componentes y funciones principales de un sistema gestor de bases de datos.
- (opcional) Presentar la evolución histórica de las bases de datos desde su origen hasta nuestros días.

Esquema de Contenidos:

1.1 BASES DE DATOS Y SUS USUARIOS.

1.1.1 Definición de *base de datos*, *sistema de gestión de bases de datos* y *sistema de bases de datos*

En la actualidad, las bases de datos tienen una importancia decisiva en la práctica totalidad de las áreas de aplicación de la informática, como la ingeniería, la medicina, la educación, la biblioteconomía, los negocios, etc. Esto ha fomentado el desarrollo de una gran cantidad de conceptos y técnicas para la gestión eficiente de los datos.

Una primera definición de base de datos podría ser la siguiente:

"una base de datos es un conjunto de datos relacionados entre sí"

Un ejemplo sería el conjunto de nombres y números de teléfono de nuestros amigos, que tenemos registrados en una agenda.

Pero esta definición resulta demasiado general, puesto que una **base de datos** (BD) tiene las siguientes **propiedades implícitas**:

- Representa algún aspecto del mundo real, llamado **minimundo** o **universo de discurso** (UdD) del cual provienen los datos. Los cambios en el minimundo se reflejan en la base de datos.
- Es un conjunto de datos lógicamente coherente, con significado implícito. Un montón de datos sin relación entre sí, agrupados de forma aleatoria, no se considera una base de datos.
- Toda base de datos se diseña, se crea y se carga con datos, con un objetivo determinado, y está dirigida a un grupo de usuarios, interesados en el contenido y en el uso de la base de datos.

Las bases de datos pueden tener cualquier tamaño y complejidad. Cuando la cantidad de información es grande y las relaciones entre los diferentes datos son muchas, es necesario organizar y controlar toda esta información almacenada, para que los usuarios puedan buscar, obtener y actualizar los datos cuando les sea necesario.

Una base de datos puede ser creada y mantenida de forma manual (como el catálogo de fichas de una biblioteca), o bien estar informatizada. En este último caso, la creación y mantenimiento de la base de datos puede realizarse mediante un conjunto de programas de aplicación diseñados específicamente para dichas tareas, o bien mediante un sistema de gestión de bases de datos.

Un **sistema de gestión de bases de datos** (SGBD, o en inglés *database management system DBMS*) es un conjunto de programas dedicado a servir de interfaz entre las bases de datos y las aplicaciones (programas) que las utilizan, y por tanto permite a los usuarios crear y mantener bases de datos. Es un sistema software de propósito general, que facilita el proceso de **definir, construir** y **manipular** bases de datos para diversas aplicaciones.

Definir una base de datos consiste en especificar los *tipos* de los datos, las *estructuras* de los datos y las *restricciones* de los datos. **Construir** una BD es el proceso de almacenar los datos en algún medio de almacenamiento controlado por el SGBD. **Manipular** la BD es a) *consultar* los datos para obtener cierta información, b) *actualizar* la base de datos (*modificar* o *eliminar* datos, o *introducir nuevos*) para reflejar los cambios ocurridos en el minimundo, o c) *generar informes* a partir de los datos almacenados.

El objetivo principal de un SGBD es proporcionar un entorno a la vez práctico y eficiente a la hora de almacenar y recuperar la información de la base de datos.

No es obligatorio utilizar software de SGBD de propósito general para implementar una base de datos informatizada. Sería posible construir un conjunto de programas propio para crear y mantener la base de datos, es decir, crear software de SGBD de *propósito específico*.

Al conjunto formado por la base de datos y el software (tanto del SGBD como el de los programas de aplicación) lo llamaremos **sistema de bases de datos** (**SBD**). Véase la figura 1.

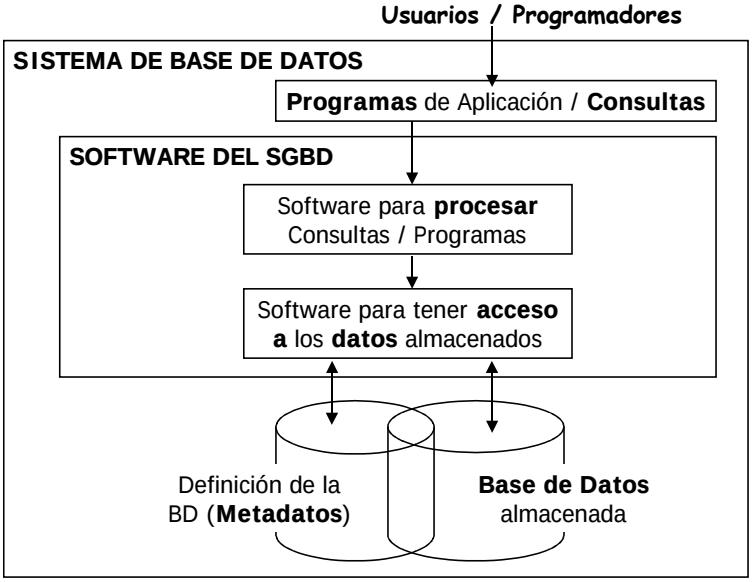

figura 1. Entorno simplificado de un sistema de bases de datos (extraído de [EN 2002])

1.1.2 Características del enfoque de bases de datos

Antes de la llegada de los SGBD, las empresas almacenaban su información empleando el enfoque clásico de *procesamiento de ficheros*, en el cual la definición, implementación y manejo de los datos (de los ficheros) necesarios para una aplicación específica se realiza como parte de la programación de la aplicación.

Las características que distinguen el enfoque de bases de datos del clásico de procesamiento de ficheros son varias:

- ✓ **Naturaleza autodescriptiva de los sistemas de bases de datos**

En los sistemas clásicos de procesamiento de ficheros, la definición de los datos es parte del código de los programas de aplicación (un ejemplo es un programa escrito en lenguaje *Pascal* que contenga declaraciones de estructuras de datos). De este modo, se puede decir que los ficheros están creados para una aplicación específica.

En cambio, en un sistema de bases de datos el SGBD no está escrito para una determinada aplicación, sino para *cualquier* aplicación de bases de datos. Por ello, el sistema no sólo almacena **datos**, sino que también guarda la **descripción** completa de los datos. Esta descripción es metainformación[1], ya que consiste en información sobre la *estructura* de cada fichero, el *tipo y formato de almacenamiento* de cada elemento y las *restricciones* que se aplican a los datos. La metainformación se guarda, concretamente, en el **catálogo del sistema**. El software del SGBD accede al catálogo para conocer la estructura de los ficheros de cada BD (la de una universidad, o la de un banco) y así *saber* cómo acceder a ellos. El catálogo también es consultado a veces por usuarios que necesitan información sobre la estructura de la BD.

- ✓ **Separación entre los programas y los datos**

En los sistemas tradicionales de procesamiento de ficheros, como ya hemos indicado, la estructura de los ficheros de datos está definida dentro de los programas. Así, un cambio en la estructura de un fichero puede implicar la modificación de todos los programas que acceden al mismo.

En cambio, los programas de acceso del SGBD se escriben para que sean independientes de cómo y dónde estén almacenados los datos. La estructura de los ficheros se guarda en el catálogo del sistema, separada de los programas. Esta propiedad es la *independencia entre programas y datos*.

Por ejemplo, en un sistema de procesamiento de ficheros es posible escribir un programa que permita el acceso a un fichero ACTOR cuya longitud de registro es de 62 caracteres. Si se añade otro campo a los registros de ACTOR, por ejemplo el *lugar de nacimiento*, ese programa no podrá seguir funcionando: habrá que modificarlo. Sin embargo, en un entorno de SGBD, basta modificar la descripción de los registros de ACTOR almacenada en el catálogo, y no es necesario cambiar un programa, por ejemplo, que muestre todos los campos de todos los registros de ACTOR. La próxima vez que el software del SGBD consulte el catálogo, tendrá acceso a la nueva estructura de los registros de ACTOR y la utilizará de forma adecuada.

La flexibilidad que proporciona esta independencia entre programas y datos es crucial para conseguir, sin esfuerzo excesivo, la adaptación continua del sistema de información a la evolución de las organizaciones.

La característica que hace posible la independencia entre programas y datos es la ***abstracción de los datos*** proporcionada por el SGBD. Trataremos este concepto ampliamente en el apartado 1.2 de este tema.

- ✓ **Datos compartidos y procesamiento multiusuario**

Una base de datos contiene datos que se definen una sola vez, y dan servicio a diversas aplicaciones, por lo que son utilizados al mismo tiempo por muchos usuarios. La base de datos no pertenece sólo a un departamento o sección, sino que se comparte por toda la organización o empresa.

[1] La metainformación (metadatos) es información (datos) acerca de la información (datos).

Es imprescindible que el SGBD incluya software de control de concurrencia para asegurar que, cuando varios usuarios intenten actualizar los mismos datos, lo hagan de manera controlada, de forma que el resultado final sea correcto.

