

# Introducción a los sistemas de gestión de bases de datos

Rafael Camps Paré

P03/75053/02083



# Índice


<b>Introducción</b>	5
<b>Objetivos</b>	5
<b>1. Funcionalidad y objetivos de los SGBD</b>	7
1.1. Consultas no predefinidas y complejas	7
1.2. Flexibilidad e independencia	7
1.3. Integridad de los datos	8
1.4. Concurrencia de usuarios	9
1.5. Seguridad	9
1.6. Otros objetivos	10
<b>2. Arquitectura de tres niveles de los SGBD</b>	11
2.1. Esquemas y niveles	11
2.2. Independencia de los datos	13
<b>3. Modelos de BD</b>	14
<b>4. Modalidades de uso del lenguaje</b>	16
<b>5. Arquitectura externa e interna</b>	18
<b>6. Los SGBD comerciales</b>	21
<b>Resumen</b>	23
<b>Actividades</b>	25
<b>Ejercicios de autoevaluación</b>	25
<b>Solucionario</b>	26
<b>Glosario</b>	26
<b>Bibliografía</b>	27



# Introducción a los sistemas de gestión de bases de datos

## Introducción

Al empezar esta asignatura partimos del supuesto de que el estudiante ya sabe qué son los SGBD y conoce sus características principales. Estos contenidos se desarrollaron en la asignatura *Bases de datos I*. En este módulo didáctico hacemos una breve introducción a los SGBD, revisamos brevemente algunos de los temas que ya explicamos con detalle e introducimos nuevos.



Los conceptos que se repasan en este módulo los encontraréis en el módulo "Introducción a las bases de datos" de la asignatura *Bases de datos I*.

## Objetivos

Como es preciso que el estudiante recuerde y complemente los conocimientos generales que adquirió en la asignatura *Bases de datos I*, en los materiales didácticos que se facilitan en este módulo encontrará las herramientas indispensables para alcanzar los objetivos siguientes:

1. Conocer los objetivos principales de los SGBD.
2. Distinguir entre los modelos de BD más utilizados.
3. Reconocer las ventajas que ofrecen las diferentes modalidades de uso del lenguaje SQL.
4. Saber identificar los rasgos principales de la arquitectura de los SGBD del mercado.
5. Poder distinguir los componentes más habituales de un SGBD.
6. Identificar las tendencias de los SGBD actuales.



## 1. Funcionalidad y objetivos de los SGBD

Una **BD** es un conjunto estructurado de datos que representa entidades y sus interrelaciones. Esta representación informática integrada tiene que poder ser consultada y actualizada de forma compartida por muchos usuarios de tipos diversos. El **SGBD** es el *software* especializado que facilita su utilización, un *software* complejo que actualmente constituye una parte esencial de todo sistema de información (SI).

A continuación recordaremos los objetivos y las funcionalidades de los SGBD.

### 1.1. Consultas no predefinidas y complejas

Los usuarios pueden hacer consultas de cualquier tipo y complejidad directamente al SGBD. Éste tiene que responder inmediatamente sin que estén preestablecidas, es decir, sin que se tenga que escribir, compilar y ejecutar un programa específico para cada consulta.

El usuario tiene que poder formular la consulta en un lenguaje y el sistema tiene que interpretarlo directamente. Pero también se tiene que poder escribir programas con consultas incorporadas. El lenguaje que se utiliza universalmente para alcanzar este doble objetivo (consultas no predefinidas y complejas) es el lenguaje SQL, que ya conocéis.

Podéis ver el lenguaje SQL en el módulo "El lenguaje SQL" de la asignatura *Bases de datos I*.



### 1.2. Flexibilidad e independencia

Interesa obtener la máxima independencia entre los datos y los procesos usuarios. Es necesario que se pueda hacer todo tipo de cambios tecnológicos y cambios en la descripción de la BD sin que se tengan que modificar los programas de aplicación ya escritos ni variar la manera de hacer las consultas directas.

Decimos que hay **independencia física de los datos** cuando los usuarios hacen consultas o actualizaciones en la base de datos (tanto si son usuarios directos como si son programadores de aplicaciones) no necesitan conocer las características físicas de la BD con la que trabajan: soporte físico, SO utilizado,

índices, etc. Gracias a la independencia física podemos hacer cambios tecnológicos y físicos para mejorar el rendimiento sin afectar a ningún usuario.

También pretendemos que los usuarios no tengan que hacer cambios cuando se modifica la descripción lógica o el esquema de la BD; por ejemplo, cuando se le añade o suprimen entidades o interrelaciones, atributos, etc. Y aún más: queremos que los diferentes procesos usuarios puedan tener diferentes visiones lógicas de una misma BD y que estas visiones se puedan mantener lo más independientes posible de la BD y entre sí. Este tipo de independencia, que da flexibilidad y elasticidad en los cambios lógicos, se llama **independencia lógica de los datos**.

### 1.3. Integridad de los datos

Los SGBD tendrán que asegurar el mantenimiento de la calidad de los datos en cualquier circunstancia.

Cuando tenemos redundancia en la BD, las actualizaciones pueden provocar inconsistencias o incoherencias entre los datos. Convendrá, pues, evitarla. Pero a menudo nos interesa tenerla por razones de fiabilidad, disponibilidad, rendimiento o costes de comunicaciones. Por eso, los SGBD tienen que permitir que el diseñador defina datos redundantes, pero entonces tendrá que ser el propio SGBD el que haga automáticamente la actualización de los datos en todos los sitios donde estén repetidos.

La redundancia no es la única causa posible de pérdida de integridad de los datos. La corrección o la consistencia de los datos se puede perder por muchas otras razones: errores de programas, errores de operación humana, avería del disco, transacciones incompletas por corte de la alimentación eléctrica, etc.

