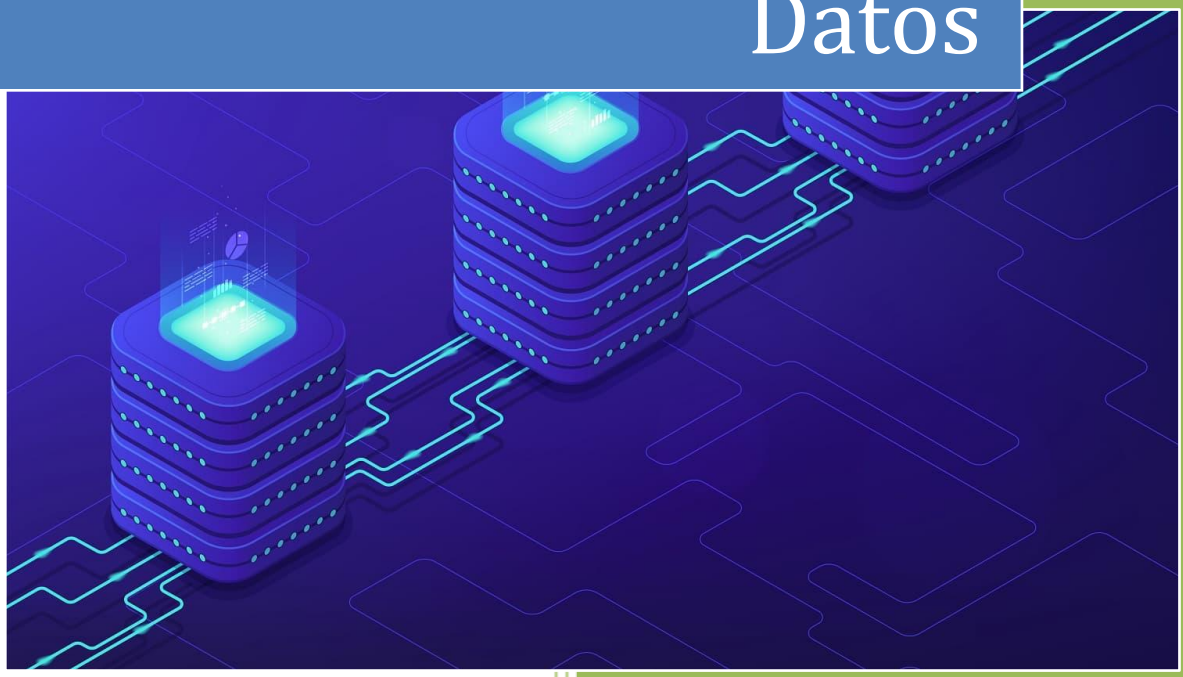
Base de Datos

1. Introducción a las Base de Datos



Curso 2021-2022

Tabla de contenido

Los datos que se utilizan de manera informatizada se almacenan, habitualmente, en bases de datos (BD).

Con el fin de estar en condiciones de abordar el estudio de las BD y del software que sirve para gestionarlas, es decir, los sistemas gestores de bases de datos (SGBD), es imprescindible entender previamente algunos conceptos teóricos fundamentales, referentes a los datos ya su representación. Aparte, pues, de los conceptos relativos a los datos ya las bases de datos, será necesario conocer qué representaciones de la información se utilizan habitualmente así como la evolución del software de este ámbito.

1. LOS DATOS Y LAS BASES DE DATOS

Para todo informático que tenga que trabajar con bases de datos (BD), es imprescindible saber distinguir tres ámbitos bien diferenciados, pero que al mismo tiempo están fuertemente interrelacionados, los cuales hacen referencia, respectivamente, a la realidad, a su conceptualización, ya su representación informática ulterior. Así pues, consideramos los tres "mundos" siguientes:

- **El mundo real**. Está constituido por los objetos (materiales o no) de la realidad que nos interesan y con los que tendremos que trabajar.
- **El mundo conceptual**. Es el conjunto de conocimientos o informaciones obtenidos gracias a la observación de la parte del mundo real que nos interesa. Un mismo mundo real puede dar lugar a diferentes mundos conceptuales, en función del modo de percibir la realidad, o los intereses del observador de esta.
- **El mundo de las representaciones**. Está formado por las representaciones informáticas, o datos, del mundo conceptual, necesarias para poder trabajar.

En la figura 1 se representan, de forma esquematizada, los tres mundos que los informáticos deben considerar, y las interrelaciones que mantienen.

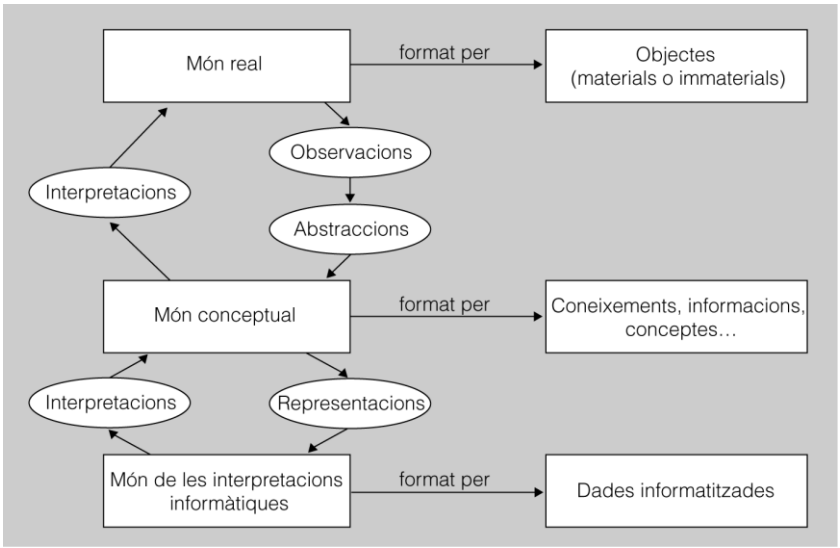


Figura 1 Los tres mundos

1.1. Los datos y su representación

Una vez estructurados, los conceptos en torno a la realidad pasan a ser verdaderas informaciones, con las que los humanos nos podemos comunicar y empezar a trabajar.

Pero aún hay que dar otro paso que nos permita representar estas informaciones, de tal manera que las podamos tratar informáticamente mediante BD y aplicaciones, y aprovechamos así todo el potencial de las nuevas tecnologías.

Los **datos** son representaciones informáticas de la información disponible, relativa a los objetos del mundo real de nuestro interés.

El **mundo de las representaciones** está formado por los datos informatizados con las que trabajamos.

Ahora bien, la conversión de las concepciones en datos no es automática, ni mucho. Requiere pasar por dos fases sucesivas de diseño, en el que se toman decisiones que pueden derivar en resultados dispares. Estas dos fases de diseño son las siguientes:

1. **Fase de diseño lógico**. Se trabaja con el modelo abstracto de datos obtenido al final de la etapa de diseño conceptual, para traducirlo al modelo de datos utilizado por el SGBD con el que se quiere implementar y mantener la futura BD.
2. **Fase de diseño físico**. Se pueden hacer ciertas modificaciones sobre el esquema lógico obtenido en la fase de diseño anterior, con el fin de incrementar la eficiencia en algunas de las operaciones que deban hacerse con los datos.

Por lo tanto, hay que ser conscientes de que, en un mismo conjunto de conocimientos en torno a una misma realidad, estos se pueden representar de maneras diferentes debido, por ejemplo, los siguientes factores:

- Las decisiones de diseño tomadas (tanto a nivel conceptual, como a nivel lógico y físico).
- La tecnología empleada (ficheros, BD relacionales, BD distribuidas, etc.).

La posibilidad de que haya estas diferencias no implica que todos los resultados se puedan considerar equivalentes, sin más, ya que, normalmente, las representaciones diferentes dan lugar a niveles de eficiencia también diferentes. Este hecho puede tener consecuencias importantes, ya que la responsabilidad de todo informático incluye garantizar la corrección de las representaciones, pero también la eficiencia de las implementaciones.

1.2. Entidades, atributos y valores

Tres elementos caracterizan fundamentalmente las informaciones:

Las **entidades** son los objetos del mundo real que conceptualizamos. Son identificables, es decir, distinguibles unos de otros. Y nos interesan algunas (como mínimo una) de sus propiedades.

Los **atributos** son las propiedades de las entidades que nos interesan.

Los **valores** son los contenidos concretos de los atributos.

En principio, los atributos deberían almacenar un solo valor en cada instante. De esta manera, nuestros modelos serán, desde el principio, compatibles con el modelo lógico de datos más ampliamente utilizado: el **modelo relacional** .

Atributos Multivaluados

Permiten almacenar más de un valor simultáneamente. Su uso no es muy recomendable porque son incompatibles con el modelo relacional y con la inmensa mayoría de los SGBD que hay en el mercado.

Ejemplo de entidad, atributos y valores

Consideraremos que una película concreta es una entidad, porque es un objeto del mundo real, que hemos conceptualizado dentro de una categoría (la de los filmes cinematográficos), y que al mismo tiempo es distinguible de otras entidades de la misma categoría (es decir, de otros films).

De esta película nos interesarán algunos aspectos, que llamaremos atributos , como por ejemplo, el título, el director y el año de producción.

Finalmente, estos atributos adoptarán unos valores concretos como, y respectivamente, Paths of glory , Stanley Kubrick y 1957.

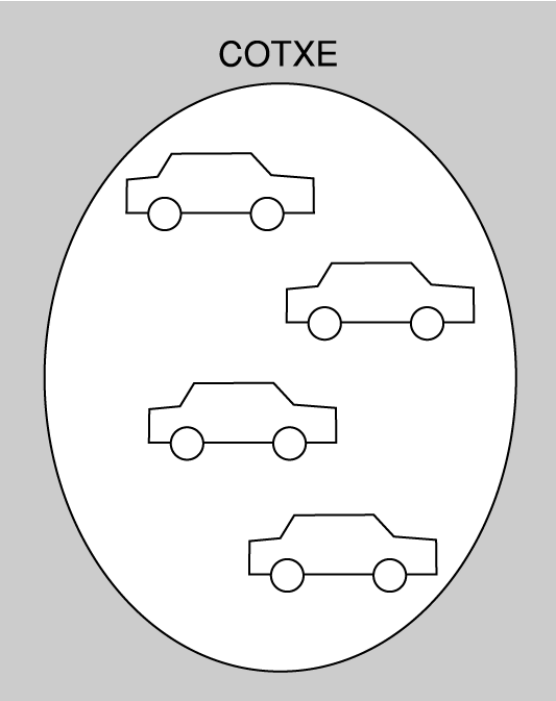
Si sólo conocemos dos de estos tres elementos, no dispondremos de una verdadera información, ya que se producirá alguna de las siguientes carencias:

- Desconoceremos la entidad (el objeto) al que va asociado el atributo y el valor respectivo, y por tanto no servirá de mucho conocer los otros aspectos.
- Desconoceremos qué atributo (propiedad) de la entidad adopta el valor obtenido, lo que puede dar lugar a equívocos.
- Sabremos que la entidad tiene una cierta propiedad, pero en desconoceremos el valor, y por lo tanto este conocimiento difícilmente nos resultará útil.

1.3. Entidades tipo y entidades instancia

El término *entidad* hace referencia a algún objeto concreto del mundo real, conceptualizado, y del que nos interesan algunas características. Pero este término puede tener dos significados diferentes, según a que haga referencia. Para distinguirlos, se puede añadir un adjetivo al sustantivo mencionado y obtener, así, los dos sintagmas siguientes:

Entidad tipo. Se trata de un tipo genérico de entidad o, si se prefiere, de una abstracción, que hace referencia a una clase de cosas como, por ejemplo, los coches, en general.



