WIKIPEDIA
La enciclopedia libre

Sistema de gestión de bases de datos

Un **sistema gestor de base de datos** o **SGBD** (del inglés: Data Base Management System o DBMS) es un software que permite administrar una base de datos. Esto significa que mediante este programa se puede utilizar, configurar y extraer información almacenada[1] . Los usuarios pueden acceder a la información usando herramientas específicas de consulta y de generación de informes, o bien mediante aplicaciones al efecto.

Estos sistemas también proporcionan métodos para mantener la integridad de los datos, para administrar el acceso de usuarios a los datos y para recuperar la información si fallo del sistema y hacer copias de seguridad. Las bases de datos y los sistemas para su gestión son esenciales para cualquier área de negocio, y deben ser gestionados con esmero.

Algunos ejemplos de SGBD son MySQL, MariaDB, PostgreSQL, Microsoft SQL Server, Oracle Database y Microsoft Access.

Introducción

Las bases de datos generalmente funcionan en computadoras que se dedican de forma exclusiva a este campo. Por las prestaciones requeridas, generalmente funcionan en computadoras multiprocesador con abundante memoria.

Para el almacenamiento de los datos puede contar con sistemas de disco propio o almacenamiento de conexión directa (DAS), puede conectarse a una red de almacenamiento (SAN) o conectarse a un sistema de almacenamiento en red (NAS).

Existen aceleradores hardware, usados en grandes sistemas de proceso de transacciones. Los SGBD se encuentran en el corazón de toda aplicación que maneje datos. Los SGBD se basan en sistemas operativos estándar para efectuar dichas funciones.

Historia

Las bases de datos han estado en uso desde los primeros días de las computadoras electrónicas. A diferencia de los sistemas modernos, que se pueden aplicar a datos y necesidades muy diferentes, la mayor parte de los sistemas originales estaban enfocados a bases de datos específicas y pensados para ganar velocidad a costa de perder flexibilidad. Los SGBD originales solo estaban a disposición de las grandes organizaciones que podían disponer de las complejas computadoras necesarias.

Sistemas de navegación (1960)

Según las computadoras fueron ganando velocidad y capacidad, aparecieron sistemas de bases de datos de propósito general; a mediados de 1960 ya había algunos sistemas en uso. Apareció el interés en obtener un estándar y Charles Bachman —autor de uno de los primeros productos, el *Integrated Data Store* (IDS)— fundó el Database Task Group dentro de CODASYL, el grupo responsable de la creación y estandarización de COBOL. En 1971 publicaron su estándar, que pasó a ser conocido como la «aproximación CODASYL», y en breve aparecieron algunos productos basados en esta línea.

La estrategia de CODASYL estaba basada en la navegación manual por un conjunto de datos enlazados en red. Cuando se arrancaba la base de datos, el programa devolvía un enlace al primer registro de la base de datos, el cual a su vez contenía punteros a otros datos. Para encontrar un registro concreto el programador debía ir siguiendo punteros hasta llegar al registro buscado.

Para responder a preguntas simples como «buscar todas las personas en Japón» el programa debía recorrer todos los datos para escoger los registros correctos. No existían los conceptos «buscar» ni «encontrar», algo que sería inaceptable hoy en día, pero que en los tiempos en que los datos se guardaban en cintas no era viable llevarlos a la práctica.

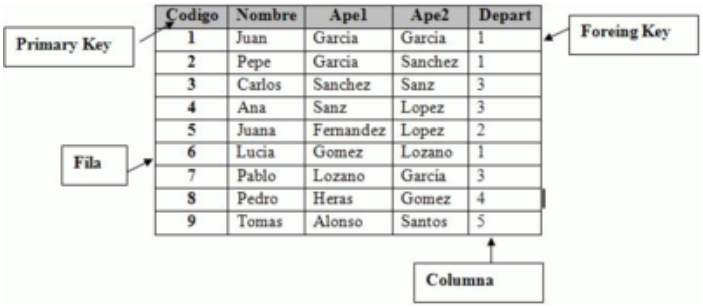
Se encontraron soluciones a muchos de esos problemas. El fabricante Prime creó un SGBD ajustado a CODASYL basado en árboles binarios que atajaba la navegación de registro en registro proveyendo caminos alternativos de acceso. También aportaba un lenguaje de consulta muy claro. De hecho no hay razón para no poder aplicar los

conceptos de normalización a bases de datos CODASYL, pero en último término CODASYL resultaba muy complejo y requería de mucho esfuerzo y práctica para producir una aplicación útil.