Podremos declarar para el SGBD un conjunto de **reglas de integridad**, o restricciones, para garantizar que los programas mantendrán la calidad de los datos.

Cuando el SGBD detecta que un programa, o un usuario, intenta hacer una operación que va contra las reglas declaradas al definir la BD, no se lo tiene que permitir, y tiene que devolverle un estado de error. Al diseñar una BD para un SI concreto y escribir su esquema no definiremos solamente los datos, sino también las reglas de integridad.

El concepto de integridad de los datos va más allá de la prevención de que los programas o los usuarios almacenen datos incorrectos. En casos de errores o desastres también podríamos perder la integridad de los datos. El SGBD nos tiene que dar las herramientas para poder reconstruir o restaurar los datos dañados.



Los **procesos de restauración\*** de que todo SGBD dispone pueden reconstruir la BD hasta llegar a un estado consistente, correcto, anterior al incidente. Eso se acostumbra a hacer gracias a la obtención de copias periódicas de los datos, que se suelen llamar **copias de seguridad\*\***, y al mantenimiento continuo de un **diario\*\*\*** en el cual el SGBD anota todos los cambios que se hacen en la BD.

\* En inglés, *restore o recovery*.

\*\* En inglés, *backups*.

\*\*\* En inglés, *log*.

#### 1.4. Concurrencia de usuarios

Un objetivo fundamental de los SGBD es permitir que un número elevado de usuarios pueda acceder concurrentemente a la misma BD.

Cuando los accesos concurrentes son todos de lectura, nos enfrentamos simplemente a un problema de rendimiento causado por las limitaciones físicas de los soportes de que se dispone. Pero si uno o más de un usuario actualizan los datos, se pueden producir **problemas de interferencia**, los cuales pueden dar lugar a la obtención de datos erróneos y a la pérdida de integridad de la BD. Para tratar los accesos concurrentes, los SGBD utilizan el concepto de *transacción*.

Una **transacción** consiste en un conjunto de operaciones simples que se ejecutan como una unidad indivisible. Los SGBD tienen que conseguir que el conjunto de operaciones de una transacción nunca se ejecute parcialmente: o se ejecutan todas o no se ejecuta ninguna.

Entre las transacciones que se ejecutan concurrentemente se pueden producir problemas de interferencia que produzcan resultados erróneos o que motiven la pérdida de la integridad de los datos. Para evitarlo, se pide que el SGBD trate cada transacción como un proceso aislado de los demás.

#### 1.5. Seguridad

En el campo de los SGBD, el término *seguridad* se acostumbra a utilizar para hacer referencia a los temas relativos a la confidencialidad, la privacidad, las autorizaciones, los derechos de acceso, etc.

Los SGBD permiten definir autorizaciones o derechos de acceso por categorías: la global de toda la BD, la de entidad e incluso la de atributo. Estos mecanismos de seguridad se basan en la identificación del usuario, que se efectúa mediante los códigos de usuario (y grupos de usuarios) acompañados de contraseñas\*.

\* En inglés, *passwords*.

Para tener un grado de seguridad más elevado cuando se quiere mantener los datos en secreto, muchos de los SGBD actuales tienen prevista la encriptación de los datos que almacenan\*.

\* En inglés, *encryption*.

## 1.6. Otros objetivos

A medida que los SGBD evolucionan, los diseñadores se plantean nuevos objetivos para adaptarlos a nuevas necesidades y a nuevas tecnologías. En estos momentos podríamos citar como objetivos más recientes los siguientes: !

! Podéis ver las tendencias actuales en SGBD en el subapartado 2.4 del módulo "Introducción a las bases de datos" de la asignatura *Bases de datos I*.

- 1) Servir eficientemente a los *data warehouse*.
- 2) Adaptarse al desarrollo orientado a objetos.
- 3) Incorporar el tiempo como un elemento de caracterización de la información.
- 4) Adaptarse al mundo de Internet.

## 2. Arquitectura de tres niveles de los SGBD

Como ya conocéis la arquitectura ANSI/SPARC de tres niveles, ahora sólo recordamos brevemente sus aspectos más destacados.

### 2.1. Esquemas y niveles


Los SGBD necesitan que les demos una descripción o definición de la BD. Esta descripción recibe el nombre de **esquema de la BD**.

El esquema de la BD es un elemento fundamental de la arquitectura de un SGBD que permite independizar el SGBD de la BD. Por ejemplo, nos permite cambiar el diseño de la BD, el esquema, sin tener que introducir ningún cambio en el SGBD.

La distinción clásica entre dos niveles de abstracción, el nivel lógico y el nivel físico, es ampliada por la arquitectura ANSI/SPARC a tres niveles de abstracción, tres niveles de esquemas.

Podéis ver los niveles lógicos y físicos en el subapartado 3.7 del módulo "Los datos: conceptos introductorios" de la asignatura *Bases de datos I*.