Un ejemplo sería el caso de varios encargados de realizar reservas de asientos numerados en una sala de cine: el SGBD debe asegurar que sólo un empleado tenga acceso a un asiento específico en un momento dado, para asignarlo a un cliente, y que en cuanto un empleado reserve un asiento, los demás lo vean inmediatamente. Cada operación de reserva sería una *transacción*. Una función fundamental del SGBD es asegurar que las transacciones concurrentes se realizan de manera correcta, sin interferencias entre ellas.

- ✓ **Soporte de múltiples vistas de los datos**

Como ya hemos indicado, un sistema de BD suele tener muchos usuarios. Algunos de ellos no deberían poder acceder a toda la base de datos por cuestiones de seguridad, o simplemente no necesitan acceder más que a una parte de los datos.

Por ejemplo, en un *sistema de gestión de una productora de películas de cine*, el personal de nóminas necesita ver sólo la parte de la base de datos que contiene información acerca de los empleados de la productora, y no necesita saber nada acerca de la recaudación de las películas proyectadas en diferentes salas de cine.

Por tanto, cada usuario (o grupo de usuarios) puede necesitar una *vista* o perspectiva diferente de la BD. Una vista puede ser un subconjunto de la base de datos, y puede contener *datos virtuales* (no almacenados, sino que se derivan o calculan a partir de otros datos).

Los usuarios normalmente no necesitan saber (de hecho, no lo saben) si ven y utilizan todos o sólo parte de los datos, y tampoco si son datos derivados o no.

Un SGBD multiusuario debe proporcionar mecanismos para definir muchas vistas. Por ejemplo, es posible que a un usuario de la base de datos de la productora sólo le interesen los datos relacionados con el presupuesto y gastos de los rodajes de las películas dirigidas por cada director, mostrados por la vista de la figura 2. Por otro lado, si otro usuario está dedicado a realizar estudios de recaudación en taquilla de las películas en cartelera, tan sólo necesitaría ver los datos mostrados por la vista de la figura 3.

GASTOS	Director	Nacionalidad Director	Películas Rodadas			
			Título de Película	Presupuesto	Año de Rodaje	Gasto Realizado
	Vicente Aranda	Española	Celos	*XXXX*	1999	*XXXX*
	Fernando León	Española	Familia	*XXXX*	1996	*XXXX*
			Barrio	*XXXX*	1998	*XXXX*

figura 2. Una vista de la base de datos. [*XXXX* corresponde a datos reales, en este caso una cantidad de dinero]

TAQUILLA	Película	Películas en Cartelera			
		Ciudad	Nº de Salas	Fecha de Estreno	Recaudación Actual
	Barrio	Murcia	3	*d-m-1998*	*XXXX*
	Celos	Yecla	1	*d-m-1999*	*XXXX*
		Archena	1	*d-m-1999*	*XXXX*
		Lorca	2	*d-m-1999*	*XXXX*

figura 3. Otra vista de la base de datos. [*d-m-* corresponde a datos reales, en este caso, el día y mes del estreno]

Las características anteriores son muy importantes para distinguir entre un SGBD y el software tradicional de procesamiento de ficheros. Más adelante, en el apartado 1.1.4, veremos otras funciones que caracterizan a los SGBDs. Antes de ello, estudiaremos una clasificación de los diferentes tipos de personas que trabajan en un entorno de bases de datos.

1.1.3 Actores en un sistema de BD

Son muchas las personas que participan en el **diseño**, **mantenimiento** y **uso** de una base de datos, sobre todo si ésta es grande. En primer lugar veremos los que emplean de forma cotidiana la base de datos, es decir, la información que ésta contiene: administradores, diseñadores, usuarios finales, ingenieros software. Posteriormente las personas ocupadas de mantener el entorno del sistema de bases de datos: diseñadores e implementadores del SGBD, desarrolladores de herramientas, y operadores y personal de mantenimiento.

Administrador de la Base de Datos, ABD (en inglés DBA: *Data Base Administrator*)

Una de las principales razones para usar SGBD es tener un control centralizado de los datos, así como de los programas que acceden a dichos datos. La persona que tiene ese control central sobre el sistema es el *administrador de la base de datos*. Es el responsable de administrar los recursos del SBD, es decir la propia base de datos (recurso primario), el SGBD y el software relacionado con éste (recursos secundarios).

Las funciones del ABD incluyen las siguientes:

- Definir y modificar el esquema de la base de datos y las restricciones de los datos.
- Crear y modificar las estructuras de almacenamiento y definir los métodos de acceso.
- Autorizar el acceso a la BD, y coordinar y controlar tales accesos.
- Garantizar el funcionamiento correcto del sistema y prestar servicio técnico: se ocupa de los problemas de violación de la seguridad del sistema de BD, o de respuesta lenta del sistema.
- Definir y poner en práctica planes adecuados de copias de seguridad del contenido de la BD.
- Adquirir los recursos necesarios de software y hardware.

Diseñadores de la Base de Datos

Identifican los datos que se almacenarán y eligen las estructuras adecuadas, para representar y almacenar dichos datos. Estas tareas se realizan antes de que se implemente la base de datos.

Los diseñadores interactúan con los grupos de futuros usuarios de una BD, para comprender sus necesidades, desarrollan una *vista de la base de datos* que satisfaga los requisitos (de datos y de procesamiento) de cada grupo, e integran todas las vistas creadas para obtener un *diseño final de la base de datos* que cumpla con las necesidades de todos los grupos.

Usuarios finales

Son los que necesitan tener acceso a la base de datos para consultar sus datos o modificarlos.

- *Usuarios ocasionales*. Acceden a la BD esporádicamente, posiblemente para obtener información diferente cada vez. Usan un lenguaje de consulta de BD para especificar sus solicitudes. Un usuario de este tipo suele pertenecer a la plantilla de la organización en la que se ha implantado el SBD, al que se le dan algunas nociones de un lenguaje de consultas; no tiene por qué conocer con qué recursos cuenta el SGBD.
- *Usuarios paramétricos (usuarios normales)*. En su trabajo realizan consultas y actualizaciones constantes de la BD, utilizando operaciones que se han programado y probado (conocidas como *transacciones programadas*). Usuarios de este tipo serían el personal de caja de un banco, encargados de reservas de hotel o empleados en empresas de reparto a domicilio. No necesitan saber con qué recursos cuenta el SGBD, sino las operaciones diseñadas para que ellos las usen.
- *Usuarios avanzados (usuarios sofisticados)*. Ingenieros, científicos, analistas de empresas. Conocen los recursos del SGBD para satisfacer sus complejas necesidades. Hacen consultas a la BD desde una terminal utilizando un lenguaje de consulta (sin programas escritos) para explorar los datos en la base de datos.
- *Usuarios autónomos*. Usan BD personales a través de una aplicación comercial o paquete software específico. Un usuario autónomo sería el de una aplicación de contabilidad que gestiona los datos contables de su propio negocio.

Analistas de sistemas y programadores de aplicaciones (ingenieros de software)

Son profesionales informáticos que deben conocer perfectamente las capacidades y recursos del SGBD. Los **analistas** determinan los requisitos de los usuarios finales (sobre todo de los paramétricos) y desarrollan especificaciones de conjuntos de operaciones (transacciones programadas) que satisfagan esos requisitos. Los **programadores** implementan esas especificaciones en forma de programas de aplicación, las prueban, depuran, documentan y mantienen.

Los usuarios de los que hablaremos a continuación son las personas encargadas del diseño, creación y operación del *software y entorno del sistema*. No les suele interesar demasiado la BD en sí misma, es decir apenas utilizan el contenido de la base de datos para sus propios propósitos.

Diseñadores e implementadores del SGBD

Diseñan e implementan los módulos e interfaces del SGBD en forma de paquetes software. Un SGBD es un sistema software compuesto de diversos componentes o módulos (que veremos más adelante). Además debe disponer de interfaces que le permitan comunicarse con otros programas, como el sistema operativo o compiladores de lenguajes de programación.

Desarrolladores de herramientas

Son los encargados de diseñar e implementar herramientas, es decir paquetes software que facilitan el diseño y uso de los sistemas de bases de datos, y que permiten aumentar el rendimiento de los mismos. Suelen adquirirse por separado. Incluyen paquetes para diseñar esquemas de bases de datos, supervisar el rendimiento, proporcionar interfaces para el usuario (de lenguaje natural o gráficos), crear prototipos, realizar simulaciones y generar datos de prueba.

Operadores y personal de mantenimiento

Personal de administración del sistema, encargados del funcionamiento y mantenimiento del entorno hardware y software del sistema de bases de datos.

1.1.4 Ventajas del uso de un SGBD

Un SGBD debe ofrecer una serie de beneficiosas capacidades, que el ABD debe aprovechar para conseguir los objetivos de diseño, de administración y de uso de una gran base de datos multiusuario. Entre ellas, destacamos las siguientes:

1. Disminución y control de la redundancia de datos.

En el desarrollo de software tradicional con procesamiento de ficheros, para cada (grupo de) usuario(s) se define e implementa los ficheros necesarios para sus propias aplicaciones (programas).

Normalmente, los ficheros y los programas son creados por diferentes programadores durante un largo período de tiempo, lo cual puede provocar que los ficheros tengan diferentes formatos y los programas estén escritos en lenguajes de programación distintos.