Entidad instancia. Se refiere a la conceptualización de un objeto concreto del mundo real, como un coche concreto, distinguible de los otros objetos del mismo tipo, gracias a alguna propiedad (como podría ser el valor del atributo Matrícula).

En terminología de teoría de conjuntos, podríamos decir que una entidad tipo es un conjunto, y que cada entidad instancia es un elemento del conjunto, tal como se refleja en la figura adyacente.

1.4. Tipo de dato y dominio de los atributos

Llamamos **dominio** todo el conjunto de valores que un atributo determinado puede tomar válidamente.

El concepto dominio no se corresponde con el de *tipo de dato* , utilizado tanto en el ámbito de las BD como también en el de programación.

Un **tipo de dato** define un conjunto de valores con unas características comunes que los hacen compatibles, por lo que también define una serie de operaciones admisibles sobre estos valores.

Ejemplo de tipo de dato

Podemos considerar los números enteros como un tipo de dato (diferente de otros tipos, como por ejemplo los números reales, los caracteres, etc.), sobre el que se pueden definir ciertas operaciones, como la suma, la resta, la multiplicación o la división entera (pero no la división exacta, que sólo es posible entre los números reales).

Así pues, ambos conceptos (dominio y tipo de dato) se parecen, ya que ambos limitan los valores aceptables. Lo que los diferencia es que un dominio no establece por sí mismo ningún conjunto de operaciones, mientras que un tipo de dato, por definición, si que lo hace.

Otra diferencia es que, en la práctica, un dominio es un subconjunto de valores posibles de un tipo de dato. En algunos casos, puede interesar delimitar el rango de

valores admitidos por un tipo de dato determinado. Esto se hace estableciendo un dominio.

Ejemplo de dominio

Imaginemos que, en el ámbito de unos estudios determinados, se exige un mínimo de asistencia a clases presenciales para conseguir el título correspondiente, independientemente de las calificaciones obtenidas.

Imaginemos que se admite, durante todo el curso académico, un máximo de veinte faltas injustificadas. Pues bien, podría haber un atributo de la entidad ALUMNO, llamado, por ejemplo, *SobreNombre*, que recogiera esta circunstancia.

Este atributo podría almacenar datos de tipo entero. Y también se podría limitar el dominio de 0 (para indicar que no ha habido ninguna inasistencia) a veinte faltas injustificadas, ya que al llegar a este límite se produciría la expulsión del alumno.

1.5. Valor nulo de los atributos

A veces, el valor de un atributo es desconocido o, incluso, no existe. Para representar esta circunstancia, el atributo en cuestión deberá admitir el valor nulo.

La expresión **valor nulo** indica que no hay ningún valor asociado a un atributo determinado de una entidad instancia concreta.

Para que un atributo admita el valor nulo, se debe especificar esta posibilidad a la hora de definir el dominio.

Ejemplo de valor nulo

Consideremos ahora que la entidad ALUMNO dispone de un atributo llamado Teléfono, para almacenar un número telefónico de contacto de cada una de las personas matriculadas.

Si el atributo Teléfono no admitiera valores nulos, el sistema no permitiría que se matricularan personas que no dispusieran de teléfono.

En cambio, si se ha admitido esta posibilidad en definir el dominio del atributo que ahora nos ocupa, el sistema aceptará la matrícula de las personas que no dispongan de teléfono, o bien de las que sencillamente no se lo saben de memoria y que, por tanto, no lo pueden indicar en el momento de formalizar la matrícula, por lo que su incorporación en la BD queda pendiente.

No se debe confundir valor nulo con el cero, si estamos tratando con valores numéricos, o con el espacio en blanco, si estamos trabajando con caracteres. Tanto el uno como el otro son valores con un significado propio. En cambio, el valor nulo implica la ausencia total de valor.

1.6. Atributos identificadores y claves

Un **atributo identificador** es lo que permite distinguir inequívocamente cada entidad instancia del resto, debido a que su valor es único, y no se repite en diferentes entidades instancia.

Los atributos de una entidad serán identificadores, o no, en función del objeto del mundo real que la entidad quiera modelar.

Ejemplo de atributo identificador

El atributo DNI puede servir muy bien para identificar las instancias de una entidad que modelice los alumnos de un centro docente, ya que cada alumno tendrá un DNI diferente.

Pero si la entidad con la que trabajamos quiere modelar las calificaciones finales de los alumnos, el DNI por sí solo no permitirá identificar las diferentes entidades instancia, ya que cada alumno corresponderá una nota final para cada asignatura en la que se haya matriculado.

A veces, un solo atributo no es suficiente para identificar inequívocamente las diferentes instancias de una entidad. Entonces, hay que recurrir a la combinación de los valores de dos o más atributos de la misma entidad.

Todo atributo o conjunto de atributos que permiten identificar inequívocamente las instancias de una entidad se denominan **claves**.

Todo atributo identificador es, al mismo tiempo, una llave. Pero los atributos que forman parte de un conjunto de más de un atributo que actúa como clave de la entidad no son identificadores, ya que, por sí mismos, no son capaces de identificar las entidades instancia.

Ejemplo de clave formada por más de un atributo

Con el fin de diferenciar las instancias de una entidad que quiere reflejar las notas finales de los alumnos en cada asignatura en que se hayan matriculado, es necesario combinar los valores de dos atributos: uno que designe el alumno de que se trate (típicamente, el DNI), y otro que indique la asignatura a la que corresponde la nota (que podría ser algo así como CodigoAsignatura).

Esto es así porque un alumno podrá estar matriculado en diferentes asignaturas, por lo que su DNI se repetirá en diferentes instancias. Y, al mismo tiempo, varios de alumnos podrán estar matriculados de una misma asignatura y, por tanto, los valores del atributo CodigoAsignatura también se podrán repetir.

En cambio, cada alumno tendrá una instancia para cada asignatura en que se haya matriculado, hasta que la apruebe, para reflejar la nota final. Modelizada la entidad de este modo, la combinación de valores de DNI y CodigoAsignatura no se repetirá nunca, y el conjunto formado por estos dos atributos podrá servir como clave.

Por definición, ni los atributos identificadores ni los que forman parte de una clave pueden admitir el valor nulo, porque entonces no servirían para distinguir las entidades instancia sin valor en uno de los tipos de atributo mencionado del resto.

1.7. El mundo de las representaciones

Ya sabemos que los datos son informaciones representadas informáticamente. Por tanto, también podríamos llamar ***mundo de los datos*** al mundo de las representaciones.

La representación informática más frecuente en el ámbito de las BD es la representación tabular, la que se implementa habitualmente en archivos que se estructuran en registros y campos.

En el fondo, las BD sólo son conjuntos de archivos interrelacionados (o, si se prefiere, que almacenan datos que están interrelacionadas). Pero no sirve de nada almacenar

datos si, posteriormente, no accedemos. Hay diferentes tipos de acceso a los datos que conviene examinar: secuencial, directo, por valor y por posición.

1.8. Las representaciones tabulares y su implementación: los archivos

Las informaciones son conceptualizaciones obtenidas a partir de la observación del mundo real. Ahora bien, las informaciones no son muy cómodas para trabajar.

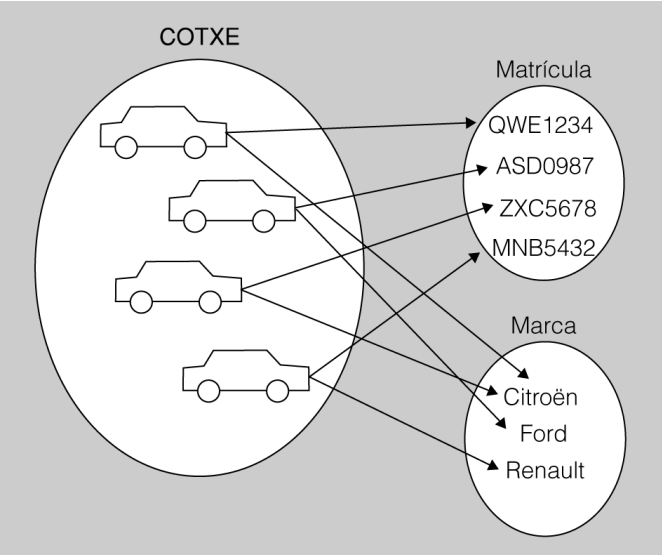


Figura 2 Ejemplo de representación no informatizada

En la figura 2 podemos ver una representación gráfica, no informatizada, de la entidad COCHES, que consta de dos atributos: Matrícula y Marca. Evidentemente, si aumentara el número de entidades instancia, o bien el número de atributos a considerar, este tipo de representación no serviría para nada.

Es necesario representar las informaciones para facilitar las tareas a hacer con ellas, como las consultas, los procesamientos, las transmisiones, etc.

La representación más frecuente en el ámbito informático de las BD es la llamada **representación tabular** (o, lo que es lo mismo, *en forma de tabla*).

Cada tabla representa una entidad genérica, y está estructurada en filas (agrupaciones horizontales de celdas) y columnas (agrupaciones verticales de celdas):

- Cada fila representa una entidad instancia.
- Cada columna representa un atributo.
- Cada celda (es decir, cada intersección de una fila y una columna) almacena el valor que tenga el atributo de la entidad instancia de que se trate.

Archivo

El término archivo tiene otras acepciones, como en el ámbito de los sistemas operativos, que no tienen mucho que ver con el concepto expuesto aquí.

La implementación informática de las representaciones tabulares se materializa mediante los llamados **ficheros de datos**. Se entiende por archivo la

implementación informática de una mesa, con los datos estructurados en registros y campos.

Los ficheros se implementan siguiendo estas consideraciones:

- La implementación de cada entidad instancia llama **registro**, y equivale a una fila de la representación tabular.
- La implementación de cada atributo se denomina **campo**, y equivale a una columna de la representación tabular.
- Cada intersección de un registro y de un campo almacena el **valor** que tenga el campo del registro de que se trate.

Los archivos se almacenarán en algún dispositivo de memoria externa del ordenador, típicamente un disco duro, para conservar los datos permanentemente. El almacenamiento en la memoria interna no satisface este objetivo porque es volátil.

En la tabla 1, podemos ver una representación tabular de la entidad COCHES, que sólo consta de dos campos: Matrícula y Marca. Si aumentara el número de registros a almacenar sólo habría que añadir más filas. Y si tuviéramos que considerar más campos, sólo habría que añadir más columnas.

Pero en ningún caso se comprometería la complejidad de la estructura tabular, que sería, esencialmente, la misma.

coches	
matrícula	**marca**
QWE1234	Citroën
ASD0987	Ford
ZXC5678	Citroën
MNB5432	Renault

Tabla 1 Ejemplo de representación tabular

1.9. Las BD

Normalmente, cuando hayamos de representar informáticamente ciertas informaciones (correspondientes, pues, en el mundo conceptual), no nos encontraremos con una sola entidad tipo, sino con unas cuantas.

Intuitivamente, podemos presuponer que si partimos de un número concreto de entidades tipo necesitaremos, como mínimo, el mismo número de mesas para representarlas (y probablemente algunas más). Ahora bien, estas tablas, o archivos, no serán objetos inconexos, sino que deberán estar interrelacionados.

Las **interrelaciones** son informaciones que permiten asociar las entidades entre ellas.