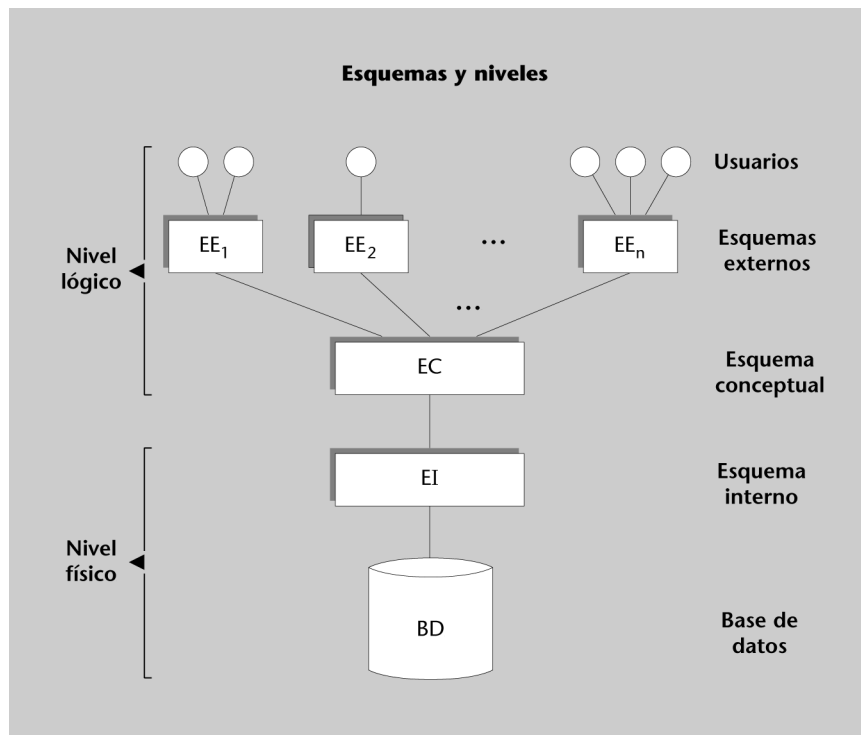
La idea básica de la arquitectura ANSI/SPARC consiste en descomponer el nivel lógico en dos: el **nivel externo** y el **nivel conceptual**. En esta arquitectura llamamos **nivel interno** a lo que habíamos llamado *nivel físico*.

Así pues, de acuerdo con la arquitectura ANSI/SPARC, existirían los tres niveles de esquemas que podéis ver en el gráfico de la página siguiente: 

a) En el nivel conceptual hay una sola descripción lógica básica, única y global, que llamamos **esquema conceptual**, y que sirve de referencia para el resto de esquemas.

En el esquema conceptual se describen las entidades tipo, sus atributos, las interrelaciones y también las restricciones o reglas de integridad.

b) En el nivel externo se sitúan las diferentes visiones lógicas que los diferentes procesos usuarios tendrán de las partes de la BD que utilizarán. Estas visiones se llaman **esquemas externos**.



Cada aplicación podrá tener su visión particular, y seguramente parcial, del esquema conceptual: los usuarios verán la BD por medio de esquemas externos apropiados a sus necesidades. Estos esquemas se pueden considerar redefiniciones del esquema conceptual, con las partes y los términos que convengan para las necesidades de las aplicaciones (o grupos de aplicaciones). Algunos sistemas los llaman *subesquemas*.

Los usuarios pueden ser programas o usuarios directos.

Al definir un esquema externo se citarán sólo aquellos atributos y entidades que interesen. Podremos cambiar su nombre, definir datos derivados, redefinir una entidad para que las aplicaciones que usan este esquema externo creen que son dos o definir combinaciones de entidades para que parezcan una sola, etc.

c) En el nivel físico hay una descripción física única, que llamamos **esquema interno**.

El esquema interno o físico contendrá la descripción de la organización física de la BD: caminos de acceso (índices, *hashing*, apuntadores, etc.), codificación de los datos, gestión del espacio, tamaño de la página, etc.

### Esquemas y niveles en los SGBD relacionales

En los SGBD relacionales; es decir, en el mundo de SQL, se ha fijado una terminología ligeramente diferente de lo habitual. No se separan claramente los tres niveles de descripción. Se habla de un solo **esquema** (*schema*), pero dentro de éste se incluyen descripciones de los tres niveles. En el *schema* se describen los elementos de aquello que en la arquitectura

ANSI/SPARC se llama *esquema conceptual* (entidades tipo, atributos y restricciones) más las **vistas** (*views*), que corresponden aproximadamente a los esquemas externos.

El modelo relacional en el que se inspira el SQL se limita a moverse en el mundo lógico. Por eso el estándar ANSI-ISO del SQL no habla en absoluto del mundo físico o interno, sino que lo deja en manos de los SGBD relacionales del mercado. Pero éstos ofrecen la posibilidad de incluir en el *schema* descripciones de estructuras y características físicas (índice, *tablespace*, *cluster*, espacios para excedentes, etc.).

## 2.2. Independencia de los datos

La arquitectura de tres niveles nos proporciona los dos tipos de independencia de los datos: la independencia física y la independencia lógica. El cambio de datos de un soporte a otro, o de sitio dentro de un soporte, no afecta a los programas de aplicación ni a los usuarios directos porque no modifica el esquema conceptual ni los esquemas externos, como tampoco les tendría que afectar el cambio del método de acceso a unos registros determinados\*. El cambio de formato o de codificación tendría que afectar a la BD física y al esquema interno, pero no al mundo exterior.

Podéis ver la independencia física y la independencia lógica en el subapartado 1.2 de este módulo didáctico.

\* Por ejemplo, eliminando un índice en árbol-B o sustituyéndolo por un *hashing*.

Si hay independencia física de los datos, la única cosa que cambia al modificar el esquema interno son las correspondencias entre el esquema conceptual y el interno. Obviamente, la mayoría de los cambios del esquema interno obligan a rehacer la BD real (física).

Con relación a la independencia lógica, como en la arquitectura ANSI/SPARC hay dos niveles lógicos diferentes, distinguimos las dos situaciones siguientes: !

**a) Cambios en el esquema conceptual:** este tipo de cambios no afectará a los esquemas externos que no hagan referencia a las entidades o a los atributos modificados.

**b) Cambios en los esquemas externos:** efectuar un cambio en un esquema externo afectará a los usuarios que utilicen los elementos modificados, pero no tendría que afectar a los otros ni al esquema conceptual y, en consecuencia, tampoco debería incidir en el esquema interno ni en la BD física.