Es más, si varios grupos de usuarios necesitan acceder a la misma información, ésta podría estar almacenada en varios sitios (ficheros) simultáneamente.

Por ejemplo, podemos considerar que un grupo de usuarios está compuesto por el *personal de contabilidad* encargado de la gestión de las nóminas de los directores cinematográficos, el cual está interesado en los datos personales y económicos de los directores; por otro lado, el *personal de publicidad* constituiría otro grupo de usuarios, interesado en los datos personales y profesionales –películas realizadas, etc.– de los mismos directores. Así, los datos personales estarían duplicados: se almacenarían tanto en un fichero para la aplicación de contabilidad y nóminas, como en un fichero para la aplicación de publicidad.

Esta situación se conoce como *redundancia de datos*.

La redundancia de datos provoca varios problemas:

- Duplicación del trabajo, pues al introducir nuevos datos en el sistema (un nuevo director) es necesario copiarlos en varios sitios (en cada fichero en el que se guarden datos de directores).
- Desperdicio del espacio de almacenamiento (mayor coste de almacenamiento).
- La obligación de controlar que, cada vez que cambie un dato, todas sus copias sean actualizadas correctamente.
  En nuestro ejemplo, si se modificara el teléfono de un director en el fichero de la aplicación de contabilidad y nóminas, también habría que hacerlo en el fichero de la aplicación de publicidad (y viceversa). En caso contrario, es decir, si las copias no se actualizan al nuevo valor, o se comete algún error al actualizarlas, se incurriría en una *inconsistencia de datos* (véase el siguiente apartado).

2. *Evitar inconsistencias en los datos*

La inconsistencia surge cuando existen varias copias del mismo dato y tras la modificación de una de ellas, las demás no son actualizadas, o sí lo son pero de forma incorrecta.

Es posible evitar la inconsistencia de dos maneras:

- Si se elimina la redundancia. Esto se consigue si se diseña la BD de forma que se integren las vistas de los diferentes grupos de usuarios y cada dato lógico se almacene en un *único* lugar.
- Si existe *redundancia controlada*. A veces conviene tener redundancia de algunos datos, para mejorar el rendimiento de las consultas. Supongamos por ejemplo que siempre que se necesita obtener los datos asociados a una película, también interesa obtener la nacionalidad del director que la ha rodado y el nombre del autor de su guión. Esto implica el acceso a tres ficheros: PELICULA, DIRECTOR y GUION. Podríamos almacenar de manera redundante la nacionalidad del director y el nombre del guionista en el fichero PELICULA (añadiendo los campos *nacioDire* y *nombreGuionista*). En ese caso, al solicitar el listado de películas antes mencionado, ya no sería necesario buscar la información en tres ficheros, sino únicamente en PELICULA.

  En casos de este tipo, el SGBD debe aplicar automáticamente al resto de copias cualquier modificación realizada sobre un dato. En nuestro ejemplo, debe verificar que el valor de *nacioDire* de todo registro almacenado en PELICULA, coincide con algún valor del campo *nacionalidad* de algún registro en el fichero DIRECTOR, y también que si se modifica el nombre de un guionista en GUION, tal cambio se realiza también en los correspondientes registros (campo *nombreGuionista*) en PELICULA, es decir, en aquellas películas cuyo guión esté escrito por el guionista modificado. Esto se denomina *propagación de actualizaciones*.

  Pocos sistemas de bases de datos comerciales actuales soportan la redundancia controlada.

3. *Mejora en la integridad de datos*

La integridad se refiere a la validez y consistencia de los datos. Mantener la integridad es asegurar que la información almacenada (y utilizada por una aplicación de bases de datos) es correcta, es decir, refleja fielmente la realidad, el minimundo.

Por tanto, se incumple la integridad de datos (se viola la integridad) cuando…

a) Existe *inconsistencia*. Esto sólo puede darse cuando existe redundancia de datos (véase apartado anterior).
b) Existe *información imposible* (40/MAYO/1972 como fecha de fin de un rodaje, una película sin director) o *información falsa* que no se ajusta a la realidad (una película con 523 actores protagonistas, cuando en la realidad son 5).

Estas situaciones se evitan si los datos cumplen las llamadas *restricciones de integridad* (RI).

Es necesario, pues, expresar la integridad en forma de restricciones o reglas que los datos no deben incumplir. Las RI más sencillas son las restricciones sobre los *tipos de datos* de los elementos de información, como por ejemplo:

- El nombre propio de un actor debe ser una cadena de hasta 25 caracteres alfabéticos.
- La duración de una película ha de ser un nº entero entre 1 y 180 (minutos).

Otras RI son las llamadas *semánticas*, pues tienen que ver con el *significado* de los datos y del mini-mundo, como por ejemplo:

- Toda película debe estar relacionada con al menos un director.
- Cada película ha de tener un título diferente del resto (título único).
- Ningún actor o actriz en una película puede percibir más dinero que el protagonista.

Los diseñadores deben identificar las restricciones de integridad durante el diseño de la BD.

Algunas RI (como las que indican cuándo una fecha es válida) pueden ser verificadas de forma automática por el SGBD; así pues, el SGBD debe permitir la definición de dichas restricciones.

Pero para otras restricciones de integridad es necesario crear programas que verifiquen su cumplimiento; en este caso, el SGBD debe permitir la creación y ejecución de tales programas.

Evitar las violaciones de reglas de integridad es crucial en los sistemas de BD multiusuario, en los que muchos usuarios acceden a la misma información: el que uno de ellos modifique un dato o introduzca información errónea, afecta al resto de usuarios del sistema. Será necesario verificar el cumplimiento de las restricciones de integridad en cada actualización (introducción, modificación o eliminación) de datos.

Sin embargo, puede darse el caso de que se introduzca información errónea o falsa (por ejemplo, escribir el nombre *Jines*, en lugar de *Ginés*) **sin** violar restricciones de integridad. En casos como este, el SGBD no es capaz de detectar el error de forma automática, pues el dato, aunque mal escrito, es del tipo de datos correcto (una cadena de menos de 25 caracteres).

4. Mejora en la seguridad

El hecho de que en un sistema de bases de datos los datos estén centralizados supone mayor peligro de accesos no autorizados que si estuvieran almacenados en un sistema de ficheros. Es imprescindible que sólo tengan acceso al SBD los usuarios acreditados. Además, es muy habitual (y conveniente) que no todos los usuarios puedan acceder a toda la información almacenada, pues existen datos confidenciales que sólo ciertas personas pueden ver o utilizar. Incluso es posible que ciertos usuarios sólo tengan permiso para obtener (consultar o ver) los datos, mientras que otros sí puedan actualizarlos además de consultarlos.

El SGBD debe disponer de un robusto *subsistema de seguridad y autorización*, mediante el cual el ABD pueda:

- Crear cuentas de usuario protegidas con contraseñas (para asegurar que sólo acceden a los datos los usuarios que tengan permiso para ello).
- Crear restricciones para cada cuenta, de forma que se controle *a)* a qué datos tiene acceso el usuario y *b)* el tipo de operaciones que puede realizar sobre esos datos (es decir, si puede verlos o modificarlos o crear nuevos o eliminarlos).

El SGBD obligará (de forma automática) el cumplimiento de estas restricciones.

Otros controles de seguridad son, por ejemplo, que sólo el ABD pueda usar el software de administración y monitorización de la BD (para crear nuevas cuentas de usuario, por ejemplo), o bien que los usuarios paramétricos sólo puedan acceder a la BD mediante los programas que se crearon para ellos.

5. Mejora en la accesibilidad de los datos

Puesto que las BD son usadas por muchos usuarios, con variados niveles de conocimientos técnicos, el SGBD debe ofrecer diferentes interfaces, para todos ellos.

- *Lenguajes de consulta* ..... usuarios ocasionales
- *Interfaces de Lenguajes de Programación* ..... programadores de aplicaciones
- *Formularios* (*forms*) y *comandos* (órdenes) ..... usuarios paramétricos
- *Interfaces controladas por menús* y *de lenguaje natural* ... usuarios autónomos

Las interfaces con formularios y las controladas por menús se denominan habitualmente *interfaces gráficas de usuario* (en inglés, *GUI: graphic user interface*), las cuales pueden especificarse empleando diversos lenguajes y entornos especializados.

Además, cada vez es más común ofrecer acceso a bases de datos a través de interfaces Web.

6. Representación de relaciones complejas entre los datos

Los datos en la BD están relacionados entre sí de diversas formas. Por ejemplo, el registro de *Ernesto Alterio* en el fichero ACTOR, puede estar relacionado con varios registros del fichero PELICULA (*Insomnio*, *Cuarteto de La Habana*, *Los lobos de Washington, El otro lado de la cama*). Por otro lado, cada registro de PELICULA (*Celos*) se relaciona con un registro de DIRECTOR (*Vicente Aranda*) y con varios de ACTOR (*Aitana Sánchez-Gijón*, *Daniel Giménez Cacho*, *María Botto*, ...), uno por cada actor participante en la película en cuestión.