Las interrelaciones entre los registros de dos (o más) tablas se hacen mediante campos del mismo tipo de dato que almacenen los mismos valores.

Ejemplo de entidades interrelacionadas

Imaginemos ahora que construimos una pequeña BD. Sólo hay dos entidades tipo de nuestro interés: COCHES y MARCAS.

También tenemos otra información complementaria, sobre la interrelación entre ambas entidades: cada coche será de una marca concreta, pero podrá haber muchos coches de cada marca.

En la figura 3, se muestra una representación de las dos entidades (COCHES y MARCAS) interrelacionadas.

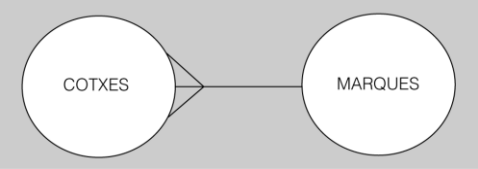


Figura 3

La entidad MARCAS sólo tiene el atributo Marca, y la entidad COCHES sólo tiene el atributo Matrícula. El archivo para representar COCHES deberá añadir un campo adicional, para permitir la interrelación con el fichero que representa la entidad MARCAS.

La figura .4 muestra la interrelación entre los dos ficheros correspondientes a la entidad COCHES ya la entidad MARCAS.

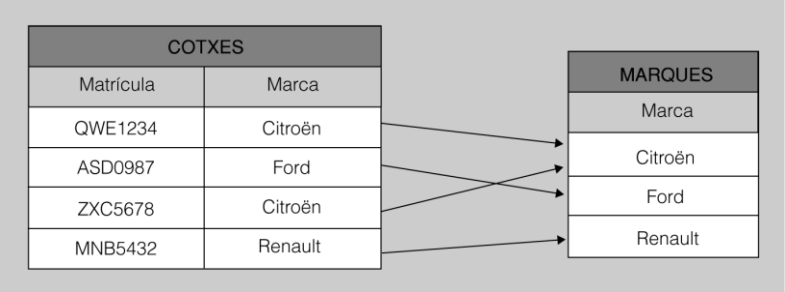


Figura 4 Ejemplo de ficheros interrelacionados

La interrelación entre archivos implica que los cambios de valor de los campos que sirven para interrelacionarlos (o, incluso, su supresión) deben quedar reflejados en todos los archivos implicados, con el fin de mantener la coherencia de los datos. Por tanto, los programas que trabajan con ficheros de datos interrelacionados siempre tendrán un plus de complejidad, derivado de la exigencia que acabamos de comentar.

Una **BD** consiste en un conjunto de ficheros de datos interrelacionados.

Un sistema gestor de bases de datos (SGBD) es un tipo de software especializado en gestionar y administrar bases de datos.

En este sentido, los **SGBD** se han ido desarrollando teniendo como uno de los objetivos principales facilitar la programación con acceso a datos persistentes, y para gestionar el acceso simultáneo a los datos por parte de diferentes usuarios.

1.10. El nivel lógico y el nivel físico

La organización de los datos, y su grabación y acceso, se pueden considerar desde dos puntos de vista, más o menos cercanos a la implementación física de la grabación de los datos:

- **Nivel lógico**. Permite trabajar con los datos de manera más sencilla, independientemente de la implementación física concreta, que no es necesario conocer. Es la manera de trabajar más productiva y, por tanto, la más recomendable, siempre que las circunstancias no nos obliguen a hacer optimizaciones a niveles más bajos.
- **Nivel físico**. Implica un conocimiento a bajo nivel de la implementación física de la organización de los datos y su acceso.

En la figura 5, se muestra la doble perspectiva apuntada, la lógica y la física. En el disco duro se guardan los datos, organizadas de determinada manera a nivel físico. El SGBD nos permite acceder a los datos almacenados en el disco duro considerando sólo aspectos de nivel lógico, tales como secuencias de registros.

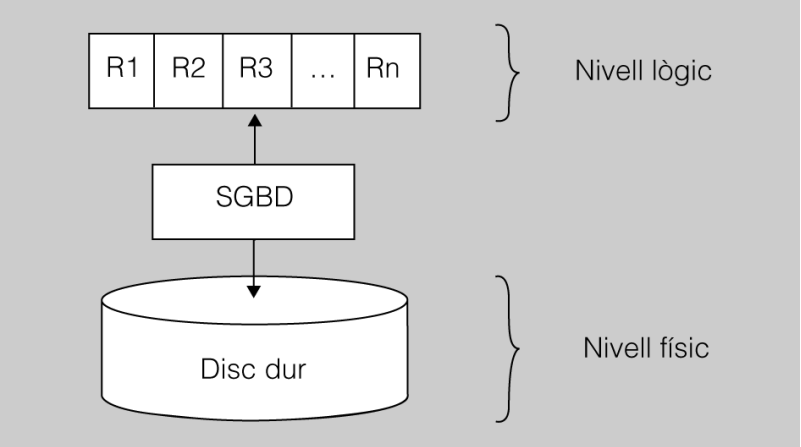


Figura 5 Nivel lógico y nivel físico

Ejemplos de trabajo a nivel lógico ya nivel físico

Nivel lógico: se trabaja teniendo en cuenta, fundamentalmente, las tablas, con los campos y registros correspondientes, y sus interrelaciones.

Nivel físico: se trabaja considerando otros factores a más bajo nivel, como el encadenamiento de los registros físicos, la compresión de datos, las tipologías de índices, etc.

2. CONCEPTOS DE ARCHIVOS Y BASES DE DATOS

Las BD son conjuntos estructurados de datos organizados en entidades, las cuales están interrelacionadas.

Es importante conocer el concepto de BD y su origen. También es interesante establecer una comparativa de ventajas y de inconvenientes según si se utilizan archivos tradicionales o BD.

Hay diferentes perspectivas desde las que se puede observar una BD, y diferentes tareas y métodos de trabajo sobre una BD, que todo informático debe conocer.

En el mercado existen diferentes modelos de BD: el jerárquico, en red, relacional, orientado a objetos, etc. Es interesante, pues, conocer las características de cada uno de los modelos.

2.1. Concepto y origen de las BD

Las BD no existen por casualidad. Aparecieron para dar respuesta a una serie de necesidades. La mejor manera de entender tanto estas circunstancias como el concepto de BD que originaron es hacer una pequeña aproximación histórica a su evolución.

Las aplicaciones informáticas de los años sesenta del siglo XX tenían, por regla general, las características siguientes:

Normalmente, consistían en procesos por lotes (en inglés, batch processing), con las siguientes particularidades:

- Un lote es un conjunto finito de trabajos que se quieren tratar como un todo.
- Su procesamiento, una vez puesto en marcha, no necesita ninguna interacción con el usuario.
- Normalmente, este tipo de ejecución se realiza en tareas repetitivas sobre gran volúmenes de información.
- Realizaban tareas muy específicas, relacionadas con muy pocas entidades tipo, como por ejemplo la emisión de facturas, la confección de nóminas de personal, etc.
- Normalmente, los programas trabajaban de manera secuencial sobre un solo archivo maestro, que estaba almacenado en una cinta magnética (aunque no, en un disco duro), y generaban otro archivo como resultado.
- Cuando se detectaba la necesidad de implementar una nueva aplicación que utilizara parcialmente los datos contenidos en un archivo y, además, algunas otras nuevas, se diseñaba un nuevo fichero con todos los campos necesarios, y se llenaba con los datos correspondientes:
- Los datos que ya estaban en el antiguo archivo se podían copiar en el nuevo, justamente, mediante otro procesamiento por lotes (batch).
- De este modo se conseguía que el nuevo programa no tuviera que trabajar con muchos archivos, lo que en simplificaba el código, por una parte, y de otra optimiza el tiempo de ejecución.

- Como contrapartida, esta manera de trabajar comportaba la redundancia de algunos datos, que eran repetidas en diferentes archivos. Este hecho dificultaba el mantenimiento de la coherencia de estos datos.

Posteriormente, la evolución tecnológica hizo posible la implantación progresiva de tres nuevos elementos:

- Los terminales. Dispositivos de hardware para introducir o mostrar datos de las computadoras.
- Los discos duros. Dispositivos de almacenamiento de alto rendimiento, que no estaban limitados de facto el acceso secuencial.
- Las redes de comunicación.

A partir de estas innovaciones, los programas informáticos tuvieron que implementar la posibilidad de realizar consultas y actualizaciones de los mismos datos, simultáneamente, por parte de diferentes usuarios.

Al mismo tiempo, se fue produciendo un fenómeno que consistía en la integración de las diferentes aplicaciones informáticas que utilizaba cada organización (por ejemplo, gestiones de stocks, facturaciones, proveedores...). Esta tendencia requería las acciones siguientes:

- Interrelacionar los archivos de las antiguas aplicaciones.
- Eliminar la redundancia, es decir, la repetición innecesaria de datos, que además de resultar ineficiente pone en riesgo su coherencia.

Tanto la interrelación de los ficheros como el hecho de que cada vez accedían simultáneamente más usuarios exigían unas estructuras físicas que proporcionaran accesos razonablemente rápidos, tales como índices.

Al principio de los años setenta, estos conjuntos de archivos interrelacionados, compartidos por diferentes procesos y usuarios simultáneamente, y con estructuras complejas, recibieron inicialmente el nombre de *data bases* , o DB (*bases de datos* , o BD, en catalán).

Y al final de los años setenta fueron saliendo al mercado softwares aún más sofisticados, que permitían gestionar más fácilmente las relaciones entre archivos, y ya estaban en condiciones de garantizar la actualización simultánea de datos por parte de diferentes usuarios, etc. Estos softwares se llamaron **sistemas gestores de bases de datos** , o **SGBD** (*data base management systems* , o **DBMS** , en inglés).

Con esta perspectiva histórica, pues, podemos dar una definición de BD más completa.

Una **BD** es la representación informática de los conjuntos de entidades instancia correspondientes a diferentes entidades tipo y de las relaciones entre estas. Este conjunto estructurado de datos debe poder ser utilizado de manera compartida y simultánea por una pluralidad de usuarios de diferentes tipos.

2.2. Ficheros y BD

Los ficheros tradicionales (y los programas necesarios para trabajar) se han encontrado con serias dificultades para satisfacer las crecientes necesidades de los usuarios en prácticamente todos los ámbitos.

Por esta razón, las BD se han ido implantando como mecanismo por excelencia de almacenamiento, procesamiento y obtención de información, desplazando progresivamente los archivos de su posición preeminente anterior. La tabla 2 contiene una breve descripción de las principales diferencias entre los sistemas basados en ficheros tradicionales y las BD.