### 3. Modelos de BD

Como ya sabéis, el componente fundamental para modelizar en un SGBD relacional son las tablas o relaciones, pero hay ciertos tipos de SGBD que utilizan otros componentes.

El conjunto de componentes, o herramientas conceptuales, que un SGBD proporciona para modelizar recibe el nombre de **modelo de BD**.

Podéis ver el modelo relacional en el módulo "El modelo relacional y el álgebra relacional" de la asignatura *Bases de datos I*.

El **modelo relacional**, que ya conocéis, es el que se utiliza habitualmente. Antes del modelo relacional aparecieron los modelos jerárquico y en red. Así como en el modelo relacional hay un solo elemento estructural, la tabla, que está formada por filas y columnas, en estos modelos prerelacionales hay dos elementos básicos: los registros y las interrelaciones.

En las BD prerelacionales, las interrelaciones entre los registros estaban representadas por punteros (más o menos físicos) y el programador utilizaba estas interrelaciones explícitamente. Este tipo de programación se llama *programación navegacional* porque el programa explicita los caminos (punteros) que quiere recorrer para obtener los conjuntos de datos que le interesan. En cambio, en las BD relacionales no se indica el camino, sino el elemento que se quiere obtener. Así, se puede decir que se trata de una *programación declarativa*.

#### Comparación entre los modelos prerelacionales y el modelo relacional

Analizamos, por ejemplo, una BD con información de los departamentos de una empresa y con información de los empleados de esta empresa. En un SGBD relacional que utilizara SQL para indicar que entre los departamentos y los empleados hay una interrelación *empleado\_del\_dept* pondríamos un atributo (clave foránea) en la tabla de empleados con el valor del código del departamento donde trabaja. Para mostrar el nombre del departamento de cada empleado tendríamos que hacer una combinación (*join*) declarando la condición siguiente:

```
dept.codigodept = empl.codigodept;
```

En una BD jerárquica o en red, en el esquema de la BD tendríamos que explicitar la interrelación *empleado\_del\_dept* y ponerle un nombre, y el programa usuario tendría que seguir para cada empleado el camino *empleado\_del\_dept* para encontrar el nombre del departamento donde trabaja.

La descripción de las interrelaciones, el esquema y el uso explícito de éstas por los programas hace que en los SGBD prerelacionales haya menos independencia de datos que en los sistemas relacionales. !

En el **modelo jerárquico** los registros están interrelacionados en forma de árboles. El SGBD clásico de este modelo es el IMS/DL1 de IBM.

En el **modelo en red** los registros no están limitados a ser “hijos” de un solo registro tipo. El comité CODASYL-DBTG propuso un estándar basado en este modelo y ya definía tres niveles de esquemas.

La diferencia más importante entre los modelos prerrelacionales y el modelo relacional es que éste se limita al nivel lógico y no hace absolutamente ninguna consideración sobre las representaciones físicas; es decir, nos da una independencia total física de los datos.

En estos últimos años se ha ido extendiendo el **modelo de BD relacional con objetos**. Se trata de una ampliación del modelo relacional al que se añade la posibilidad de que los tipos de datos sean tipos abstractos de datos (TAD) definibles por el usuario informático con una estructura tan compleja como se desee. Las instancias de un TAD son los objetos.

Los tipos de datos que se pueden definir en los SGBD relacionales de los años ochenta y noventa son muy limitados. La incorporación de multimedia –imagen y sonido– a los SI obliga a los SGBD relacionales a aceptar atributos de estos tipos. Pero en algunas aplicaciones no basta con incorporar tipos especializados en multimedia: también necesitan tipos complejos definibles por el desarrollador a la medida de la aplicación.

#### Ejemplo de tipo especializado multimedia

En la entidad *alumno* nos puede interesar tener un atributo *foto* tal que su valor sea una tira de bits muy larga, resultado de la digitalización de la fotografía del alumno.

En definitiva, se necesitan tipos abstractos de datos, TAD. Los SGBD más recientes incorporan esta posibilidad y ya se ha iniciado un amplio mercado de TAD predefinidos o librerías de clases.

Esto nos lleva a la **orientación a objetos**. El éxito de la OO a finales de los años ochenta en el desarrollo de *software* básico, en las aplicaciones al diseño en ingeniería, y en la construcción de interfaces gráficas de usuario, ha hecho que en los años noventa se haya extendido a casi todos los campos de la informática. En los SI se inicia también la adopción, tímida de momento, de la OO. La utilización de lenguajes como el C++ o el Java supone que actualmente los SGBD relacionales se adaptan a ellos con interfaces adecuadas. Esta incorporación de los objetos a los sistemas relacionales los acerca al paradigma de la OO. Los primeros SGBD relacionales que han ofrecido esta posibilidad han sido Oracle (versión 8), Informix (versión 9) y IBM/DB2/UDB (versión 5).

### Modelos de datos y modelos de BD

Lo que en este apartado hemos llamado *modelos de BD* se acostumbra a llamar *modelos de datos*, ya que permiten modelizar los datos. Pero hay modelos de datos que no son usados por los SGBD del mercado; como no disponen de operaciones, no se utilizan en las realizaciones, sino sólo durante los procesos de análisis y diseño. El más conocido de estos tipos de modelos de datos es el modelo entidad relación (*entity-relationship*), o modelo ER, que ya conocéis.