El SGBD debe permitir la representación de estas relaciones (o vínculos) entre los datos, así como la obtención y actualización (rápida y eficiente) de datos que estén relacionados (obtener el título de todas las películas, y los nombres de sus directores, en las que actúe *Javier Bardem*).

7. Mejora en los servicios de respaldo y recuperación

Todo SGBD debe proporcionar una manera eficiente de realizar copias de seguridad de la información que contiene, así como de restaurar, a partir de dichas copias, los datos que se hayan podido perder debido a fallos del hardware o software. De esto se encargará el *subsistema de respaldo y recuperación* del SGBD. Si el fallo ocurre mientras estaba en marcha un programa que actualizaba gran cantidad de datos, entonces una vez el sistema ha sido reiniciado tras el fallo, el subsistema de recuperación debe asegurar a) que la base de datos se restaura al estado en que estaba justo antes de comenzar el programa, o bien b) que el programa continúa su ejecución por el punto en donde la dejó cuando se produjo el fallo, y finaliza su trabajo correctamente.

1.1.5 Otras ventajas del enfoque de bases de datos

Además de los aspectos que hemos visto en el apartado anterior, existen otras implicaciones del uso del enfoque de bases de datos que pueden resultar beneficiosas.

1. Datos compartidos actualizados

Los datos están disponibles para todos los usuarios y cuando alguno actualiza la información, los demás ven los cambios inmediatamente. Esto es posible gracias a los subsistemas de control de concurrencia y recuperación del SGBD.

2. Mejora en el mantenimiento y la flexibilidad

Cuando los requisitos del sistema varían o surgen nuevas necesidades de datos, normalmente es necesario modificar la estructura de la base de datos, como ocurre cuando es necesario añadir un nuevo fichero (por ejemplo, para almacenar animales actores) o ampliar un fichero ya existente (un nuevo campo en el fichero PELICULA para indicar si ha salido o no a la venta en vídeo). Algunos SGBD permiten realizar estos cambios en la estructura de la BD sin afectar ni a los datos ya almacenados ni a los programas de aplicación ya existentes.

3. Creación rápida de nuevas aplicaciones

Diseñar e implementar una BD desde cero resulta bastante más costoso que crear una sola aplicación de procesamiento tradicional de ficheros. Sin embargo, una vez que la BD está creada y en funcionamiento, crear una aplicación nueva (como la obtención de ciertos datos para imprimir un informe nuevo) necesita de mucho menos tiempo.

4. Cumplimiento de normas de empresa

En un entorno centralizado de base de datos, el administrador de los datos puede definir e imponer normas, reglas o políticas de empresa con más facilidad que en un entorno en el que cada grupo de usuarios (departamento) tenga el control de sus propios ficheros y programas.

El ABD puede establecer el cumplimiento de estándares de empresa, nacionales e internacionales por parte de los usuarios de la BD de una gran organización, de forma que se potencie el intercambio de información y la cooperación entre departamentos y/o proyectos. Entre ellos pueden incluirse estándares sobre nombres y formatos de los datos, sobre la estructura de la documentación y de los formularios de pantalla e informes, sobre los procedimientos de acceso y de actualización de los datos, etc.

1.1.6 Inconvenientes de los SGBD

Ya hemos visto el conjunto de ventajas que conlleva el uso de un SGBD. Sin embargo, existen situaciones en las que emplear un SBGD puede generar costes adicionales innecesarios, que se evitarían con el procesamiento de ficheros tradicionales. Tales costes pueden surgir de lo siguiente:

- **Complejidad del SGBD.** Los SGBD son sistemas software muy complejos, pues ofrecen amplia generalidad para definir y procesar datos, además de las funciones de seguridad, integridad, concurrencia, recuperación, etc. Es necesario comprender bien estas funcionalidades para poder aprovecharlas adecuadamente.
- **Alto consumo de recursos.** El SGBD requiere gran cantidad de espacio en disco y memoria para funcionar eficientemente.
- **Coste económico.** Montar un sistema de bases de datos supone una gran **inversión inicial** en equipo, software y formación del personal. Además, suele ser necesario el pago de una cuota anual de mantenimiento. Por otro lado, si la organización desea migrar su sistema de procesamiento de ficheros actual a un sistema de bases de datos, hay que sumar los costes de conversión correspondientes.
- **Disminución de la eficiencia de algunas aplicaciones.** En un sistema de procesamiento de ficheros, los ficheros están creados para una aplicación específica, por lo que ésta suele tener muy buenas prestaciones. Sin embargo, un SGBD da servicio a diversas aplicaciones, por lo que algunas de éstas pueden no ser tan eficientes.

Así, puede ser más conveniente usar un sistema de procesamiento de ficheros en las siguientes situaciones:

- La estructura de los datos y las aplicaciones son simples, están bien definidas y no se espera que cambien,
- Algunos programas tienen requisitos estrictos de tiempo real que no podrían cumplirse por el coste extra del SGBD,
- No se necesita el acceso multiusuario a los datos.

1.2 CONCEPTOS Y ARQUITECTURA DEL SISTEMA DE BASES DE DATOS.

1.2.1 Modelos de datos, esquemas e instancias

Una característica fundamental (y un objetivo importante) del enfoque de bases de datos es proporcionar al usuario una visión abstracta de los datos, es decir, ocultarle detalles de almacenamiento y mantenimiento de los datos, que no necesita conocer.

Para conseguir la abstracción de datos se utilizan los modelos de datos. Un **modelo de datos** es un conjunto de conceptos (herramientas conceptuales) que sirve para describir la estructura de una BD, es decir los *tipos de datos*, las *relaciones* entre ellos y las *restricciones* que deben cumplir. La mayoría de los modelos de datos contienen además un conjunto de *operaciones básicas* para especificar lecturas y modificaciones de la base de datos.

Tipos de modelos de datos

Una forma de clasificar los modelos de datos es la basada en el nivel de abstracción de los conceptos que ofrecen para describir la estructura de la BD.

- ✓ Los ***modelos** de alto nivel o **conceptuales*** constan de conceptos muy cercanos al modo en que el usuario percibe la realidad, y describen ésta como un conjunto de entidades y las relaciones entre ellas. Usan conceptos como:

  Entidad: representa una cosa, objeto o concepto del mundo real (un director, una película) almacenado en la base de datos.

  Atributo: representa una propiedad interesante de alguna entidad (nombre del director, título de la película).

  *Relación (*o *Interrelación)*: describe una asociación entre varias entidades (vínculo que existe entre una película y el director que la ha realizado).

  · Un modelo de alto nivel es el *Modelo Entidad-Relación* (en inglés, *ER: Entity/Relationship Model*).

  · Los *modelos de datos orientados a objetos* también suelen usarse como modelos conceptuales de alto nivel, sobre todo en el área de la Ingeniería del Software. Este tipo de modelos caen fuera del ámbito de esta asignatura.

- ✓ Los ***modelos** de representación* o ***lógicos***, proporcionan conceptos que pueden ser entendidos por los usuarios finales, aunque no están muy alejados de la forma en que los datos se organizan dentro del ordenador. Ocultan algunos detalles de almacenamiento, pero sus conceptos pueden implementarse directamente en un sistema informático. Son los más utilizados en los SGBD comerciales actuales (*Oracle*, por ejemplo).

  · Los más comunes son el *Modelo Relacional* (el más utilizado actualmente y que estudiaremos con detalle en esta asignatura), el *Modelo de Red* y el *Modelo Jerárquico* (muy utilizados en el pasado). Los tres son *modelos de datos basados en registros*, porque usan estructuras de registros para representar los datos.

  · Los *modelos de datos orientados a objetos* también pueden ser considerados modelos de representación, aunque de un nivel próximo a los modelos conceptuales.

- ✓ Los ***modelos** de bajo nivel o **físicos*** disponen de conceptos que describen los detalles de almacenamiento de los datos en el ordenador. Estos conceptos no están dirigidos a los usuarios finales, sino a especialistas en informática. Describen cómo se almacenan los datos, indicando el formato y el ordenamiento de los registros y los caminos de acceso. Un *camino de acceso* es una estructura que permite realizar búsquedas de datos de forma eficiente (por ejemplo, un *fichero índice*). Más adelante, estudiaremos técnicas de almacenamiento y estructuras de acceso.

Esquemas, instancias y estado de la base de datos

En todo modelo de datos es preciso diferenciar entre *descripción de base de datos* y la *base de datos* en sí. La descripción es el **esquema** de la base de datos (los metadatos), el cual se especifica en el diseño y rara vez es modificado.

El **diagrama del esquema** es la representación de un esquema utilizando la notación de algún modelo de datos. Un diagrama (como el de la figura 4) representa la estructura de los **tipos de registro** DIRECTOR y PELICULA, pero no su contenido (es decir, los datos sobre directores y películas reales, almacenados en la BD). Cada tipo de registro es un **objeto** o **elemento del esquema**.

DIRECTOR

nombreDirector	nacionalidad	fechaNacimiento	numPeliculas

PELICULA

títuloPelicula	director	género	guión	añoRodaje	nacionalidad	duración

figura 4. Una posible representación (diagrama del esquema) de los esquemas de dos tipos de registro.