IBM también tenía su SGBD propio en 1968, conocido como IMS. Se trataba de un software desarrollado para el programa Apolo sobre System/360. IMS tenía conceptos similares a CODASYL, pero usaba una jerarquía estricta de ordenación de los datos, frente a la estructura en red de CODASYL. Ambos conceptos fueron englobados posteriormente en el concepto de bases de datos de navegación debido al modo de acceso a los datos, de hecho Bachman recibió al premio Turing en 1973 por su ponencia *El programador como navegador*.[2]

Sistemas relacionales (1970)

Edgar Codd trabajaba en IBM, en una de esas oficinas periféricas que estaba dedicada principalmente al desarrollo de discos duros. Estaba descontento con el modelo de navegación CODASYL, principalmente con la falta de operación de búsqueda. En 1970 escribió algunos artículos en los que perfilaba una nueva aproximación que culminó en el documento "A Relational Model of Data for Large Shared Data Banks".[3]



Codigo	Nombre	Ape1	Ape2	Depart
1	Juan	Garcia	Garcia	1
2	Pepe	Garcia	Sanchez	1
3	Carlos	Sanchez	Sanz	3
4	Ana	Sanz	Lopez	3
5	Juana	Fernandez	Lopez	2
6	Lucia	Gomez	Lozano	1
7	Pablo	Lozano	Garcia	3
8	Pedro	Heras	Gomez	4
9	Tomas	Alonso	Santos	5

Ejemplo de una tabla en el modelo relacional.

En este artículo descubrió un nuevo sistema para almacenar y trabajar con grandes bases de datos. En vez de almacenar registros de tipo arbitrario en una lista encadenada como en CODASYL, la idea de Codd era usar una "tabla" de registros de tamaño fijo. Una lista encadenada tiene muy poca eficiencia al almacenar datos dispersos donde algunos de los datos de un registro pueden dejarse en blanco. El modelo relacional resuelve esto dividiendo los datos en una serie de tablas —o relaciones— normalizadas, en las que los elementos optativos han sido extraídos de la tabla principal para que ocupen espacio solo si lo necesitan. En este modelo relacional los registros relacionados se enlazan con una "clave".

Un uso común de las bases de datos puede mantener una agenda de usuarios, su nombre, información de acceso, dirección y teléfono. En la solución de navegación todos esos datos estaría localizados en un solo registro, y las características no usadas simplemente no estarían en la base de datos. En la solución relacional, los datos estarían normalizados en una tabla de usuario, una de teléfono y una de dirección, en la que serían añadidos registros si tuviéramos que incorporar teléfono y dirección.

Reconciliar toda la información es la clave de este sistema. En el modelo relacional, una parte de la información se usa como clave, identificando de manera biunívoca un registro concreto. Cuando se recopila información acerca de un usuario, se accederá a la información de las tablas optativas buscando mediante esa clave. Por ejemplo si el nombre de usuario es único, la dirección y número de teléfono de ese usuario será guardada con el nombre de usuario como clave. La recopilaciòn de esta información en un solo registro es algo para lo que los lenguajes tradicionales no están pensados.

Así como el enfoque de navegación requiere programas que realicen bucles para recolectar registros, el enfoque relacional también los requerirá. La solución de Codd para los necesarios bucles se basa en un lenguaje orientado a conjuntos, una sugerencia que más tarde cristalizaría en el ubicuo SQL. Planteó el uso de una rama del álgebra llamada cálculo de tuplas, y demostró que con ella se podrían realizar todas las operaciones típicas sobre una base de datos, además de extraer conjuntos de datos de una forma sencilla.