## 4. Modalidades de uso del lenguaje

El lenguaje SQL es el lenguaje universal de comunicación con los SGBD para crear, consultar y mantener las BD. Hasta ahora lo habéis utilizado de manera directa, interactiva, sin escribir ningún programa. EL SGBD dispone de un interpretador que analiza, verifica y ejecuta las instrucciones\* que le damos directamente desde un teclado. Pero ésta no es la modalidad más habitual de uso del lenguaje SQL. Lo más habitual será que queramos utilizar el SQL desde dentro de un programa.

\* Por ejemplo, `SELECT`, `CREATE TABLE`, etc.

Si queremos escribir un programa de aplicación que trabaje con BD, seguramente nos interesará utilizar nuestro lenguaje habitual de programación\*. Pero estos lenguajes normalmente no tienen instrucciones propias para las BD, por tanto, posiblemente deseemos poder utilizar el SQL desde dentro del programa. Podremos escoger entre las dos modalidades de SQL siguientes:

\* Lenguajes como Pascal, C, Cobol, Basic, Fortran, Java, etc.

1) Las **llamadas a funciones**: en el mercado hay librerías de funciones especializadas en pedir al servidor de BD la ejecución de instrucciones SQL\*. Sólo hay que incluir dentro del programa, escrito en el lenguaje habitual, llamadas a las funciones apropiadas. Normalmente a estas funciones se les da como parámetro la instrucción de SQL que se quiere ejecutar. Serán las funciones las que se encargarán de dar las instrucciones de SQL al SGBD en tiempo de ejecución.

\* Como las librerías ODBC o las JDBC.

2) El **SQL hospedado** consiste en incluir directamente las instrucciones del lenguaje SQL en nuestro programa. Pero eso exige disponer de un precompilador especializado que acepte dentro de nuestro lenguaje de programación habitual las instrucciones del SQL. Entonces se dice que el SQL se ha hospedado o incorporado\* a nuestro lenguaje de programación (Pascal, C, Cobol, Java, etc.), que hace de lenguaje anfitrión\*\*. Tenemos las dos variantes de SQL hospedado que presentamos a continuación:

\* En inglés, *embedded*.  
\*\* En inglés, *host*.

a) El **estático**: con esta variante el programador escribe explícitamente en su programa las instrucciones que quiere que se ejecuten.

b) El **dinámico**: con esta variante el programador puede escribir parcialmente las instrucciones que se tienen que ejecutar y dejar partes de ellas para ser completadas en tiempo de ejecución del programa. EL SQL hospedado dinámico permite, pues, escribir programas genéricos\* que dialogan con un usuario final para determinar qué tienen que hacer. Cuando se escribe el programa no se sabe exactamente cuáles serán las instrucciones que se tendrán que enviar al SGBD, y se concretarán en tiempo de ejecución, ya que dependen de lo que pida el usuario del programa. La modalidad de SQL hospedado estático presenta la ven-


\* Como las herramientas de acceso visual a BD.



taja de ahorrar tiempo de ejecución, ya que la verificación y la traducción se hacen en tiempo de compilación.

Fijémonos que la primera de las modalidades (llamadas a funciones) es dinámica, ya que el programa pasa las instrucciones SQL como el valor de un parámetro\* y, por lo tanto, las puede construir en tiempo de ejecución.

\* Por ejemplo, en forma de literal.

En otro módulo se estudian como podemos acceder a una BD desde un lenguaje de programación de propósito general. 

## 5. Arquitectura externa e interna

La **arquitectura general de los SGBD** está determinada por tres elementos básicos: el SGBD centralizado o cliente/servidor (C/S), los datos centralizados o distribuidos y el aprovechamiento o no de plataformas con paralelismo. A continuación explicamos en qué consisten estos elementos:

1) Como ya sabemos, los SGBD han evolucionado desde una arquitectura centralizada a una de tipo C/S. Recordamos que estas arquitecturas se caracterizan por las particularidades siguientes:

Podéis ver la evolución de los SGBD en el apartado 2 del módulo "Introducción a las bases de datos" de la asignatura *Bases de datos I*.

a) La **arquitectura centralizada** consta de un solo ordenador, que puede estar conectado a una red de terminales. En este ordenador se ejecuta el SGBD y reside la BD.

b) La **arquitectura C/S** tiene una parte del SGBD en los ordenadores clientes (la interfaz con el usuario), y el corazón del SGBD, llamado **motor**, se ejecuta en el servidor. Puede haber muchos servidores diferentes en una misma red.

2) La mayoría de los SGBD actuales permite que una transacción trate datos distribuidos; es decir, que pueda trabajar de forma coordinada con las BD locales de diferentes servidores viendo el conjunto como si fuera una única BD.

3) El motor puede estar más o menos preparado para aprovechar el paralelismo de ciertas plataformas. Prácticamente todos los SGBD del mercado son capaces de ejecutar con un solo procesador diversos procesos (transacciones y procesos del motor). Eso se hace utilizando técnicas multitarea; es decir, repartiendo el uso del procesador. Bastantes SGBD pueden aprovechar las plataformas multiprocesadoras para ejecutar unos cuantos procesos del motor y unas cuantas transacciones en paralelismo real. Pero hay pocos SGBD capaces de ir más lejos, de descomponer una consulta en un conjunto de procesos ejecutables en paralelo. Eso último es lo que hacen los llamados **SGBD paralelos**.


El informático usuario del SGBD lo ve como un conjunto de partes más o menos bien estructurado. La **arquitectura externa** o arquitectura aparente del SGBD acostumbra a estar bastante determinada por razones comerciales. Los fabricantes modifican el empaquetado o la presentación de sus productos\* de acuerdo con la política de ventas, las modas, etc., pero de hecho no hay grandes diferencias externas entre diferentes productos.

\* Por ejemplo, cambian los nombres de los diferentes componentes.