	Archivos	**Bases de datos**
entidades tipo	Las entidades instancia de un fichero pertenecen a una sola entidad tipo.	Las BD contienen entidades instancia de infinidad de entidades tipo interrelacionadas.
interrelaciones	El sistema no interrelaciona ficheros.	El sistema tiene previstas herramientas para interrelacionar ficheros.
redundancias	Es necesario crear archivos a medida de cada aplicación, con todos los datos necesarios, aunque estén repetidas en otros archivos.	Técnicamente, todas las aplicaciones pueden trabajar con la misma BD, lo que evita la redundancia de datos y los riesgos que conlleva.
inconsistencias	Es posible que los valores de unos mismos datos en diferentes ficheros no coincidan, si los programadores no las han actualizado debidamente.	Si las interrelaciones están bien diseñadas, los datos sólo deben estar almacenados en la BD una sola vez. Por tanto, no hay riesgo de inconsistencias.
Obtención de datos	Si no hay una aplicación que obtenga los datos que queremos, o bien se debe hacer un programa a medida, o bien se debe aprovechar la salida de un programa con objetivos similares, y hacer los cálculos necesarios manualmente.	Permiten obtener cualquier conjunto de datos, según las necesidades, de los de su propio entorno de trabajo, sin tener que escribir, compilar y ejecutar ningún nuevo programa de aplicación contra la BD.
Aislamiento de datos	Los datos están dispersos y aislados en diferentes archivos, lo que dificulta el desarrollo de las aplicaciones.	Todos los datos son en la misma BD, interconectadas, lo que facilita la obtención.
Integridad de datos	Los programas deben implementar todas las restricciones sobre los datos,	La BD se encarga directamente de implementar las restricciones sobre los

	añadiendo el código fuente correspondiente. El mantenimiento es complicado cuando la información se contiene en diferentes ficheros utilizados para diferentes aplicaciones.	datos. Los programas no deben incorporar código fuente adicional para garantizarlas.
atomicidad	Algunos conjuntos de operaciones sobre los datos deben ejecutarse de forma indivisible (o todos o ninguno), independientemente de los fallos que el sistema pueda presentar (como por un corte de suministro eléctrico). Pero esto es muy difícil de garantizar con un sistema de información basado en ficheros.	Las BD incorporan la técnica de las transacciones para garantizar fácilmente la ejecución atómica de una pluralidad de procesos sobre los datos.
acceso concurrente	La actualización simultánea de datos de un mismo archivo por parte de diferentes usuarios o aplicaciones puede provocar fácilmente la inconsistencia.	Con la técnica del bloqueo, las BD garantizan automáticamente la consistencia de los datos, a pesar de que más de un usuario o más de una aplicación las quieran actualizar simultáneamente.
seguridad	Habitualmente, cada fichero sirve para un solo usuario o una sola aplicación (sobre todo simultáneamente), y ofrece una visión única del mundo real. Pero no siempre todos los usuarios que utilizan un fichero deberían tener acceso a todos los datos que contiene.	Una BD puede ser compartida por muchos usuarios de diferentes tipos (incluso, simultáneamente), los cuales pueden tener diferentes visiones (vistas) del mundo real, en función de su perfil y de los permisos que se hayan de conceder en cada caso.

Tabla 2 Ficheros y BD

Evidentemente, las prestaciones de las BD son muy superiores a las que proporcionan los sistemas de ficheros. Pero esto no quiere decir que en algunos casos no sea mejor utilizar archivos, por ejemplo cuando el volumen de los datos a contener es muy pequeño, o cuando sólo tenemos que trabajar con una entidad instancia y, por tanto, no hay que considerar interrelaciones.

Algunas utilizaciones posibles de los ficheros en la actualidad son las siguientes:

- Archivos de configuración de aplicaciones.
- Archivos de configuración de sistemas.
- Ficheros de **registros de eventos** (*logs*).

En casos como estos, no compensaría cargar innecesariamente el sistema con una BD (y con el sistema gestor correspondiente), ya que no se aprovecharían las ventajas de las BD pero, en cambio, empeoraría el rendimiento del sistema.

2.3. El acceso a los datos: tipologías

No sirve de nada estructurar datos y almacenarlas si después no se puede acceder, por consultarlas, modificarlas o transmitirlas.

En general, hay dos maneras básicas de acceder a los datos:

- El **acceso secuencial** a un registro determinado, que implica el acceso previo a todos los registros anteriores.
- El **acceso directo** a un registro concreto, que implica la obtención directa del registro.

Además, hay otra clasificación habitual de tipologías de accesos:

- El **acceso por valor**, que permite obtener el registro en función del valor de algún (o algunos) de sus campos, sin considerar la posición que ocupa el registro.
- El **acceso por posición**, que abre el acceso a un registro que ocupa una posición determinada, sin considerar el contenido del registro.

Combinando las dos dicotomías anteriores, resultan cuatro métodos de acceso a los datos, tal como se muestra en la tabla 3, que se ajustan más a la realidad.

	P-por posición	V-por valor
S- secuencial	SP	SV
D-directo	DP	DV

Tabla 3 Métodos de acceso a los datos

Examinemos, pues, las cuatro tipologías de acceso a datos más frecuentes:

- **SP (acceso secuencial por posición)**. Después de haber accedido a un registro que se encuentra en una posición determinada, se accede al registro que ocupa la posición inmediatamente posterior.
- **DP (acceso directo por posición)**. Se obtiene directamente un registro por el hecho de ocupar una posición determinada.
- **SV (acceso secuencial por valor)**. Después de haber accedido a un registro que tiene un valor concreto, se accede al registro que ocupa la posición inmediatamente posterior, según la ordenación establecida a partir de un campo determinado (o más). El orden será creciente o decreciente, si se trata de un campo numérico, o alfabético ascendente o descendente, si se trata de un campo de caracteres.

- **DV (acceso directo por valor)**. Se obtiene directamente un registro por tener un valor determinado en uno de sus atributos (o más).

Ejemplos de tipos de acceso a los datos

Imaginemos que disponemos un fichero en el que se almacena información relativa a los alumnos de un centro docente: DNI, nombre, apellidos, fecha de nacimiento, dirección, teléfono, etc. A continuación, se da un ejemplo de cada uno de cuatro métodos de acceso estudiados.

SP (acceso secuencial por posición): la lista de todos los alumnos, sin establecer ninguna ordenación.

DP (acceso directo por posición): en el ámbito de la programación, el caso más típico es el de las búsquedas dicotómicas en vectores ordenados; en el ámbito de las BD, este tipo de acceso se produce al utilizar un índice de tipo *hashing* .

SV (acceso secuencial por valor): la lista de todos los alumnos, siguiendo un orden alfabético ascendente, en primer lugar los apellidos y después del nombre.

DV (acceso directo por valor): obtención de los datos almacenados en un registro correspondiente a un alumno concreto, que se diga, por ejemplo, Pedro García Manent (es decir, que el campo Nombre contiene el valor "Pedro", y el campo Apellidos contiene el valor "García Manent").

2.4. Las diferentes visiones de los datos

Uno de los principales objetivos de las BD es proporcionar, a los usuarios, una **visión abstracta de los datos**. Con este fin, el sistema oculta a los usuarios ciertos detalles relativos al almacenamiento y mantenimiento de los datos, para facilitarles el trabajo, por un lado, pero también para garantizar ciertos aspectos en materia de seguridad.

Porque las BD resulten útiles, deben ser mínimamente eficientes a la hora de recuperar los datos. Por este motivo, los sistemas de BD tienen implementadas, a bajo nivel, estructuras de datos bastante complejas.

Se utilizan tres niveles de abstracción **físico**, **lógico** y de **vistas** para ocultar estas estructuras complejas y simplificar, de este modo, la interacción de los usuarios con el sistema.

Nivel físico: es el nivel de abstracción más **bajo** de todos los utilizados.

- Describe cómo se almacenan realmente los datos a bajo nivel, especificando en detalle las complejas estructuras que se necesitan.
- No es frecuente trabajar a este nivel. Sólo se hace cuando se necesitan optimizaciones en la estructuración de los datos a bajo nivel.

Nivel lógico: es el nivel de abstracción **intermedio.**

- Describe todos los datos almacenados en la BD y sus interrelaciones, mediante un número no muy elevado de estructuras bastante simples (típicamente, tablas).
- La implementación de estas estructuras lógicas puede conllevar la presencia de estructuras mucho más complejas a nivel físico. Pero los usuarios del nivel lógico no tienen que preocuparse de esta complejidad. Ni siquiera necesitan conocerla.

- Los administradores de BD trabajan habitualmente con este nivel de abstracción.

Nivel de vistas: es el nivel de abstracción más **alto.**

- La mayoría de los usuarios no necesitan conocer toda la estructuración lógica de la BD con que trabajan. Tratándose de una BD grande, además, conocer toda su estructura puede conllevar un esfuerzo considerable.
- Por otra parte, a menudo, y por motivos tanto de seguridad como de privacidad, no resulta conveniente que los usuarios tengan acceso a todos los datos, sino sólo a la parte que estrictamente necesitan para realizar su trabajo.
- Cada vista sólo describe una parte de la BD. El establecimiento de vistas simplifica la interacción del usuario con el sistema, lo hace más seguro, y proporciona más privacidad. Se pueden establecer diferentes vistas, según las necesidades, sobre la misma BD.

En la figura 6, se pueden ver los diferentes niveles de abstracción utilizados para facilitar la interacción de los usuarios con las BD.

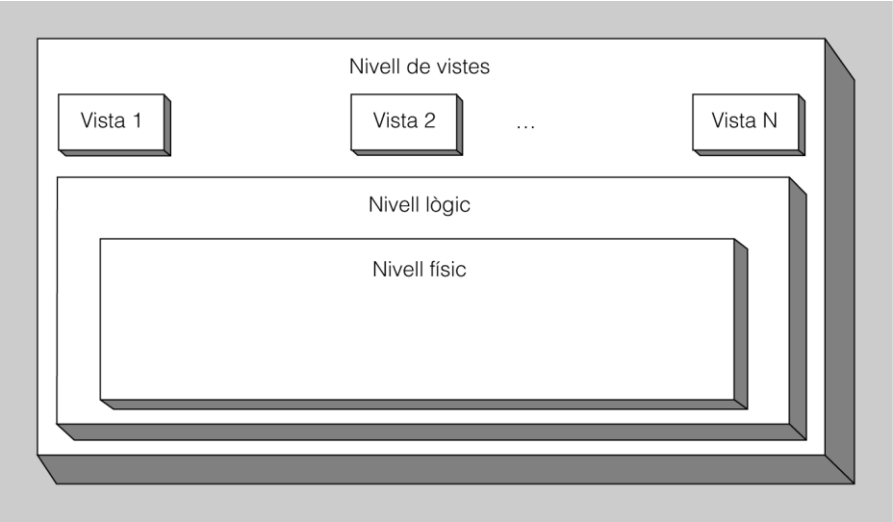


Figura 6 Niveles de abstracción de datos

3. LOS SGBD

Los **SGBD** son un tipo de software que tiene como fines la gestión y el control de las BD.

Es interesante conocer la evolución de este tipo de software a lo largo de su historia, y los objetivos que todos ellos persiguen con más o menos acierto. También hay que destacar que hay nociones relativas tanto a la arquitectura de los SGBD como las aplicaciones que los usan. También se debe destacar que hay diferentes tipologías de usuarios y administradores de BD, y lenguajes que todos deben utilizar para comunicarse con los sistemas gestores.

3.1. Evolución de los SGBD

Para entender mejor por qué los SGBD son hoy en día tal como los conocemos, conviene repasar brevemente su historia.

Al igual que en otros ámbitos de software (como, por ejemplo, en el de los sistemas operativos), la evolución de los SGBD ha sido, a menudo, intrínsecamente ligada a la evolución del hardware.

La figura 7 muestra un esquema de las etapas evolutivas por las que han pasado los SGBD, en el que se indican las principales características.