Otra posible representación de los esquemas de la figura 4 sería la siguiente:

Director (nombreDirector, nacionalidad, fechaNacimiento, numPeliculas)

Pelicula (títuloPelicula, director, género, guión, añoRodaje, nacionalidad, duración)

Un diagrama de esquema sólo muestra *parte* del esquema. El de nuestro ejemplo muestra los nombres de los elementos del esquema (tipos de registro) y de las características o atributos de cada uno. Sin embargo, no describe restricciones tales como los tipos de datos de cada atributo o la relación que existe entre cada película y el director que la ha realizado.

Los datos reales cambian muy a menudo, por ejemplo cada vez que introducimos un nuevo director, se elimina una película, o aumenta el número de películas rodadas por un director. El **estado de la base de datos**[2] es el conjunto de datos que contiene en un momento determinado, es decir el conjunto de **instancias** (ejemplares, ocurrencias) de los elementos que contiene.

En un estado concreto de la BD, cada elemento del esquema tiene su propio conjunto actual de instancias. Por ejemplo, en cierto instante, el elemento PELICULA podría contener estas dos instancias, correspondientes a las películas tituladas *Torrente* y *The Matrix*:

Torrente	Santiago Segura	Comedia Casposa	Santiago Segura	1997	España	110
The Matrix	Andy Wachowski	Ciencia-ficción	Andy Wachowski	1999	EEUU	138

Mientras que en un instante posterior puede contener tres instancias más: *Episodio I: la amenaza fantasma*, *Celos* y *Locos en Alabama*, de forma que el conjunto actual de instancias sería este:

Torrente	Santiago Segura	Comedia Casposa	Santiago Segura	1997	España	110
The Matrix	Andy Wachowski	Ciencia-ficción	Andy Wachowski	1999	EEUU	138
Episodio I: la amenaza fantasma	**George Lucas**	**Ciencia-ficción**	**George Lucas**	**1999**	**EEUU**	**133**
Celos	**Vicente Aranda**	**Drama**	**Álvaro del Amo**	**1999**	**España**	**122**
Locos en Alabama	**Antonio Banderas**	**Comedia**	**Mark Childress**	**1999**	**España**	**108**

Más tarde, podemos eliminar la instancia de *Torrente* y cambiar el valor (erróneo) del campo *nacionalidad* de la película *Locos en Alabama* para introducir su valor correcto, de modo que el conjunto actual de instancias de PELICULA queda como se muestra a continuación:

The Matrix	Andy Wachowski	Ciencia-ficción	Andy Wachowski	1999	EEUU	138
Episodio I: la amenaza fantasma	George Lucas	Ciencia-ficción	George Lucas	1999	EEUU	133
Celos	Vicente Aranda	Drama	Álvaro del Amo	1999	España	122
Locos en Alabama	Antonio Banderas	Comedia	Mark Childress	1999	**EEUU**	108

Los tres estados anteriores corresponden a un *mismo esquema* de la base de datos en la que está el elemento PELICULA.

[2] También puede denominarse "*ejemplar de la base de datos*".

Es muy importante **distinguir entre esquema y estado de la bases de datos**. A veces, al esquema se le denomina ***intensión*** y al estado, ***extensión***. Cuando se define una base de datos sólo se especifica su esquema en el SGBD, el cual lo almacena en su catálogo. En ese momento, la base de datos está en el *estado vacío*: no contiene información. Cuando se introducen (se cargan) los datos iniciales (aquellos necesarios para que comiencen a trabajar las aplicaciones), la BD pasa al *estado inicial*. Cada vez que insertamos o eliminamos una instancia, o modificamos el valor de algún elemento de información, la base de datos cambia a un nuevo estado, que se convierte en su *estado actual*.

El SGBD asegura que todo estado de BD es válido o consistente, es decir que satisface la estructura y restricciones especificadas en el esquema. Por esto es muy importante diseñar correctamente el esquema de base de datos.

1.2.2 Arquitectura de tres niveles de un SGBD

Para que el SGBD sea útil, debe recuperar la información de forma eficiente. Esta necesidad ha llevado al diseño de estructuras de datos complejas para la representación de los datos en la base de datos. Como muchos usuarios de sistemas de BD no están familiarizados con los ordenadores, se oculta la complejidad a través de varios niveles de abstracción, y así se simplifica la interacción de los usuarios con el sistema.

La arquitectura de tres niveles para sistemas de bases de datos ayuda a la consecución de dos de las características del enfoque de bases de datos: la separación entre los programas y los datos y el soporte de múltiples vistas de usuario.

Arquitectura ANSI/X3/SPARC[3]

El objetivo de esta arquitectura es separar las aplicaciones del usuario de la base de datos física. Los **esquemas** pueden ser definidos en **tres niveles**:

1. **Nivel Interno**

   Es el nivel más bajo de abstracción, un paso por encima del nivel físico. Tiene un **esquema interno** (EI) que describe *cómo* se almacenan realmente los datos, utilizando un **modelo físico de datos**, y muestra detalles de la organización física de los ficheros (estructuras físicas de almacenamiento, orden de secuencia de los registros físicos, tamaño de página, de bloque, etc.) y caminos de acceso (tipos de índice, etc.).

2. **Nivel Conceptual**

   Tiene un **esquema conceptual** (EC), que describe la estructura de *toda* la BD para el conjunto de usuarios. El EC oculta los detalles físicos y describe *qué* datos se almacenan en la base de datos y *qué* vínculos existen entre ellos, es decir, entidades, tipos de datos, relaciones, operaciones de los usuarios y restricciones (seguridad, integridad).

   En este nivel, para describir el esquema conceptual puede utilizarse un **modelo de datos conceptual** o bien un **modelo de representación o lógico**[4].

3. **Nivel Externo o de Vistas**

   Es el nivel más alto de abstracción. A muchos usuarios no les preocupa toda la información almacenada en la base de datos, sino que necesitan acceder sólo a una porción. Para simplificar su interacción con el sistema, se define este nivel de abstracción de vistas, que está compuesto de varios **esquemas externos** (EE) o **vistas** de usuario. Cada EE describe la parte de la BD que interesa a un grupo de usuarios determinado (la porción de la realidad que perciben), ocultándoles el resto de la BD. Para dicho grupo de usuarios, su vista *es* la base de datos.

   En este nivel puede usarse un **modelo de datos conceptual** o un **modelo de representación** para describir cada esquema externo.

[3] ANSI/X3/SPARC es un grupo de estudio sobre sistemas de administración de bases de datos, del *Standard Planning And Requirements Committee* (SPARC) del ANSI (*American National Standards Institute*), dentro del comité X3 que se encarga de informática y ordenadores.
[4] De hecho, en algunos libros como [SKS 1998], tanto este nivel como el esquema que lo describe se denominan *lógicos* en lugar de *conceptuales*.

La mayor parte de los SGBD comerciales actuales no distinguen del todo los tres niveles. Algunos incluyen detalles del nivel físico en el esquema conceptual. Los esquemas externos (vistas) suelen especificarse mediante el mismo modelo de datos que se usa para describir el esquema conceptual (es decir, un modelo de representación).

Es importante señalar que **los tres esquemas son descripciones de datos**, puesto que los *datos reales* están (sólo) en el nivel físico, almacenados en un dispositivo como puede ser un disco.

Correspondencia entre esquemas (transformación o mapping)

En un SGBD con esta arquitectura de tres esquemas, cada grupo de usuarios trabaja únicamente con su esquema externo, así que el SGBD debe traducir cada petición (consulta) expresada en términos de un esquema externo a una petición en un esquema conceptual, y luego a una petición expresada en el esquema interno, la cual se procesará sobre la BD física.

Si la petición es la obtención de ciertos datos, será preciso que el SGBD modifique el formato de los datos extraídos de la base de datos física, para que coincida con la vista externa del usuario.

Este proceso de transformar peticiones y resultados de un nivel a otro se denomina *correspondencia* o *transformación*.

Independencia de datos

La arquitectura de tres niveles es muy útil para explicar el concepto de *independencia de datos.* La independencia de datos es la **capacidad para modificar el esquema en un nivel del SBD sin tener que modificar el esquema del nivel inmediato superior**. Existen dos tipos de independencia de datos:

1. *Independencia **lógica** de datos*.

   Capacidad de modificar el esquema conceptual (su estructura) sin alterar los esquemas externos (lo que ven los usuarios), ni (el código de) los programas de aplicación.

   Las modificaciones en el nivel conceptual son necesarias siempre que la estructura lógica de la base de datos sea alterada. Por ejemplo, puede modificarse el esquema conceptual para **ampliar** la base de datos al añadir un nuevo tipo de registro ANIMAL_ACTOR, o al añadir un campo, como *nacioDire* (nacionalidad del director) o *enVideo* para PELICULA.

PELICULA

tituloPelicula	director	género	guión	añoRodaje	nacionalidad	duración	**nacioDire**	**enVideo**

figura 5. Modificación del esquema de PELICULA: adición de dos nuevos campos de información.