En cambio, la **arquitectura interna**; es decir, la manera como los SGBD se estructuran en partes, realmente es muy diversa. Cada SGBD estructura interna-

mente el conjunto de su funcionalidad de manera diferente. Es más, un mismo SGBD va cambiando la estructura interna en versiones sucesivas para mejorar el rendimiento y facilitar la adición de nuevas funcionalidades.


El motor del SGBD, aquello que siempre se ejecuta en el servidor, suele tener módulos, como mínimo, para las funciones siguientes: la interpretación del SQL, la optimización de consultas, la gestión de vistas, la gestión de memorias intermedias (*buffers*), la gestión del espacio de la memoria externa, los métodos de acceso (por ejemplo, los índices), la gestión de procesos (multitarea, paralelismo), la gestión de transacciones, disparadores y procedimientos almacenados, el control del acceso concurrente (por ejemplo, la gestión de bloqueos), el mantenimiento de dietarios de actualizaciones (*log*), la creación, restauración y recuperación de copias de seguridad (*backup*), y el control de la seguridad y privacidad.

La mayoría de las funciones que acabamos de exponer se estudian con detalle a lo largo de esta asignatura. 

Existen solapamientos entre algunas de las funciones de un SGBD (las más físicas o básicas) y las del SO. Para determinados aspectos como, por ejemplo, la gestión de memorias intermedias, la del espacio de disco, los controles de seguridad o la gestión de procesos, los SGBD pueden aprovechar (o complementar) la funcionalidad del SO o bien resolverlos directamente. Una razón para no aprovechar el SO es que el SGBD lo puede hacer más eficientemente, o para aumentar la confidencialidad.

#### Sin SO

Un caso extremo de invasión del terreno de los SO es el Oracle Appliance. Se trata de un SGBD que se vende incorporado a un *hardware* provisto de un pequeño subconjunto del núcleo de un SO. Se vende, pues, un equipo con un SGBD y SO.

El motor del SGBD está rodeado de toda una serie de programas complementarios que se suele considerar parte integrante del SGBD. Algunos de estos programas son herramientas que se ejecutan en máquinas cliente. Los más frecuentes son los que enumeramos a continuación: 

- Precompiladores para SQL hospedado en diferentes lenguajes de programación\*.
- Herramientas para participar en redes de servidores heterogéneos\*.
- Herramientas básicas de administración y herramientas de control y ajuste del rendimiento.
- Librerías de funciones especializadas para el acceso a BD\*.
- Herramientas visuales para acceder fácilmente a la BD sin saber gran cosa del SQL.

\* Como Cobol, C, PL/I, Java, etc.

\* Como pueden ser SGBD de marcas diferentes.

\* Por ejemplo, ODBC o JDBC.

Pero además del SGBD propiamente dicho, los proveedores ofrecen muchas herramientas complementarias, generalmente con precio separado. Algunas de estas herramientas necesitan que el motor incorpore una funcionalidad específica. Aparte de los proveedores de los SGBD, hay un gran número de proveedores de herramientas independientes. Veamos una lista típica, pero no exhaustiva, de tipos de estas herramientas:

- Herramientas complementarias para diseñar la BD\*.
- Lenguajes 4GL, visuales, para desarrollar aplicaciones con rapidez.
- Herramientas para administrar la BD\*.
- Herramientas para interoperar con otros sistemas del mercado.
- Herramientas para aplicaciones especiales. Por ejemplo, el procesamiento de texto libre y la recuperación de información documental, datos espaciales o el análisis y prospección de datos\*.
- Gestor de réplicas.
- Herramientas específicas para la informática móvil\*.
- *Software* intermediario (*middleware*) entre clientes y servidores para entornos transaccionales de gran volumen y criticidad\*.
- Herramientas para utilizar los SGBD como servidores de páginas web.

\* Por ejemplo, herramientas que ayudan a hacer el diagrama ER y su traducción a un esquema SQL.

\* Como la supervisión del rendimiento, la gestión de copias de seguridad, la gestión de usuarios, etc.

\* Como las aplicaciones OLAP, data warehouse o data mining.

\* Como ordenadores portátiles de conexión esporádica a la red.

\* Por ejemplo, monitores de transacciones.

## 6. Los SGBD comerciales

Ya conocéis a grandes rasgos la historia de la evolución de los SGBD a lo largo de los años. Recordad que la mayoría de los SGBD relacionales actuales aparecieron antes de 1985. Actualmente hay en el mercado más de un centenar de marcas diferentes de SGBD relacionales y unas cuantas más de otros tipos de SGBD (orientados a objetos, especializados en documentos, especializados en multimedia, especializados en gestión geográfica, etc.).

Aunque la funcionalidad básica de todos los productos relacionales es muy parecida y aunque todos se basan en el SQL, cada producto tiene sus particularidades y una arquitectura (interna y externa) diferente, que se adapta a los tipos de entornos a los que va enfocado.

Actualmente todos los SGBD del mercado están pensados para que se puedan integrar en entornos cliente-servidor, y tienen que dar servicio en situaciones de operación muy diversas.


Hay SGBD específicos para ser utilizados como herramienta monousuario sobre un pequeño ordenador personal. Este ordenador puede estar aislado o bien conectado como cliente en una red C/S. En este último caso, en el equipo cliente sólo tendríamos la BD local y el SGBD local, con el cual también podríamos acceder a los SGBD remotos.

Hay servidores de BD muy potentes para dar servicio a millares de usuarios simultáneamente. Estos SGBD, que a menudo son especializados en paralelismo, funcionan generalmente sobre SO propietarios\*.

Por razones de disponibilidad y precio nos puede interesar tener, en lugar de un servidor muy potente, todo un conjunto de servidores de menos potencia, quizás distribuidos por la red\*\*.