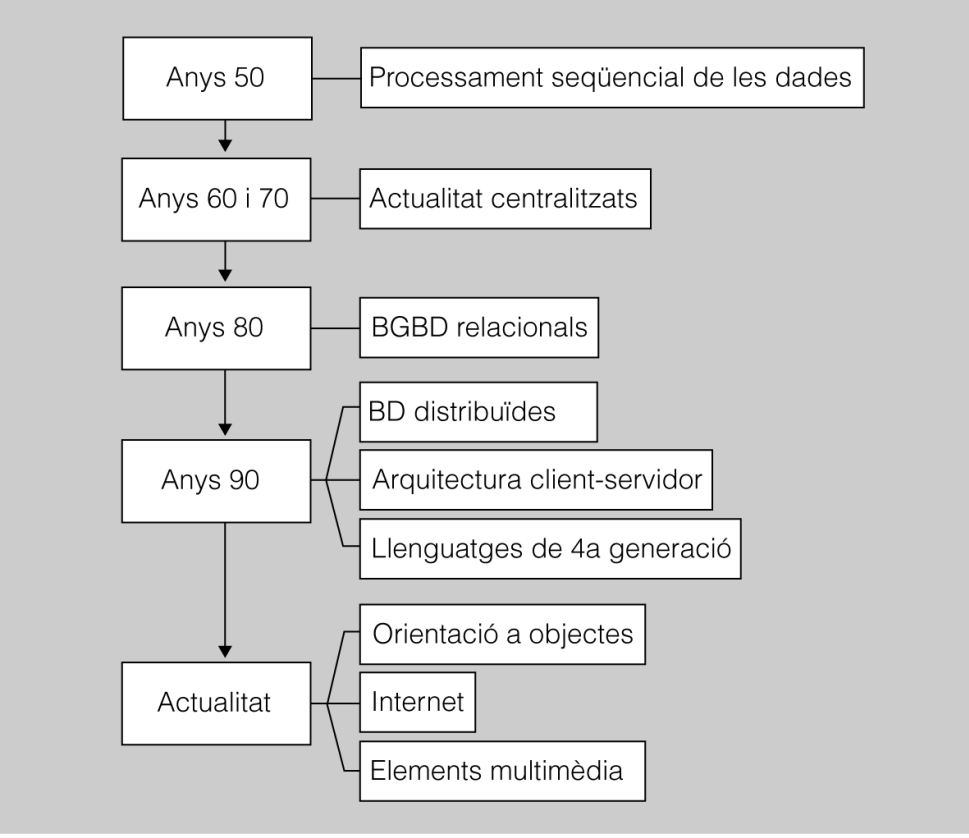


Figura 7 Evolución histórica de los SGBD

AÑOS CINCUENTA: PROCESAMIENTO SECUENCIAL

Inicialmente, el único hardware disponible para almacenar la información consistía en paquetes de cintas perforadas y en cintas magnéticas. Estos dispositivos sólo se podían

leer de forma secuencial y, por tanto, el software de la época estaba limitado por esta circunstancia. Como el tamaño de los datos a procesar era muy superior a la de la memoria principal de las computadoras, los programas sólo podían realizar procesos por lotes, de la siguiente forma:

- Obteniendo los datos en un orden determinado (desde una o más cintas).
- Haciendo algún cálculo sobre los datos.
- Escribiendo el resultado (en una nueva cinta).

Ejemplo de procesamiento secuencial

Imaginemos que una empresa necesita actualizar los precios de los productos que ofrece: en primer lugar, habría que grabar los incrementos de los precios en tarjetas perforadas.

A continuación, se iría leyendo el paquete de las cintas perforadas anteriores, sincronizadamente con la cinta maestra que contuviera todos los datos relativos a los productos. Las tarjetas perforadas deberían respetar el orden de los registros del archivo de productos contenido en la cinta.

Con todos los datos relativos a los productos contenidos en la cinta maestra, más los precios actualizados en función de los nuevos valores reflejados en las tarjetas perforadas, se grabaría una nueva cinta, que pasaría a ser la nueva cinta maestra de los productos de la empresa.

AÑOS SESENTA Y SETENTA: SISTEMAS CENTRALIZADOS

Durante la mayor parte de las dos décadas de los años sesenta y setenta, los SGBD tuvieron una estructura centralizada, como correspondía a los sistemas informáticos de entonces: un gran ordenador para cada organización que se le pudiera costear, y una red de terminales no inteligentes, sin capacidad propia para procesar datos.

Inicialmente, sólo se utilizaban para gestionar procesos por lotes con grandes volúmenes de datos. Posteriormente, con la aparición de los terminales, a veces conectados mediante la línea telefónica, se fueron elaborando aplicaciones transaccionales, por ejemplo para reservar y comprar billetes en líneas de transportes, o para realizar operaciones financieras.

Los programas aún estaban muy ligados al nivel físico, y debían modificar siempre que se hacían cambios en el diseño de la BD, ya que estos cambios implicaban, a su vez, modificaciones en la estructura física de la BD. El personal que realizaba estas tareas tenía que estar altamente cualificado.

AÑOS OCHENTA: SGBD RELACIONALES

Aunque desde el principio de los años setenta ya se había definido el modelo relacional, y el acceso no procedimental a los datos organizados siguiendo este modelo, no fue hasta los años ochenta cuando fueron apareciendo SGBD relacionales en el mercado.

La razón de esta demora en el uso de los sistemas relacionales fue el pobre rendimiento que ofrecían inicialmente los productos relacionales en comparación con las BD jerárquicas y en red. Pero la innovación en el hardware, primero con los

miniordenadores y posteriormente con los microordenadores, supuso un cierto abaratamiento de la informática y su extensión a muchas más organizaciones.

Hasta entonces, el trabajo de los programadores que trabajaban con BD prerrelacionales había sido demasiado pesada, ya que, por un lado, debían codificar sus consultas de manera procedimental, y de otra, tenían que estar pendientes de su rendimiento y hacer consideraciones de índole física a la hora de codificarlas.

Pero debido a la expansión de la informática que tuvo lugar durante la década que comentamos, era necesario simplificar el desarrollo de las aplicaciones. Los SGBD lo consiguieron, todo independizando los programas de los aspectos físicos de los datos.

SQL

En 1986, el Instituto Nacional Norteamericano de Normalización (American National Standards Institute, o ANSI, en inglés) publicó las primeras normas que enunciaban la sintaxis y la semántica del SQL.

Además, la aparición del lenguaje de consulta estructurado (*structured query Language* , o SQL, en inglés) y, sobre todo, su estandarización a partir de 1986 facilitaron enormemente el uso de los sistemas relacionales y, por tanto, su implantación masiva.

Finalmente, las BD relacionales pudieron competir, incluso, en materia de rendimiento con las jerárquicas y con las estructuradas en red, con lo cual terminaron reemplazando sus competidoras en la mayoría de los casos.

AÑOS NOVENTA: BD DISTRIBUIDAS, ARQUITECTURAS CLIENTE / SERVIDOR, Y LENGUAJES DE CUARTA GENERACIÓN

Como ya sabemos, los primeros sistemas de BD eran centralizados: todos los datos del sistema estaban almacenadas en un único gran ordenador al que se podía acceder desde diferentes terminales. Pero el éxito gradual de los ordenadores personales (*personal computers* , o PC, en inglés), cada vez más potentes y con precios más competitivos, junto con el desarrollo de las redes, posibilitó la distribución de una misma BD en diferentes ordenadores (o nodos).

En función del número de SGBD utilizados, los sistemas distribuidos pueden ser de dos tipos:

- **Homogéneos**, si todos los nodos utilizan el mismo SGBD. Las interacciones entre los diferentes nodos son más sencillas. Pero las actualizaciones del sistema gestor implicarán necesariamente a todos los nodos.
- **Heterogéneos**, si cada nodo utiliza un SGBD diferente. Las interacciones entre los diferentes nodos pueden ser más complicadas. Pero habrá más flexibilidad a la hora de actualizar el sistema gestor de cada nodo.

Los puntos a favor de los sistemas distribuidos son fundamentalmente dos:

- **Rendimiento**. Si el sistema está bien diseñado, la mayoría de las operaciones se realizarán con datos almacenados localmente. De esta manera las respuestas serán más rápidas, disminuirá el gasto en

comunicaciones, y se evitará la dependencia de un nodo central colapsado.

- **Disponibilidad**. Los sistemas distribuidos son más resistentes a los paros que los centralizados. En un sistema centralizado, el paro del nodo central para todo el sistema. En cambio, en un sistema distribuido, si uno de los nodos queda temporalmente fuera de servicio por cualquier eventualidad, el resto continuará funcionando normalmente, y podrá dar servicio siempre que no se necesiten los datos almacenados en el nodo parado. Pero, además, según qué esquema de diseño se haya seguido al hacer la distribución, si los datos del nodo parado están duplicadas en otro, continuarán estando disponibles.

Pero, evidentemente, no todo son ventajas. Por ejemplo, en el caso de los sistemas heterogéneos, a menudo es necesario realizar conversiones de datos para permitir la comunicación de los diferentes nodos entre ellos. Además, en general, la comunicación es más compleja y, por tanto, la cantidad de errores y de problemas derivados de este hecho a controlar es mucho mayor que en un sistema centralizado.

Arquitectura cliente / servidor

Una arquitectura cliente / servidor se caracteriza por disponer de un sistema en red donde hay unos ordenadores que actúan como servidores de peticiones y otros (los clientes) que piden servicios.

La tecnología utilizada habitualmente en la distribución de BD es la **arquitectura cliente / servidor** (conocida también como **arquitectura C / S**). Actualmente, todos los SGBD comerciales están adaptados a esta realidad.

El funcionamiento de los sistemas basados en este tipo de arquitectura es, esquemáticamente, el siguiente: dos procesos se ejecutan en un mismo sistema o en dos diferentes, de tal manera que uno hace de cliente (o peticionario de un servicio), y el otro de servidor (o proveedor del servicio solicitado).

La clasificación de los procesos en las categorías de cliente y de servidor es de tipo lógico (y no físico) y, por tanto, hay que tener en cuenta algunos aspectos que derivan de esta circunstancia, como los siguientes:

- Un cliente puede pedir servicios a varios servidores.
- Un servidor puede recibir peticiones de muchos clientes.
- El cliente y el servidor pueden residir en un mismo sistema.

4GL

Estos son algunos entornos actuales de desarrollo que utilizan lenguajes de cuarta generación: eDeveloper, de Magic Software, Oracle Developer, de Oracle Corporation, SAS System, de SAS Institute Inc., etc.

Finalmente, durante los años noventa, la implantación en todas las BD, incluso en pequeños sistemas personales, motivó la aparición de los llamados **lenguajes de cuarta generación** (*fourth generation languages* , o 4GL, en inglés), los cuales se siguen utilizando en la actualidad.

Se trata de lenguajes muy sencillos, pero al mismo tiempo muy potentes, especializado en el desarrollo de aplicaciones centradas en el acceso a BD. Ofrecen muchas facilidades para definir, generalmente de manera visual, ventanas desde las que se pueden consultar, introducir, modificar o borrar datos, incluso en entornos cliente / servidor.

TENDENCIAS ACTUALES: ORIENTACIÓN A OBJETOS, INTERNET, Y ELEMENTOS MULTIMEDIA

Actualmente, los SGBD relacionales acaparan el mercado. Han evolucionado de tal manera que ya no tienen competidores en rendimiento, fiabilidad o seguridad. Por otra parte, ya no necesitan habitualmente tareas de mantenimiento planificadas que comporten la parada periódica. Por tanto, la disponibilidad que ofrecen es muy elevada, ya que se acerca a las veintidós cuatro horas de todos los días del año.

Pero, al mismo tiempo, todos están inmersos en un complejo proceso de transformación con el fin de adaptarse a las innovaciones tecnológicas de más éxito:

1. Las tecnologías multimedia. Los tipos de datos que tradicionalmente admitían los SGBD ahora se ven incrementados por otros nuevos que permiten almacenar imágenes y sonidos.