   También se puede **reducir** la base de datos, por ejemplo eliminando un tipo de registro, o un campo de información. En este caso, la modificación sólo debe afectar a los esquemas externos que se refieran a los elementos que desaparecen, es decir, *esos* sí deberán ser modificados, claro está. Pero los esquemas externos que sólo se refieran a los datos restantes no se verán afectados. Por ejemplo, el esquema externo de la figura 2 no debería alterarse si se modificara el esquema PELICULA de la figura 4 para convertirlo en el esquema de la figura 5.

   Si en el SGBD se cuenta con independencia lógica de datos, sólo será necesario modificar la *definición* de la vista y las correspondencias, o sea, indicar al SGBD de dónde debe obtener los datos que debe mostrar en la vista *Gastos*: antes tomaba la nacionalidad de cada director a partir del contenido de DIRECTOR, y ahora debe obtenerla directamente de PELICULA (del campo *nacioDire*).

   Las aplicaciones que utilizan los elementos del EE funcionarán igual que antes de modificar la estructura del esquema conceptual (por ejemplo, una consulta como "*listar todas las películas de Alex de la Iglesia rodadas antes de 1997*"). Esto es porque las vistas siguen mostrando la misma in-

formación, y las aplicaciones las usan directamente, sin importar de dónde extraen las vistas los datos que muestran.

Además, también pueden modificarse las restricciones en el esquema conceptual, sin afectar a los esquemas externos ni a los programas de aplicación.

La independencia de datos lógica es más difícil de conseguir que la independencia de datos física (que veremos a continuación), ya que los programas de aplicación suelen ser fuertemente dependientes de la estructura lógica de los datos a los que acceden.

2. *Independencia **física** de datos.*

Es la capacidad de modificar el esquema interno sin alterar el esquema conceptual (o los esquemas externos), ni los programas de aplicación.

Las modificaciones en el nivel interno suelen ser necesarias para mejorar el rendimiento, por ejemplo es posible que haya que añadir una nueva estructura de acceso, para agilizar las operaciones de obtención y actualización; si la base de datos sigue conteniendo los mismos datos, no habrá que modificar el esquema conceptual ni los programas de aplicación.

Este tipo de independencia es más fácil de lograr que el anterior (independencia lógica), pues se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento. Por ejemplo, si se añade un camino de acceso nuevo para incrementar el rendimiento de la obtención de registros de PELICULA por director y año, no será necesario modificar ninguna otra consulta (ni aplicación) del tipo "*listar todas las películas rodadas por Francis F. Coppola en 1972*", aunque, por supuesto, el SGBD la ejecutará con mayor rapidez si utiliza el nuevo camino de acceso.

El concepto de *independencia de datos* es similar al concepto de *tipos abstractos de datos* en los lenguajes de programación modernos: ambos ocultan los detalles de implementación a los usuarios, de forma que pueden concentrarse en la estructura general, más que en los detalles de implementación de nivel más bajo.

En todo SGBD con una arquitectura de tres niveles se amplía el *catálogo* para incluir información sobre cómo hacer la correspondencia entre las consultas y los datos entre los diferentes niveles. Se consigue la independencia de datos porque, al modificar el esquema en algún nivel, el esquema del nivel inmediato superior no varía y sólo cambia la correspondencia entre los niveles. Por ello, no hace falta modificar los programas de aplicación que usan el esquema del nivel superior.

De este modo, la arquitectura de tres niveles ayuda a conseguir una verdadera independencia, física y lógica, de los datos. Pero las correspondencias entre niveles suponen un gasto extra durante la compilación o ejecución de una consulta o programa, lo cual disminuye la eficiencia del SGBD. Por esto, pocos SGBD comerciales han implementado la arquitectura de tres esquemas completa y en muchos de ellos no existe una rigurosa separación entre los niveles.

1.2.3 Lenguajes e interfaces de bases de datos

El SGBD debe proporcionar lenguajes e interfaces apropiados para las diversas categorías de usuarios que hemos visto anteriormente.

Lenguajes del SGBD

Cuando el diseño de la BD ha sido terminado y se ha elegido el SGBD concreto en el que implementar dicha BD, hay que especificar los esquemas conceptual e interno de la base de datos, así como las correspondencias entre ellos.

En muchos SGBD en los que no existe una separación estricta entre niveles, el ABD y los diseñadores usan un mismo lenguaje, llamado **lenguaje de definición de datos**, **LDD** (en inglés, *data definition language, DDL*) para especificar los dos esquemas.

El SGBD dispone de un **compilador de LDD** que procesa sentencias escritas en LDD para identificar las descripciones de los elementos de los esquemas y almacenar éstas en el catálogo del SGBD.

En caso de que el SGBD distinga claramente entre los niveles conceptual e interno, el LDD sólo se utilizará para especificar el esquema conceptual (entidades, relaciones entre ellas y restricciones). Y se emplea el **lenguaje de definición del almacenamiento**, **LDA** (en inglés *storage definition language, SDL*) para especificar el esquema interno (las estructuras de almacenamiento y de acceso). Las correspondencias entre los dos esquemas pueden escribirse en cualquiera de los dos lenguajes, LDD o LDA.

Para conseguir una verdadera arquitectura de tres esquemas, sería necesario un tercer **lenguaje de definición de vistas**, **LDV** (en inglés *view definition language, VDL*) para especificar las vistas de usuario y sus correspondencias con el esquema conceptual. No obstante, en la mayoría de los SGBD se utiliza el LDD para definir tanto el esquema conceptual como los externos y las correspondencias.

Después de compilar los esquemas de la base de datos y de introducir datos en la misma, los usuarios necesitan disponer de algún medio para acceder a y manipular dichos datos. Las operaciones más comunes son la *obtención* (recuperación o consulta), la *inserción* (introducción), la *eliminación* y la *modificación* de datos. Para esto, el SGBD proporciona un **lenguaje de manipulación de datos**, **LMD** (en inglés *data manipulation language, DML*).

Sin embargo, los SGBD comerciales actuales no disponen de varios lenguajes, sino que ofrecen un **único lenguaje** "integrado" que cuenta con elementos para definir esquemas conceptuales y vistas, manipular datos y definir su almacenamiento. Por ejemplo, el sistema gestor de bases de datos relacionales Oracle proporciona un lenguaje mezcla de LDD, LMD, LDV y LDA.

Existen dos **tipos de LMD**: declarativo y procedimental.

a. LMD declarativo o no procedimental o de alto nivel
- Requiere que el usuario especifique **qué** datos necesita obtener o actualizar, sin especificar cómo obtenerlos o modificarlos (es un lenguaje *declarativo*).
- Puede utilizarse de forma independiente para realizar operaciones complejas de base de datos.
- Es posible usarlo de dos formas:
  - · Interactivamente (desde una terminal),
  - · Incorporado en un programa escrito en un lenguaje de programación de propósito general, como C o Pascal (*LMD empotrado*).
- Puede recuperar (obtener) y actualizar muchos registros con una sola sentencia (*LMD orientado a conjuntos*).
- Un ejemplo es el lenguaje de bases de datos relacionales **SQL**, que estudiaremos con profundidad más adelante.

b. LMD procedimental o de bajo nivel
- Requiere que el usuario especifique **qué** datos necesita obtener o modificar y **cómo** obtener y actualizar tales datos.
- Siempre debe estar incrustado (*LMD empotrado*) en el código de un programa escrito en un lenguaje de programación de propósito general.
- Normalmente sólo permite obtener uno a uno los registros de la BD para procesarlos por separado (*LMD orientado a registros*).
- Así pues, necesita usar los elementos de dicho lenguaje, como los *bucles*, para poder obtener cada uno de los registros del conjunto de los que interesan almacenados en la BD, y procesarlo individualmente.

Siempre que las sentencias en LMD se incorporen en un programa escrito en cierto lenguaje, a este último se le llama ***lenguaje host*** (o anfitrión) y al LMD, sublenguaje de datos o *lenguaje embebido*[5].

Cuando el LMD no procedimental se usa de forma independiente e interactiva, se llama **lenguaje de consulta** (aunque sea usado tanto para consultar (obtener, recuperar) datos como para actualizarlos).

Los *usuarios ocasionales* utilizan un lenguaje de consulta para especificar sus solicitudes de información. Los *programadores* suelen utilizar el LMD embebido en algún lenguaje de programación (Cobol, PL/I, Pascal, C). Los *usuarios paramétricos* usan interfaces "amigables" que les permiten interactuar con la base de datos; también pueden usarlas los *usuarios ocasionales* y aquellos no interesados en aprender un lenguaje de consulta.

Interfaces del SGBD

1. *Interfaces basadas en menús*.

   Presentan al usuario una ventana con una lista de opciones de menú que lo guían para formular solicitudes. Así no debe memorizar órdenes ni aprender la sintaxis de un lenguaje de consulta.

2. *Interfaces basadas en formularios.*

   Presentan un formulario que muestra sus campos en blanco. El usuario puede rellenarlos para introducir registros nuevos, o bien rellenar sólo algunos campos para recuperar todos los registros cuyo contenido coincida con los datos especificados. Los formularios suelen ser diseñados y programados para usuarios paramétricos como interfaces de transacciones programadas. Algunos SGBDs disponen de utilidades para la creación de formularios.