Muchos de los SGBD del mercado están pensados para poder funcionar en un gran abanico de entornos y potencias.

Estos tipos de SGBD se venden con diferentes presentaciones o versiones. Es frecuente disponer de tres versiones: una versión monousuario de uso personal, una multiusuario para entornos C/S y otra para grandes servidores multiprocesadores paralelos.

EL SGBD que los estudiantes utilizáis para las prácticas de esta asignatura es una versión monousuario del sistema Informix, que tiene versiones ideadas para funcionar en equipos de gran potencia con paralelismo masivo. 

### Los SGBD más conocidos

Los SGBD relacionales más conocidos son, sin ningún tipo de duda, los siguientes:

- Oracle.
- DB2 de IBM.
- Informix.
- SQL-Server de Microsoft.

Sin embargo hay muchos otros bastante conocidos como, por ejemplo, Adaptive Server de Sybase (antes se llamaba SQL Server y fue el origen del de Microsoft) o el OpenIngres de CA (antes se llamaba Ingres, y se le considera el padre de la mayoría de los sistemas actuales).

### Access

Un ejemplo típico de SGBD utilizado como herramienta monousuario es el Access de Microsoft, aunque sería más correcto decir el Jet, que es el nombre del motor. El Access es, de hecho, una herramienta de acceso a BD que puede trabajar con cualquier gestor de datos que tenga previsto el ODBC; es decir, prácticamente con todos.

\* Por ejemplo, DB2/MVS o Teradata de NCR.

\*\* Un ejemplo de producto para este tipo de entornos sería el SQL-Server de Microsoft.

### Por ejemplo,...

... los sistemas Oracle y Informix fueron concebidos inicialmente para plataformas de tipo medio, pero posteriormente han ido ampliando el alcance hacia arriba y hacia abajo en la gama de potencias.

Obviamente, hay una cierta relación entre el precio de un SGBD y su potencia. Los precios pueden oscilar entre algunos cientos de euros y unos cuantos cientos de millares.

Las **tendencias actuales de los SGBD relacionales del mercado** se pueden resumir en los puntos siguientes: 

- Facilidad de crecimiento mediante el funcionamiento con plataformas de paralelismo masivo.
- Funcionalidad pensada para llevar a cabo análisis masivos de datos\*.
- Ampliación de los tipos de datos y posibilidad que el usuario informático los pueda definir según sus necesidades.
- Incorporación al mundo de la orientación a objetos.
- Incorporación al mundo de Internet.
- Incorporación a las arquitecturas de objetos distribuidos\*.

\* Por ejemplo, *data warehousing* y *data mining*.

\* Como CORBA o COM++.

## Resumen

En este módulo introductorio hemos recordado brevemente algunos conceptos fundamentales de los SGBD:

- 1) Hemos revisado los objetivos de los SGBD actuales y algunas de las funcionalidades que nos dan para conseguirlos.
- 2) Hemos recordado el concepto de transacción y hemos visto cómo es utilizado para velar por la integridad de los datos.
- 3) También hemos recordado que la arquitectura de tres niveles aporta una gran flexibilidad a los cambios, tanto físicos como lógicos, y que los SGBD relacionales nos dan más independencia de los datos que los prerrelacionales.
- 4) Para acabar, hemos visto que la evolución actual de los SGBD relacionales los acerca al mundo de la orientación a objetos.

Hemos visto que el lenguaje SQL se puede utilizar de forma directa, interactiva, pero que también lo podemos utilizar desde un programa si disponemos de un precompilador que nos permita considerar el SQL como incorporado u hospedado en nuestro lenguaje de programación. Además, podemos trabajar con el SQL desde los programas mediante llamadas a librerías de funciones especializadas. Hemos distinguido también la modalidad de SQL estática de la dinámica.

Hemos hablado de la arquitectura de los SGBD; es decir, de la manera como están estructurados en componentes, y hemos visto que esta arquitectura tiene un aspecto interno y uno externo. Hemos citado los componentes típicos de un SGBD y las herramientas que habitualmente lo acompañan.

Finalmente, hemos dado algunas ideas básicas sobre los SGBD relacionales que se pueden encontrar ahora en el mercado, y sobre las tendencias que se observan en su evolución.





## Actividades

1. Obtened información técnico-comercial de algún fabricante de SGBD sobre las versiones nuevas de sus productos y tratad de reconocer los conceptos tratados en este módulo.
2. Redactad un informe breve (no más de cinco páginas) sobre el estado actual de los SGBD comerciales a partir de los artículos y las noticias que se publican sobre los productos del mercado en las revistas impresas, como *Datamation*, *Byte*, *ComputerWorld*, *Computing*, y en las revistas digitales, como *Intelligent Enterprise*.

Encontraréis la revista digital *Intelligent Enterprise* en la dirección de Internet siguiente: [www.intelligententerprise.com](http://www.intelligententerprise.com)

WEB

## Ejercicios de autoevaluación

1. ¿Cuál es la diferencia esencial entre los SGBD relacionales y los prerelacionales?
2. ¿Cuáles son las ventajas principales del SQL hospedado estático respecto al basado en librerías de funciones?
3. ¿Qué caracteriza a los SGBD auténticamente paralelos?
4. Suponed que compráis un SGBD, pero en la versión personal. Sabéis que ocupa mucha menos memoria que la versión multiusuario. Es probable que de las doce funcionalidades que hemos citado como más frecuentes de los motores de los SGBD falten algunas. ¿Cuáles os parece que podrían ser?

## Solucionario

### Ejercicios de autoevaluación

1. En los sistemas prerelacionales el usuario está mucho más ligado al mundo físico. Hay muy poca independencia (tanto física como lógica) de los datos. Así, por ejemplo, prácticamente cualquier cambio que se introduzca en la estructura de los datos obliga a retocar los programas o a escribir las consultas de manera diferente.