> **Ejemplo de incorporación de tecnologías multimedia en un SGBD**
>
> De este modo, por ejemplo, la tabla que almacene los datos personales de los empleados de una empresa determinada podrá contener la foto de cada uno, lo que puede resultar especialmente interesante si la organización dispone de un entorno virtual de trabajo (de tipo intranet, por ejemplo) en el que los correos electrónicos incorporen la foto del remitente, a fin de permitir la identificación visual.

2. La orientación objetos. Este paradigma de la programación ha terminado influyendo en la orientación de muchos SGBD, que siguen alguna de las dos líneas esbozadas a continuación:

- SGBD relacionales con objetos, los cuales admiten tipos abstractos de datos (TAD), además de los tipos tradicionales.
- SGBD orientados a objetos, que estructuran los datos en clases, las que comprenden tanto los datos (atributos) como las operaciones sobre ellas (métodos). Las clases se estructuran jerárquicamente, de tal manera que las de niveles inferiores (subclases) heredan las propiedades de las de niveles superiores (superclases).

3. Internet. Hoy en día, la mayoría de SGBD profesionales incorporan los recursos necesarios para apoyar a los servidores de páginas web dinámicas (es decir, con acceso a los datos contenidos en un SGBD alojado en el servidor web correspondiente).

Otra línea de innovación que siguen algunos SGBD es el trabajo con los llamados **almacenes de datos** (*data warehouse* , en inglés). Estos almacenes consisten en réplicas elaboradas de los datos generados por el funcionamiento cotidiano de la organización o la empresa de que se trate, durante un cierto periodo de tiempo para realizar análisis estratégicos de índole financiera, de mercados, etc.

El **lenguaje de marcas extensible** (*extensible markup language* , o XML, en inglés) también influye en el mundo de los SGBD, aunque inicialmente no se concibió como una tecnología para dar servicio a las BD, sino para estructurar documentos muy

grandes. Pero esta capacidad para almacenar los datos de que se compone un documento lo hacen susceptible de ser utilizado, también, en el ámbito de las BD.

Hay dos maneras de incorporar la tecnología XML en los SGBD:

- Los **SGBD relacionales con soporte para XML** , que en el fondo siguen siendo relacionales y que, por tanto, almacenan toda la información de manera tabular. Su utilidad principal es que los datos que devuelven las consultas sobre la BD pueden estar expresadas en formato XML, si así se pide al sistema gestor.
- Los **sistemas gestores para BD nativas XML** , que no son en realidad relacionales, sino más bien jerárquicos, y que no sólo están en condiciones de ofrecer los resultados de las consultas en formato XML, sino que también almacenan la información en documentos XML. El lenguaje estándar de consultas para este tipo de SGBD llama XQuery.

XQuery

El XQuery es un lenguaje de consultas diseñado para consultar colecciones de datos XML, creado por el World Wide Web Consortium (conocido abreviadamente como W3C). El W3C es un consorcio internacional que produce estándares para la World Wide Web (conocida abreviadamente como WWW).

La utilización de datos estructurados mediante el estándar XML resulta especialmente interesante en el intercambio de información entre sistemas basados en plataformas poco compatibles entre ellas.

3.2. Objetivos y funcionalidades de los SGBD

Todos los SGBD del mercado quieren alcanzar una serie de objetivos y ofrecer una serie de funcionalidades, con más o menos acierto, que actualmente se consideran indispensables para el buen funcionamiento de cualquier sistema de información:

* Posibilitar las consultas no predefinidas de cualquier complejidad.

* Garantizar la independencia física y la independencia lógica de los datos.

* Evitar o solucionar los problemas derivados de la redundancia.

* Proteger la integridad de los datos.

* Permitir la concurrencia de usuarios.

* Contribuir a la seguridad de los datos.

1. POSIBILITAR LAS CONSULTAS NO PREDEFINIDAS DE CUALQUIER COMPLEJIDAD

Los usuarios autorizados de un SGBD deben poder plantear directamente al sistema cualquier consulta sobre los datos almacenados, de la complejidad que sea necesaria, respetando, eso sí, las reglas sintácticas para que la sentencia sea correcta.

A continuación, el SGBD debe ser capaz de responder él mismo a la consulta formulada, sin que sea necesario recurrir a ninguna aplicación externa.

Cuando no existían ni las BD ni los sistemas gestores, con el fin de obtener un subconjunto de los datos almacenados en un fichero, había dos posibilidades:

- Lanzar un listado secuencial de todos los datos y, a continuación, seleccionar manualmente las que interesaran.
- Escribir el código fuente de un programa específico para resolver la consulta en cuestión, compilarlo y ejecutarlo.

Los SGBD actuales, en cambio, están perfectamente capacitados para interpretar directamente las consultas, expresadas habitualmente como sentencias formuladas en el lenguaje de consulta SQL.

Evidentemente, esto no quiere decir que no se puedan seguir produciendo programas que incorporen consultas mediante las cuales acceden a BD. De hecho, esta es la opción más acertada y cómoda si se trata de consultas que se han de repetir a lo largo del tiempo.

2. GARANTIZAR LA INDEPENDENCIA FÍSICA Y LA INDEPENDENCIA LÓGICA DE LOS DATOS

Hay que garantizar la máxima independencia física de los datos respecto a los procesos usuarios, en general (es decir, tanto en cuanto a las consultas interpretadas por el SGBD como a los programas externos que acceden a la BD), de tal manera que se puedan llevar a cabo todo tipo de cambios tecnológicos de índole física para mejorar el rendimiento (como agregar o quitar un índice determinado), sin que ello implique tener que modificar ni las consultas a la BD ni las aplicaciones que acceden.

De manera similar, también es deseable la independencia lógica de los datos, la cual implica que las modificaciones en la descripción lógica de la BD (por ejemplo, añadir un nuevo atributo o suprimir otro) no deben impedir la ejecución normal de los procesos usuarios no afectados por ellas.

Y, con respecto a la independencia lógica de los datos, incluso puede interesar (y, de hecho, esta es una opción frecuente) que convivan diferentes visiones lógicas de una misma BD, en función de las características concretas de los diferentes usuarios o grupos de usuarios.

3. EVITAR O SOLUCIONAR LOS PROBLEMAS DERIVADOS DE LA REDUNDANCIA

Tradicionalmente, la repetición de los datos se ha considerado algo negativo, ya que conlleva un coste de almacenamiento innecesario. Hoy en día, sin embargo, esta característica, a pesar de ser cierta, no se tiene en cuenta, debido al abaratamiento de los discos duros y del aumento de su capacidad y rendimiento.

Pero hay otro aspecto a considerar que no ha perdido vigencia, y es el hecho de que la repetición de los datos es peligrosa, ya que cuando se actualizan pueden perder la integridad. Cuando se modifica el valor de un dato que está repetido, se deben modificar simultáneamente los valores de sus repeticiones para que se mantenga la coherencia entre todos.

Son datos íntegros los que se mantienen enteros y correctos.

La **redundancia** consiste en la repetición indeseada de los datos, que incrementa los riesgos de pérdida de integridad de las mismas cuando se actualizan.

Sin embargo, los SGBD deben permitir al diseñador de BD la definición de datos repetidos, ya que a veces (sobre todo en materia de fiabilidad, de disponibilidad o de costes de comunicación) es útil mantener ciertas réplicas de los datos.

Ahora bien, en todos estos casos, el objetivo del SGBD debe ser garantizar la actualización correcta de todos los datos allí donde estén duplicadas, de manera automática (es decir, sin que el usuario del SGBD haya encargarse de nada).

Otro tipo de duplicidad admisible es la que constituyen las llamadas **datos derivados** . Se trata de datos almacenados en la BD, que en realidad son el resultado de cálculos realizados con otros datos también presentes en la misma BD.

Los datos derivados también pueden ser aceptables, aunque conllevan una repetición evidente de algunos datos, si permiten realizar consultas de tipo global muy rápidamente, sin tener que acceder a todos los registros implicados.

Pero también aquí, el SGBD se encargará de actualizar debidamente los datos derivados en función de los cambios sufridos por los datos primitivas de las que dependen.

4. PROTEGER LA INTEGRIDAD DE LOS DATOS

Además de la redundancia, hay muchos otros motivos que pueden dañar la consistencia de los datos, tales como los siguientes:

- Los errores humanos.
- Las deficiencias en la implementación de los algoritmos de las aplicaciones.
- Las averías de los soportes físicos de almacenamiento.
- Las transacciones incompletas como consecuencia de las interrupciones del suministro eléctrico.

Los SGBD deben proteger la integridad de los datos en todos estos casos. Para ello disponen, por un lado, de las reglas de integridad, también llamadas *nadatriccions* , y de otra, los sistemas de restauración basados en copias de seguridad.

Mediante las **reglas de integridad**, el sistema valida automáticamente ciertas condiciones al producirse una actualización de datos, y lo autoriza si las cumple, o deniega el permiso de lo contrario.

Ejemplo de restricción del modelo

Un SGBD relacional nunca aceptará que una tabla almacene registros con una clave primaria idéntica, porque entonces la clave no serviría para identificar inequívocamente los registros entre ellos.

Las reglas de integridad pueden ser de dos tipos:

- **Restricciones del modelo**. Son reglas inherentes al modelo de datos que utiliza el SGBD (como el modelo relacional). El sistema las incorpora predefinidas, y siempre se cumplen.
- **Restricciones del usuario**. Son reglas definidas por los usuarios (diseñadores y administradores, fundamentalmente) de la BD, que no incorpora *a priori* el SGBD, y que sirven para modelar aspectos específicos del mundo real.

Ejemplo de restricción del usuario

En una tabla que almacena los alumnos de un centro docente, se quiere evitar que hayan alumnos matriculados en ciclos formativos de grado superior que fueran menores de edad, ya que la normativa vigente no lo admite. Entonces, hay que establecer una restricción en este sentido para calcular si la diferencia entre la fecha del sistema en el momento de la matrícula y la fecha de nacimiento de cada alumno es igual o superior a los 18 años. En este caso, el sistema permitiría la matrícula, y en caso contrario la prohibiría.

Los SGBD también proporcionan herramientas para realizar periódicamente **copias de seguridad de los datos** (o *backups* , en inglés) que permiten restaurar los datos dañadas y devolverlas a un estado consistente.

Ahora bien, los valores restaurados se corresponderán con los que había en el momento de realizar la copia de seguridad, antes del incidente que origina la restauración, y por lo tanto nunca estarán del todo actualizados, aunque sean correctas.

5. PERMITIR LA CONCURRENCIA DE USUARIOS

Un objetivo fundamental de todo SGBD es posibilitar de manera eficiente el acceso simultáneo a la BD por parte de muchos usuarios. Además, esta necesidad ya no está circunscrita sólo a grandes compañías o administraciones públicas con muchos usuarios, sino que cada vez es más frecuente, debido a la expansión de Internet y el éxito de las páginas dinámicas, alojadas en servidores web que deben incorporar un SGBD.

La concurrencia de usuarios en una BD consiste en el acceso simultáneo a la BD por parte de más de un usuario.

Debemos considerar dos tipologías de accesos concurrentes, con problemáticas bien diferenciadas:

- **Accesos de consulta de datos**. Pueden provocar problemas de rendimiento, derivados sobre todo de las limitaciones de los soportes físicos disponibles (por ejemplo, si la lentitud de giro del disco duro que contiene la BD, o del movimiento del brazo que lleva incorporado el cabezal, no permiten atender debidamente todas las peticiones de acceso que recibe el sistema).
- **Accesos de modificación de datos**. Las peticiones simultáneas de actualización de unos mismos datos pueden originar problemas de interferencia que tengan como consecuencia la obtención de datos erróneos y la pérdida de integridad de la BD.