3. *Interfaces gráficas*.

   Presentan al usuario los esquemas de la BD en forma de diagramas[6] y el usuario puede formular consultas manipulando el diagrama. Suelen combinarse con menús y formularios.

4. *Interfaces de lenguaje natural*.

   Aceptan solicitudes escritas en un idioma determinado y tratan de "entenderlas". Si la traducción tiene éxito, la interfaz genera la correspondiente consulta de alto nivel y la envía al SGBD para que la procese; si no se puede interpretar la solicitud, se dialoga con el usuario para aclararla.

5. *Interfaces para usuarios paramétricos*.

   Este tipo de usuarios dispone de un pequeño conjunto de operaciones que realizan muchas veces (como ocurre con los empleados de banco que atienden al público en las ventanillas). Los analistas de sistemas y los programadores diseñan e implementan una interfaz especial para cada clase de usuarios paramétricos: esta interfaz incluye un conjunto de órdenes abreviadas o comandos, para reducir al mínimo el número de teclas necesarias que es necesario presionar, o de pulsaciones de ratón, para invocar cada operación y ganar rapidez.

6. *Interfaces para el ABD*.

   Estas interfaces permiten al ABD invocar órdenes privilegiadas del SGBD, que sólo él puede ejecutar. Por ejemplo, a través de ellas el ABD creará nuevas cuentas de usuario, concederá permisos de acceso a las cuentas, establecerá los parámetros de ajuste del rendimiento del sistema, modificará los esquemas de la base de datos o las correspondencias entre ellos, accederá al catálogo, reorganizará la estructura de almacenamiento de la BD, realizará o restaurará copias de seguridad, etc.

[5] También suele denominarse lenguaje *incrustado*, o *empotrado*.
[6] Al estilo de un explorador de ficheros y carpetas.

1.3 ESTRUCTURA GENERAL DEL SISTEMA DE BASES DE DATOS.

Los SGBD son sistemas software muy complejos, compuestos de varios módulos software que se encargan de cada una de las responsabilidades del sistema completo. Algunas de estas funciones las puede proporcionar el sistema operativo (SO), pero en general los sistemas operativos sólo proporcionan los servicios más básicos y los SGBD deben construirse sobre esta base (por esto, en el diseño de un SGBD se debe considerar la interfaz entre el SGBD y el SO).

1.3.1 Módulos componentes de un SGBD

Los siguientes son los componentes de procesamiento de consultas:

- El **compilador (o intérprete) de LDD** procesa las definiciones de esquema escritas en LDD, y almacena las descripciones de los esquemas (metadatos) en el catálogo del SGBD. Otros módulos del SGBD necesitan conocer la información contenida en el catálogo.
- El **compilador de consultas** trata cada consulta (escrita en LMD) que se introduce de forma interactiva, realiza un análisis léxico y sintáctico, intenta optimizarla (transformarla en otra equivalente pero más eficiente) y genera una llamada al *procesador de consultas* para que la ejecute[7].
- El **precompilador de LMD embebido** extrae las sentencias en LMD de un programa escrito en un lenguaje host y las envía al **compilador de LMD**, el cual intenta optimizarlas y las convierte en código objeto (instrucciones de bajo nivel que entiende el *procesador de consultas*) para el acceso a la BD. El resto del programa se envía al compilador del lenguaje host. El código objeto de las sentencias LMD se enlaza (*link*) con el código objeto del resto del programa, formando una *transacción programada* cuyo código ejecutable incluye llamadas al *procesador de consultas* de la base de datos.
- El **procesador de consultas**[8] en tiempo de ejecución se encarga de recibir solicitudes de recuperación o actualización, y las ejecuta sobre la base de datos. El acceso a los datos (a disco) se realiza mediante el *gestor de datos almacenados*.

Los siguientes son los componentes de gestión de almacenamiento, que proporcionan la interfaz entre los datos almacenados y los programas de aplicación y envío de consultas al sistema.

- **Subsistema de control de concurrencia y recuperación** (o **gestor de transacciones**), que...
  - · Asegura la consistencia[9] y coherencia[10] de los datos cuando varios usuarios actualizan a la vez la misma información en la BD.
  - · Detecta fallos o caídas del sistema, en cuyo caso debe llevar a cabo la restauración de la base de datos a un estado consistente (correcto).
- **Subsistema de integridad**
  - · Determina si las actualizaciones de los datos son correctas o, por el contrario, si incumplen (violan) alguna restricción de integridad, en cuyo caso realiza la acción adecuada.
- **Subsistema de seguridad**
  - · Asegura que se cumplen las restricciones de seguridad en el acceso a la base de datos o a determinados datos.
- **Gestor de datos almacenados y de la memoria intermedia**, que controla el acceso a la información del SGBD almacenada en disco (datos o metadatos). Se encarga de la reserva de espacio de almacenamiento en disco y las estructuras de datos usadas para representar la información en disco.

[7] En los libros [EN 2002] y [EN 1997] este módulo aparece definido por separado, mientras que en [SKS 1998] aparece integrado en el "compilador de LMD".
[8] En los libros [EN 2002] y [EN 1997] lo denominan "procesador de la base de datos en tiempo de ejecución". Por otro lado, en [SKS 1998] lo llaman "motor de evaluación de consultas", puesto que el término "procesador de consultas" se usa para englobar el "precompilador de LMD", el "compilador de LMD", el "intérprete de LDD" y el "motor de evaluación de consultas".
[9] Consistencia: duración, estabilidad, solidez.
[10] Coherencia: conexión, relación o unión de unas cosas con otras.

Este componente puede emplear los servicios básicos del SO para transferir datos de bajo nivel entre el disco y la memoria principal del ordenador. Es el responsable de otros aspectos de transferencia de datos, como por ejemplo el manejo de las áreas de almacenamiento intermedio (*buffers*) en la memoria principal, donde se llevan los datos desde el disco para que después otros módulos del SGBD puedan procesarlos. También se encarga de decidir qué datos tratar en la memoria caché.

Además de los componentes anteriores, se necesitan varias **estructuras de datos** como parte de la implementación física del sistema:

- **Ficheros de datos** en disco, que almacenan los datos en sí.
- **Estructuras** (o caminos) **de acceso** (ficheros de índices, por ejemplo), que permiten acceso rápido a elementos de datos que tienen valores particulares.
- El **catálogo** del SGBD, que almacena metadatos acerca de la estructura de la base de datos.
- **Datos estadísticos** sobre los datos en la base de datos. Esta información es necesaria para seleccionar las formas eficientes de ejecutar una consulta (optimización). Se suele considerar que está contenida en el catálogo.

1.3.2 Utilidades del sistema de bases de datos

El SGBD es el componente software más importante en un sistema de bases de datos, pero no el único. Existen otras aplicaciones o utilidades que ayudan al ABD a manejar el sistema. Estas utilidades realizan las siguientes funciones:

- *Utilidades para Carga*. Se utilizan para cargar ficheros de datos ya existentes (de texto, por ejemplo) en la base de datos. Normalmente se indica el formato de los datos del fichero fuente y el que deben tener en la base de datos destino, y la utilidad de carga (herramienta de conversión) convierte los datos de un formato a otro para almacenarlos en la BD. Esto permite el intercambio de información entre bases de datos gestionadas por el mismo o por diferentes SGBD (por ejemplo pasar datos de una base de datos Oracle a una base de datos Microsoft SQL Server).
- *Utilidades para Respaldo*. Crean una copia de seguridad (*backup*) del contenido de la base de datos. Esta copia puede utilizarse para restaurar la BD después de un fallo general del sistema.
- *Utilidades para Monitorización* o vigilancia del rendimiento. Permiten supervisar el uso de la BD y proporcionan datos estadísticos al ABD, que los utiliza para tomar decisiones con el objetivo de mejorar el rendimiento o funcionamiento del sistema de bases de datos.
- *Utilidades para Reorganización de ficheros*. Permiten modificar la organización de los ficheros de la BD, para mejorar el rendimiento. Pueden incluir utilidades para ordenar y comprimir ficheros.
- *Utilidades para Control de accesos de los Usuarios* de la base de datos.
- *Utilidades para acceso al Diccionario de Datos* (véase Anexo 2).

1.3.3 Recursos de comunicaciones

El SGBD interactúa con el software de comunicaciones, el cual permite que usuarios remotos[11] accedan al sistema de bases de datos mediante terminales de ordenador, estaciones de trabajo (*workstation*), ordenadores personales (PCs), etc., los cuales se conectan al ordenador en el que reside la base de datos a través de redes de larga distancia o de área local, línea telefónica, o dispositivos de comunicación por satélite.

Muchos SGBD comerciales disponen de paquetes (software) de comunicaciones, que funcionan conjuntamente con el SGBD. El sistema integrado por el SGBD y el sistema de comunicaciones de datos suele denominarse sistema DB/DC (*data base/data communications*).

[11] Usuarios NO locales: acceden a los datos almacenados desde puestos localizados físicamente lejos del equipo en el que reside la base de datos.