2. En el SQL hospedado estático, el análisis y la traducción de las instrucciones se hace en tiempo de compilación; en cambio, si usamos librerías de funciones se hace en tiempo de ejecución. Así pues, las llamadas a funciones en principio son menos eficientes que las instrucciones incorporadas al programa. Por otra parte, la segunda técnica nos obliga a saber, además de la sintaxis del SQL, la sintaxis de las llamadas a las funciones de la librería que, por cierto, generalmente no son muy sencillas.

3. En un SGBD paralelo cada orden SQL recibida es dividida en partes, de manera que se puedan ejecutar independientemente las unas de las otras. Cada parte se ejecuta en un procesador separado (que quizás trabajaría con discos separados) y se ponen en común los resultados parciales para formar el resultado final. La ganancia de tiempo puede llegar a ser espectacular.

4. Una versión monousuario no necesita la funcionalidad del acceso concurrente y es posible que no necesite un rendimiento muy elevado. Además, si ocupa poca memoria, puede ser que las funcionalidades que pueda hacer el SO se hayan eliminado del SGBD. Por lo tanto, podría ser que las funcionalidades desaparecidas (algunas quizás sólo parcialmente) fueran las siguientes:

- Gestión de memorias intermedias.
- Gestión del espacio de la memoria externa.
- Métodos de acceso.
- Gestión de procesos.
- Control del acceso concurrente.
- Control de la seguridad y la privacidad.

## Glosario

**base de datos** *f* Conjunto estructurado de datos que representa, entre otros, entidades y sus interrelaciones, con integración y compartimentación de datos.

**sigla:** BD

**BD** *Ved* base de datos.

**cliente/servidor** *f* Tecnología habitual para distribuir datos. Dos procesos diferentes, que se pueden ejecutar en un mismo sistema o en sistemas separados, actúan de manera que uno hace de cliente, o peticionario de un servicio, y el otro hace de servidor.

Un proceso cliente puede pedir servicios a diversos servidores, y un servidor puede recibir peticiones de muchos clientes. En general, un proceso A que pide un servicio (hace de cliente) a un proceso B también puede hacer de servidor de un servicio que le pida otro proceso C.

**sigla:** C/S

**C/S** *Ved* cliente-servidor.

**esquema** *m* Descripción o definición de la BD. Esta descripción está separada de los programas y es utilizada por el SGBD para saber cómo es la BD con la que tiene que trabajar. La arquitectura ANSI/SPARC recomienda tres niveles de esquemas: el esquema externo (visión de los usuarios), el esquema conceptual (visión global) y el esquema físico (descripción de las características físicas).

**modalidades de SQL** *f pl* Conjunto de modalidades: modalidad directa o interactiva, llamadas a librerías de funciones, y modalidad de SQL hospedado.

**sistema de gestión de bases de datos** *m Software* que gestiona y controla bases de datos. Sus principales funciones son las de facilitar la utilización simultánea a muchos usuarios de tipos diferentes, independizar al usuario del mundo físico y mantener la integridad de los datos.

**sigla:** SGBD

**SGBD** *Ved* sistema de gestión de bases de datos.

**SQL** *m* Lenguaje pensado para describir, crear, actualizar y consultar, BD. Fue concebido por IBM a finales de los años setenta y estandarizado por ANSI y ISO en el año 1986 (el último estándar del SQL es de 1999). Actualmente lo utilizan casi todos los SGBD del mercado (incluso algunos SGBD no relacionales y algunos sistemas de ficheros).

*structured query language* m Ved SQL.

**transacción** *f* Conjunto de operaciones (contra la BD) que se ejecutan como una unidad (o todo o nada) y aisladamente (sin interferencias) de otros conjuntos de operaciones que se ejecutan concurrentemente.

## Bibliografía

### Bibliografía básica

**Date, C.J.** (2001). *Introducción a los sistemas de bases de datos* (7.<sup>a</sup> edición). Prentice-Hall. Es el libro más popular sobre BD en el mundo de la enseñanza universitaria. La introducción ofrece una buena visión global del tema.

**Silberschatz, A.; Korth, H.F.; Sudarshan, S.** (1998). *Fundamentos de bases de datos* (3.<sup>a</sup> edición). Madrid: Mc Graw-Hill.  
Para mejorar la comprensión de este módulo recomendamos leer los capítulos 1 y 16 de esta obra.

### Bibliografía complementaria

Para actualizar vuestra visión global de los SGBD del mercado, podéis consultar en Internet las páginas de los fabricantes de SGBD (Informix, Oracle, IBM, CA, Microsoft, NCR, etc.), especialmente los documentos de tipo *white papers*, *product overview*, *technical briefs*, etc. También encontraréis información de interés sobre los SGBD del mercado en las siguientes direcciones de Internet:

**Intelligent Enterprise.** [www.intelligententerprise.com](http://www.intelligententerprise.com).

**Oracle view.** [www.oreview.com](http://www.oreview.com).

**DB2 magazine.** [www.db2mag.com](http://www.db2mag.com).

En las siguientes direcciones encontraréis información sobre el mundo de la investigación sobre BD:

**ACM Sigmod.** [www.acm.org/sigmod](http://www.acm.org/sigmod).

**Computer Science Bibliography (Universidad de Trier).** [www.informatic.uni-trier.de/~ley/db](http://www.informatic.uni-trier.de/~ley/db)

**Teradata.** [www.teradata.com](http://www.teradata.com).

#### Recordad que...

... si x es el nombre de la empresa, su dirección de Internet acostumbra a ser [www.x.com](http://www.x.com).