Para tratar correctamente los problemas derivados de la concurrencia de usuarios, los SGBD utilizan fundamentalmente dos técnicas: las **transacciones** y los **bloqueos**.

Una **transacción** consiste en un conjunto de operaciones simples que deben ejecutarse como una unidad.

Las operaciones incluidas dentro de una transacción nunca se han de ejecutar parcialmente. Si por algún motivo no se han podido ejecutar todas correctamente, el SGBD debe deshacerse automáticamente los cambios producidos hasta entonces. De

este modo, se podrá volver a lanzar la ejecución de la misma transacción, sin tener que hacer ninguna modificación en el código de las diferentes operaciones que incluya.

COMMIT y ROLLBACK

La instrucción COMMIT indica, al SGBD, que un conjunto de operaciones determinado se debe ejecutar de manera transaccional. La operación ROLLBACK deshace los cambios producidos en caso de que las operaciones de una transacción se hayan ejecutado parcialmente.

Ejemplo de transacción

Imaginemos que el convenio colectivo de una empresa determina que los salarios de los trabajadores hay que subir el mes de enero un 3%. La mejor opción consistirá en actualizar la tabla de empleados y, más concretamente, los valores del campo que recoge los sueldos de los empleados, mediante una consulta de actualización que modifique todos estos datos y las incremente en un 3%.

Si queremos garantizar que la actualización del salarios no quede a medias, tendremos que hacer que todas las instrucciones que implique este proceso se comporten de manera transaccional, es decir, que se ejecuten todas o bien que no se ejecute ninguna.

Pero también se puede dar la situación en que diferentes transacciones quieran acceder a la BD simultáneamente. En estos casos, aunque cada transacción, individualmente considerada sea correcta, no se podría garantizar la consistencia de los datos si no fuera por el uso de la técnica del bloqueo.

Un **bloqueo** consiste en impedir el acceso a determinados datos durante el tiempo en que sean utilizadas para una transacción. Así se consigue que las transacciones se ejecuten como si estuvieran aisladas, de tal manera que no se producen interferencias entre ellas.

Ejemplo de bloqueo

Imaginemos que el departamento de recursos humanos de la empresa del ejemplo anterior dispone de una aplicación que le proporciona ciertos datos de carácter estadístico sobre las remuneraciones de los empleados, tales como el salario medio.

Si se quiere ejecutar de manera concurrente la transacción descrita, que incrementa los sueldos en un 3%, y al mismo tiempo se lanza la ejecución de la aplicación que calcula el salario medio de todos los empleados de la empresa, el resultado obtenido por este programa probablemente será erróneo.

Para garantizar la corrección del cálculo, se deberá bloquear una de las dos transacciones mientras la otra se ejecuta.

Si se bloquea la actualización de datos, el salario medio estará calculado a partir de los sueldos antiguos (es decir, antes de ser actualizados).

En cambio, si se bloquea el software estadístico, en primer lugar se actualizarán todos los datos, y luego se calcularán los resultados a partir de los nuevos valores del campo que almacene el salario.

Los bloqueos provocan esperas y retenciones, y por eso las nuevas versiones de los diferentes SGBD del mercado se esfuerzan en minimizar estos efectos negativos.

6. CONTRIBUIR A LA SEGURIDAD DE LOS DATOS

Ejemplo de datos confidenciales

Resulta evidente la necesidad de restringir el acceso a los secretos militares o, incluso, comerciales (como los datos contables). Pero también hay que respetar la privacidad, incluso por imperativo legal, en otras vertientes aparentemente más modestos, pero en realidad no menos importantes, como son los datos personales.

La expresión seguridad de los datos hace referencia a su confidencialidad. A menudo, el acceso a los datos no debe ser libre o, al menos, no lo debe ser totalmente.

Los SGBD deben permitir definir autorizaciones de acceso a las BD, estableciendo permisos diferentes en función de las características del usuario o del grupo de usuarios.

Actualmente, los SGBD permiten definir autorizaciones a diferentes niveles:

- Nivel global de toda la BD
- Nivel de entidad
- Nivel de atributo
- Nivel de tipo de operación

Ejemplos de derechos de acceso

Los usuarios del departamento de contabilidad puede que no deberían tener acceso a la entidad que almacena los datos personales de los empleados de la empresa, a diferencia del usuario que ostenta el cargo de director general, que las podrá consultar el fin de optimizar la ubicación de los trabajadores el organigrama en función del respectivo perfil, o también de los usuarios del departamento de recursos humanos, que deberían poder incluso modificarlas.

En general, los usuarios no deberían tener acceso a los atributos que almacenan la dirección particular de los empleados, a menos que se trate del personal de recepción, si resulta que es el encargado de enviarlos cierta correspondencia a domicilio (como ahora las nóminas o los certificados de retenciones de IRPF), y por tanto debería, al menos, de poder consultarlos.

Estos mecanismos de seguridad requieren que cada usuario se pueda identificar. Lo más frecuente es utilizar un nombre de usuario y una contraseña asociada para cada usuario. Pero también hay quien utiliza mecanismos adicionales, tales como tarjetas magnéticas o de reconocimiento de la voz. Actualmente, se investiga en otras direcciones como, por ejemplo, en la identificación personal mediante el reconocimiento de las huellas dactilares.

Otro aspecto a tener en cuenta al hablar de la seguridad de los datos es su encriptación. Muchos SGBD ofrecen esta posibilidad, en alguna medida.

Las **técnicas de encriptación** permiten almacenar la información utilizando códigos secretos que no permiten acceder a los datos a personas no autorizadas y que, por tanto, no disponen de los códigos mencionados.

La encriptación puede hacer disminuir el rendimiento en el acceso a los datos, ya que conlleva la utilización de algoritmos adicionales en las operaciones de consulta. Por ello se ha de dosificar el uso. Ahora bien, siempre que sea posible, es conveniente encriptar las contraseñas.

3.3. Lenguajes de SGBD

La comunicación entre los SGBD y sus usuarios se realizará mediante algún tipo de lenguaje. Los lenguajes de BD se pueden clasificar en dos grandes tipologías según la finalidad:

- **Lenguajes de definición de datos** (*data definition languages* , en inglés, o DDL). Están especializados en la definición de la estructura de las BD mediante la especificación de esquemas.
- **Lenguajes de manipulación de datos** (*data management languages* , en inglés, o DML). Posibilitan la consulta, modificación y eliminación, de los datos almacenados, así como la inserción de nuevas informaciones. Podemos considerar la existencia de dos subtipos, básicamente:
  - **Procedimentales** . Requieren especificar no sólo los datos que se necesitan, sino también los procedimientos que se deben seguir para obtenerlas. Se utilizaban de manera exclusiva antes de la llegada del modelo relacional. Actualmente, se siguen utilizando, pero sólo cuando es necesario optimizar algún proceso que no rinde lo suficiente, por estar implementado de manera declarativa. Son más eficientes que los declarativos, pero más complicados, ya que exigen tener ciertos conocimientos sobre el funcionamiento interno del SGBD utilizado.
  - **Declarativos** . Sólo requieren especificar qué datos se necesitan, sin necesidad de especificar cómo se han de obtener. Son más sencillos de aprender que los procedimentales, pero también menos eficientes.

El lenguaje más utilizado para interaccionar con los SGBD relacionales es el **SQL.**

El SQL engloba las dos tipologías de lenguajes de BD descritas. Sus operaciones se pueden clasificar, pues, en uno de los dos tipos mencionados (DDL y DML) con fines pedagógicos, pero en realidad todas forman parte de un único lenguaje.

Ejemplos de operaciones DDL y DML del lenguaje SQL

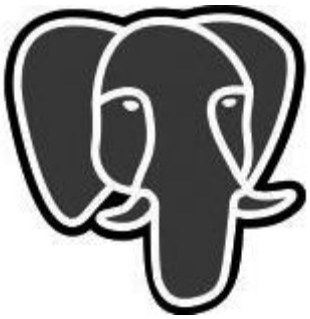
Como instrucciones de tipo DML, podemos mencionar SELECT (para hacer consultas), y también INSERT, UPDATE y DELETE (para realizar el mantenimiento de los datos).

Y como instrucciones de tipo DDL, podemos considerar CREATE TABLE (que nos permite definir las tablas, sus columnas y las restricciones que proceda).

En relación al componente DML de SQL, hay que decir que es fundamentalmente declarativo, aunque tiene posibilidades procedimentales, que se pueden explotar en diferentes SGBD:

PostgreSQL *es un SGBD distribuido con licencia BSD (Berkeley Software Distribution)*

- **PL / SQL** , lenguaje procedimental para trabajar con los SGBD creados por Oracle.
- **PL / pgSQL** , similar al PL / SQL de Oracle, pero diseñado para trabajar con PostgreSQL.



Imágen 1 Logotipo de PostgreSQL

Hay que decir que, además del respectivo lenguaje nativo de BD (habitualmente, SQL), los SGBD ofrecen, desde hace ya mucho tiempo, dos posibilidades más para incrementar la productividad en el trabajo con BD, que son los **lenguajes de cuarta generación** y las interfaces visuales, a menudo proporcionados dentro del entorno de una sola herramienta.

A menudo, el acceso a las BD también se hace desde aplicaciones externas al SGBD, escritas en lenguajes de programación (como por ejemplo C, Java, etc.), los cuales normalmente no incorporan instrucciones propias que permitan la conexión con BD.

Para responder a esta necesidad, hay dos opciones:

- Realizar, dentro del programa, llamadas a diferentes funciones que son en librerías que implementan estándares de conectividad de BD con programas escritos en ciertos lenguajes, tales como los siguientes:
- **ODBC** (*open data base connectiVity*), sistema creado por Microsoft y compatible con muchos sistemas como, por ejemplo, Informix, MS Access, MySQL, Oracle, PostgreSQL, SQL Server, etc.
- **JDBC** (*Java fecha base connectiVity*), para realizar operaciones con BD desde aplicaciones escritas en Java.
- Hospedar las sentencias del lenguaje de BD que sean necesarias, dentro de un programa anfitrión escrito en el lenguaje de programación utilizado. Es imprescindible que el compilador utilizado acepte la introducción de sentencias escritas en el lenguaje de BD utilizado (que normalmente será el SQL).

3.4. Usuarios y administradores

Las personas que trabajan con SGBD se pueden clasificar como usuarios en sentido estricto, los cuales simplemente interactúan con el sistema (aunque de diferentes maneras y con diferentes finalidades), o bien como administradores, si además realizan tareas de gestión y control.

USUARIOS DE SGBD

Podemos diferenciar tres categorías de usuarios de SGBD en función de la manera en que interactúan con el sistema: externos, sofisticados y programadores de aplicaciones.

1. Usuarios externos . Son usuarios no sofisticados, que no interactúan directamente con el sistema, sino mediante alguna aplicación informática desarrollada previamente por otras personas con esta finalidad.

Ejemplo de usuario externo

Cualquier persona asume este rol cuando saca dinero de un cajero automático, ya que accede a la BD de la entidad financiera identificándose mediante una tarjeta magnética y un número de identificación personal secreto (*personal identification number* , en inglés, o PIN).