Anexo 1. Clasificación de los SGBD.

a) Según el **modelo de datos** en el que está basado.

1. *Relacional*
2. *Orientado a Objetos*
3. *Objeto-Relacional*
4. De *Red*
5. *Jerárquico*
6. *Otros...*

b) Según el **número de usuarios** a los que da servicio simultáneamente.

1. *Monousuario*
2. *Multiusuario*

c) Según el **número de sitios** en que se **almacenan** los datos.

1. *Centralizado*. La base de datos y el SGBD residen en un único ordenador.
2. *Distribuido* (SGBDD). La base de datos y el software del SGBD pueden estar repartidos en varios sitios conectados en red. La base de datos distribuida (BDD) es una colección de datos que pertenece *lógicamente* al mismo sistema, pero que *físicamente* está dispersa en varios sitios de una red de ordenadores.

   2.a. *SGBDD Homogéneo*. Usa el mismo software de SGBD en todos los sitios.

   2.b. *SGBDD Heterogéneo*. Cada sitio puede tener un *software* de SGBD distinto. En particular, el llamado *SGBD Federado o MultiBase de datos* suele construirse a partir de varios sistemas de bases de datos ya existentes (diferentes o no entre sí).

   Este es un tipo de SGBD híbrido entre centralizado y distribuido. Por un lado, un usuario de cierto sistema de BD puede acceder al éste de forma local, como si de un sistema centralizado se tratase. Por otro lado, el sistema visto globalmente es un sistema distribuido, puesto que un usuario puede acceder a los datos almacenados en cualquier otro sitio (ser cliente de cualquier sistema componente de la multibase de datos).

   Así, los SGBD no son del mismo tipo, son independientes entre sí, están débilmente acoplados y tienen cierto grado de autonomía local (es decir, se permite a las transacciones locales tener acceso directo a su propio SGBD).

d) Según su **propósito**

1. *Propósito General*. Cualquier aplicación puede comunicar con él para acceder a la información de la base de datos.
2. *Propósito Específico*. Es decir, construido para un *tipo determinado* de aplicaciones cuyo rendimiento es muy importante, como ocurre con las denominadas OLTP[12], que son aplicaciones que consisten en un gran número de transacciones de actualización de datos, que deben ejecutarse de forma concurrente y sin retrasos excesivos (el rendimiento es de primordial importancia). Un ejemplo sería una aplicación encargada de la reserva de plazas en vuelos de diferentes líneas aéreas.

e) Según su **distribución**

1. *Comercial*. Ejemplos de SGBD cuya obtención supone un coste económico son Oracle, Microsoft SQL Server e Informix.
2. *De libre distribución.* Dos de los SGBD libres más conocidos son MySQL y PostgreSQL.

[12] *Online Transaction Processing*, es decir sistemas de procesamiento de transacciones en línea.

Anexo 2. El catálogo y el diccionario de datos del sistema.

El *diccionario de datos* almacena información más amplia y variada que el *catálogo* de la BD. Aunque en esta asignatura nos centraremos en el estudio del catálogo del sistema y no los sistemas de diccionario de datos generales, veamos algunas cuestiones interesantes acerca de ambos términos.

El catálogo del sistema

El catálogo del sistema es una mini-base de datos que almacena los esquemas (descripciones) de las bases de datos que mantiene (gestiona) el SGBD. De hecho, cada base de datos está descrita por los datos almacenados en el catálogo (llamados metadatos (es decir, datos sobre los datos) que describen su estructura, restricciones, autorizaciones, etc.).

Contiene una descripción del esquema conceptual (EC), del esquema interno (EI), de los esquemas externos (EE) y de las correspondencias entre ellos. Además contiene la información necesaria para los componentes del SGBD relacionados con el procesamiento de consultas[13] y los de seguridad y autorización.

El sistema de diccionario de datos

Un sistema de diccionario de datos (SDD) es un mini-SGBD que gestiona los **metadatos** del SBD (es decir, la información contenida en el catálogo, relativa a los esquemas, restricciones, autorizaciones, etc.), **junto con** otro tipo de información:

- **decisiones de diseño** y resultados de cada fase del diseño de bases de datos,
- **normas** de uso,
- **descripción de programas** de aplicación, y transacciones,
- información sobre los **usuarios** y **documentación** existente,
- otra información relevante para la administración del sistema.

Un SDD útil permitirá almacenar y controlar:

a. Descripciones de los *esquemas* del SBD
b. Información acerca del *diseño físico* de la BD:
   - estructuras de almacenamiento (tipos de ficheros),
   - caminos o estructuras de acceso a los datos,
   - tamaño de los ficheros, registros, etc.
c. Descripción de los *usuarios*, sus *responsabilidades* y *derechos de acceso*, etc.
d. Descripciones de alto nivel de las transacciones y aplicaciones de la BD, y de las relaciones entre los usuarios y las transacciones.
e. Relación entre las transacciones y la información a la que hacen referencia (consultan o modifican); disponer de este tipo de relaciones es útil para determinar qué transacciones son afectadas cuando se modifica la estructura de los datos.
f. Cifras estadísticas de uso: frecuencias de consultas, transacciones, nº de accesos a los datos.

Esta información está disponible para el administrador de la base de datos (ABD), los programadores de aplicaciones y los usuarios autorizados, de forma que permite:

- el control del sistema de bases de datos, por parte del ABD,
- la mayor comprensión (entendimiento) y aprovechamiento, por parte de los programadores y usuarios.

[13] Por ejemplo, los datos estadísticos necesarios para la optimización de consultas.

Es importante destacar por un lado que **el catálogo está fuertemente acoplado al SGBD**. Proporciona información a los usuarios y (sobre todo) al ABD, pero fundamentalmente es utilizado por ciertos módulos del SGBD. El catálogo se usa con mucha frecuencia, por lo que es muy importante que sea diseñado e implementado de forma tal que el acceso al mismo sea muy eficiente.

Los módulos software componentes del SGBD que utilizan el catálogo son:

- precompiladores y compiladores de LMD, LDD (y LDA) y de consultas,
- subsistemas de control de integridad y seguridad,
- subsistemas de control de concurrencia y recuperación (gestor de transacciones).

Y por otro lado, el **sistema de diccionario de datos es un paquete de software autónomo**. Puede interactuar con los módulos del SGBD citados anteriormente, así como con programas de aplicación y generadores de informes, pero es utilizado principalmente por el ABD, los usuarios finales autorizados y los programadores de aplicaciones.

En grandes organizaciones, el sistema de diccionario de datos se considera tan importante como un SGBD.

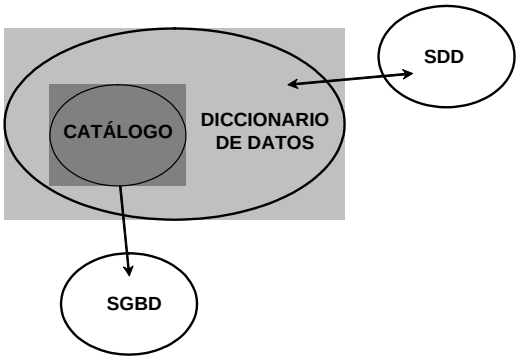

figura 6. Catálogo y Sistema de diccionario de datos

Más información en el capítulo 17 del libro [EN 2002], o en el capítulo 15 de [EN 1997].

BIBLIOGRAFÍA

[EN 2002] Elmasri, R.; Navathe, S.B. **Fundamentos de Sistemas de Bases de Datos. *3ª Edición***. Madrid [etc.]: Addison-Wesley, Pearson Educación, 2002. (Capítulos 1 y 2)

[EN 1997] Elmasri, R.; Navathe, S.B.: **Sistemas de bases de datos. Conceptos fundamentales**. ***2ª Edición***. Wilmington, Delaware, USA: Addison-Wesley Iberoamericana, 1997. (Capítulos 1 y 2)

[MPM 1999] De Miguel, A.; Piattini, M.; Marcos, E. **Diseño de bases de datos relacionales**. Madrid: Ra-Ma, 1999. (Capítulos 1 y 2)

[MP 1993] De Miguel, A.; Piattini, M.: **Concepción y diseño de bases de datos: del Modelo E/R al Modelo Relacional**. Madrid: Ra-Ma, 1993.

[SKS 1998] Korth, H; Silberschatz, A., Sudarshan, S.:**Fundamentos de bases de datos. *3ª Edición***. Madrid: McGraw-Hill, 1998. (Capítulo 1)

[SKS 2002] Silberschatz, A.; Korth, H.F.; Sudarshan, S. "**Fundamentos de Bases de Datos**". 4ª edición. Madrid, McGraw-Hill, 2002. (Capítulo 1)

[CBS 1998] Connolly, T.; Begg C.; Strachan, A. **Database Systems: A Practical Approach to Design, Implementation and Management. *2nd edition***. Harlow, England: Addison-Wesley, 1998. (Capítulos 1 y 2)

[CCM 2003] Celma, M.; Casamayor, JC.; Mota, L. "**Bases de datos relacionales**". Pearson Educación, 2003. (Capítulos 1, 2 y parte del 6)