Una vez autorizada a entrar en el sistema, podrá realizar diferentes operaciones de consulta o, incluso, de actualización. En el caso planteado, tras sacar dinero, el saldo de la cuenta corriente asociada a la tarjeta sufrirá el decremento correspondiente.

herramientas CASE

Las herramientas CASE (acrónimo de computer aided software ingeneering , o ingeniería del software asistida por ordenador) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste del desarrollo en términos de tiempo y de dinero.

2. Usuarios Sofíaticados . Interactúan directamente con el sistema, sin utilizar las interfaces proporcionadas por programas intermediarios. Formulan las consultas en un lenguaje de BD (normalmente, SQL), desde dentro del entorno que el SGBD pone a su disposición. Tradicionalmente, este entorno ha sido una consola en la que se podían escribir las consultas, pero cada vez son más frecuentes entornos que permiten construir las consultas de modo visual, como auténticas herramientas CASE.

3. Desarrolladores de aplicacions . Son profesionales informáticos que crean los programas que acceden a los SGBD y que, posteriormente, son utilizados por los usuarios que hemos llamado *externos* . Estas aplicaciones se pueden desarrollar mediante diferentes lenguajes de programación y herramientas externas al SGBD. Pero muchos SGBD comerciales también incluyen entornos propios de desarrollo y lenguajes de cuarta generación que facilitan enormemente la generación de formularios e informes que permiten visualizar y modificar los datos.

ADMINISTRADORES DE SGBD

Los **administradores** son unos usuarios especiales que realizan tareas de administración y control centralizado de los datos, y gestionan los permisos de acceso concedidos a los diferentes usuarios y grupos de usuarios, para garantizar el funcionamiento correcto de la BD.

Los administradores deben actuar, evidentemente, para solucionar las eventuales paradas del sistema, pero su responsabilidad fundamental consiste, justamente, evitar que se produzcan incidentes.

El trabajo de los administradores no es fácil, aunque los SGBD incorporan cada vez más herramientas para facilitarla, y en la mayoría de los casos con interfaz visual. Se trata, por ejemplo, de herramientas de monitorización de rendimiento, de herramientas de

monitorización de seguridad, de verificadores de consistencia entre índices y datos, de gestores de copias de seguridad, etc.

Una lista no exhaustiva de las tareas de los administradores podría ser la siguiente:

- Crear y administrar los esquemas de la BD.
- Administrar la seguridad: autorizaciones de acceso, restricciones, etc.
- Realizar copias de seguridad periódicas.
- Controlar el espacio de disco disponible.
- Vigilar la integridad de los datos.
- Observar la evolución del rendimiento del sistema y determinar qué procesos consumen más recursos.
- Asesorar a los programadores y los usuarios sobre la utilización de la BD.
- Hacer cambios en el diseño físico para mejorar el rendimiento.
- Resolver emergencias.

3.5. Componentes funcionales de los SGBD

El software que conforma los SGBD se divide en diferentes **módulos** , encargados de las respectivas funcionalidades que debe garantizar el sistema.

El componentes funcionales de los SGBD más importantes son el gestor de almacenamiento y el procesador de consultas.

1. GESTOR DE ALMACENAMIENTO

Las BD corporativas tienen enormes requerimientos de espacio de almacenamiento en disco. Los datos se transfieren entre el disco en el que están almacenados y la memoria principal del ordenador sólo cuando es necesario. Y, como la transferencia de datos hacia el disco o desde el disco es lenta en comparación con la velocidad de la unidad central de procesamiento, es muy importante que el SGBD estructure los datos de tal forma que se minimice la necesidad de transferencia entre el disco y la memoria principal.

El gestor de almacenamiento proporciona la interfaz entre los datos, consideradas a bajo nivel, y las consultas y los programas que acceden a la BD. Los datos se almacenan en disco utilizando el sistema de archivos que proporciona el sistema operativo utilizado. Y el gestor traduce las instrucciones DML a órdenes comprensibles por sistemas de archivos a bajo nivel.

Los componentes del gestor de almacenamiento son los siguientes:

- **Gestor de autorizaciones y de integridad**. Comprueba que se satisfagan tanto las restricciones de integridad como las autorizaciones de los usuarios para acceder a los datos.
- **Gestor de transacciones**. Asegura que la BD se mantenga en un estado de consistencia pesar de los fallos del sistema, y también que las transacciones concurrentes no se interfieran entre ellas.
- **Gestor de archivos**. Gestiona la reserva de espacio de almacenamiento en disco y las estructuras de datos utilizadas para representar la información almacenada en disco.
- **Gestor de memoria intermedia**. Transfiere los datos desde el disco a la memoria principal, y decide qué datos se deben tratar en caché.

Permite al sistema tratar con datos de tamaño muy superior a la de la memoria principal.

Por otra parte, el gestor de almacenamiento utiliza ciertas estructuras de datos que forman parte de la implementación física del sistema:

- **Archivos de datos**. Almacenan la BD propiamente considerada.
- **Diccionario de datos**. Almacena los metadatos relativos a toda la estructura de la BD.
- **Índices**. Proporcionan un acceso rápido a ciertos datos en función de sus valores.

2. PROCESADOR DE CONSULTAS

El procesador de consultas ayuda el SGBD simplificar el acceso a los datos. Las vistas a alto nivel contribuyen a alcanzar este objetivo, ya que evitan que los usuarios tengan que conocer detalles de la implementación física del sistema. Pero eso no quita que el sistema no tenga que perseguir la eficacia en el procesamiento de las consultas y de las actualizaciones de datos. De hecho, el sistema ha de traducir las instrucciones escritas, a nivel lógico, en un lenguaje no procedimental (típicamente, SQL), a una secuencia de operaciones a nivel físico, con unos mínimos de eficiencia.

Los componentes del procesador de consultas son los siguientes:

- **Intérprete DDL** . Interpreta las instrucciones de tipo DDL y registra las definiciones en el diccionario de datos.
- **Compilador DML** . Traduce las instrucciones DML formuladas en un lenguaje de consultas (normalmente, SQL) a una serie de instrucciones a bajo nivel que puede interpretar el motor de evaluación de consultas. Al realizar la traducción mencionada, un buen compilador DML también se encargará de hacer una optimización de consultas eligiendo, entre todas las alternativas, la de menor coste.
- **Motor de evaluación de consultes** . Ejecuta las instrucciones de bajo nivel generadas por el compilador DML.

La figura 8 muestra todos estos componentes y las conexiones entre ellos.

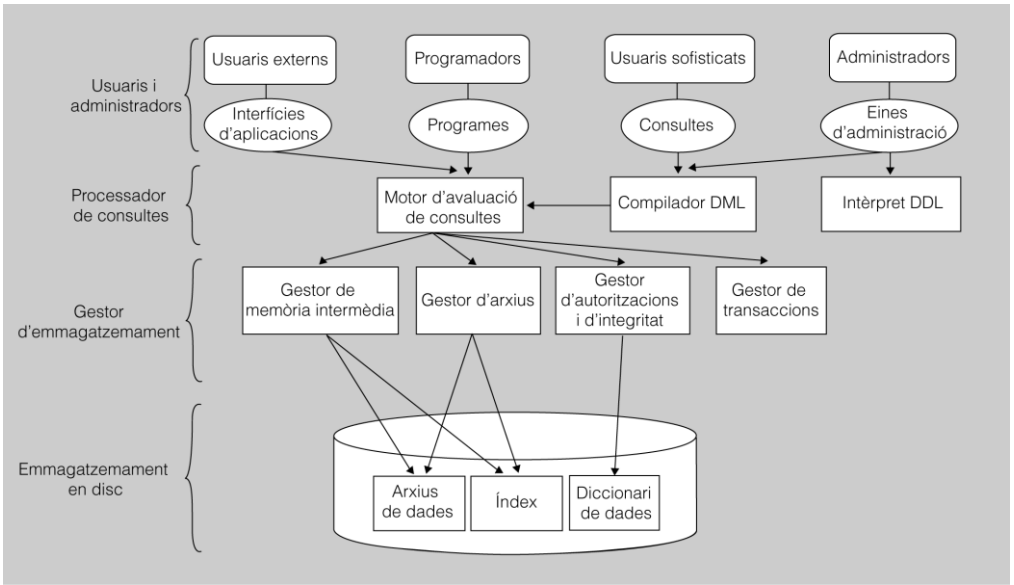


Figura 8 Componentes funcionales de los SGBD

3.6. Diccionario de datos

Un diccionario de datos de una base de datos es el conjunto de metadatos que proporcionan información sobre el contenido y la organización de la base de datos.

Un **diccionario de datos** , según *IBM Dictionary of Computing* se puede definir como un "repositorio centralizado de información sobre datos como el significado, relaciones con otros datos, origen, uso y formato."

A veces el concepto de diccionario de datos o metadatos también es conocido con el nombre de catálogo del sistema o, también, como repositorio de metadatos.

Los diferentes SGBD específicos implementan de diversas formas el diccionario de datos, de forma que cada software implementa con un conjunto de tablas y ofrece un conjunto de vistas a diferentes tipos de usuarios del sistema. Así pues, normalmente, un usuario con privilegios de administrador del sistema (DBA) puede visualizar más información y de tipo más específico que un usuario sin estos privilegios.

DBA

DBA es el acrónimo de ***Data Base Administrator*** o Administrador de Bases de Datos, que es la persona encargada de gestionar las BD, dentro del SGBD.

Los elementos que se encuentran habitualmente en un diccionario de datos incluyen:

- Definiciones del esquema de la base de datos.
- Descripciones detalladas de tablas y campos, así como de todos los objetos de la base de datos (vistas, clusters, índices, sinónimos, funciones y procedimientos, triggers, etc.).
- Restricciones de integridad referencial.
- Información de control de acceso, tales como nombres de usuario, roles, y privilegios.
- Parámetros de ubicación del almacenamiento.
- Estadísticas de uso de la base de datos.

Todo este conjunto de información, se almacena en cientos de tablas de información. Aunque hay organismos que intentan estandarizar la organización de esta metainformación, la realidad es que cada distribuidor de software específico (cada SGBD) ofrece su propia estructura.

Un administrador del sistema o DBA deberá conocer cómo acceder y consultar toda esta información consultando la guía de referencia del sistema con el que trabaje.

EL DICCIONARIO DE DATOS EN ORACLE

En Oracle es el usuario `SYS es` el propietario del diccionario de datos y lo único que puede hacer actualizaciones sobre la información que contiene. Aunque, alterar o manipular el diccionario de datos es una operación de administración que hay que hacer con muchas precauciones, pues puede provocar daños permanentes.

En Oracle se crean tres tipos de vistas diferentes, que permiten consultar información sobre diferentes tipos de datos. Así las vistas pueden tener uno de estos prefijos:

USER: Para visualizar información sobre los objetos propiedad del usuario.

ALL: Para consultar información sobre los objetos donde tiene acceso el usuario.

DBA: Que da acceso a todos los objetos del sistema, para poder ser controlados y gestionados.

Así pues, por ejemplo, se pueden consultar el conjunto de objetos a que se tiene acceso desde un usuario dado de Oracle con la siguiente sentencia:

SELECT owner, object_name, object_type FROM ALL_OBJECTS;

DICCIONARIO DE DATOS EN MYSQL

En MySql, en cambio, es la vista `INFORMATION_SCHEMA` que proporciona información sobre las bases de datos que se almacenan en el sistema.

Para obtener cierta información sobre una base de datos concreta llamada `db_name` en un SGBD MySQL podríamos ejecutar una sentencia como:

SELECT table_name, table_type, engine FROM information_schema.tables
WHERE table_schema = 'db_name';