El artículo de Codd cayó en manos de dos personas en Berkeley, Eugene Wong y Michael Stonebraker. Ellos comenzaron un proyecto llamado *INGRES* con fondos asignados a un proyecto de base de datos geográfica programada por los estudiantes. Comenzando en 1973, INGRES produjo sus primeras versiones de prueba que estuvieron listas para uso general en 1979. INGRES era muy similar a System R de IBM en varios aspectos, incluyendo un lenguaje para acceso a los datos, conocido como QUEL. Con el paso del tiempo, INGRES adoptó el estándar SQL.

IBM realizó una implementación de prueba del modelo relacional —PRTV— y una de producción —Business System 12— ambas descontinuadas. Honeywell escribió MRDS para Multics, y aparecen también dos nuevas implementaciones: Alphora Dataphor y Rel. La mayoría de las demás implementaciones de SGBD llamados relacionales son en realidad SGBD SQL.

En la década de 1970, la Universidad de Míchigan comenzó el desarrollo del *MICRO Information Management System* basado en el modelo teórico de datos de D. L. Childs. *Micro* fue utilizado para gestionar gran cantidad de datos en el Departamento de Trabajo del gobierno de EUA. Corría en *mainframe* usando *Michigan Terminal System*. Estuvo en producción hasta 1998.

Sistemas SQL (finales de década de 1970)

IBM empezó a trabajar a principios de 1970 en un prototipo lejanamente basado en los conceptos de Codd llamándolo System R. La primera versión estuvo lista en 1974 o 1975, y comenzó así el trabajo en sistemas multitabla, en los que los datos podían disgregarse de modo que toda la información de un registro (alguna de la cual es opcional) no tiene que estar almacenada en un único trozo grande. Las versiones multiusuario siguientes fueron probadas por los usuarios en 1978 y 1979, tiempo por el que un lenguaje SQL había sido estandarizado. Las ideas de Codd se revelaron como operativas y superiores a las de CODASYL, lanzando a IBM al desarrollo de una verdadera versión de producción de System R, conocido como SQL/DS, y posteriormente como Database 2 (DB2).

Muchos de los técnicos de INGRES estaban seguros del éxito comercial del sistema, y formaron sus propias compañías para comercializar el desarrollo pero con una interfaz SQL. Sybase, Informix, NonStop SQL y la misma INGRES se vendían como derivados del INGRES original en los años 1980. Incluso el SQL Server de Microsoft está basado en Sybase, y por consiguiente en INGRES. Solo Larry Ellison —el fundador de Oracle— comenzó un nuevo camino basado en el artículo de IBM sobre System R, y aventajó a IBM sacando al mercado su primera versión en 1978.

Stonebraker aplicó las lecciones de INGRES al desarrollo de una nueva base de datos —Postgres— conocida ahora como PostgreSQL. PostgreSQL se utiliza para muchas aplicaciones críticas (los registros de dominios .org y .info lo usan para su almacenamiento primario, así como grandes compañías e instituciones financieras).

En Suecia, el artículo de Codd generó la base de datos Mimer SQL[4] en la Universidad de Uppsala. En 1984 este proyecto se consolidó en una compañía independiente. A principios de 1980, Mimer introdujo la gestión de transacciones para dar robustez a las aplicaciones, una idea que fue recogida en muchos otros SGBD.

Sistemas orientados a objetos (1980)

Durante la década de 1980 el auge de la programación orientada a objetos influyó en el modo de manejar la información de las bases de datos. Programadores y diseñadores comenzaron a tratar los datos en las bases de datos como objetos. Esto quiere decir que si los datos de una persona están en la base de datos, los atributos de la persona como dirección, teléfono y edad se consideran que pertenecen a la persona, no son datos extraños. Esto permite establecer relaciones entre objetos y atributos, más que entre campos individuales.

Otro gran foco de atención durante la década fue el incremento de velocidad y fiabilidad en el acceso. En 1989, dos profesores de la Universidad de Wisconsin publicaron un artículo en una conferencia ACM en el que exponían sus métodos para mejorar las prestaciones de las bases de datos. La idea consistía en replicar la información importante —y más solicitada— en una base de datos temporal de pequeño tamaño con enlaces a la base de datos principal. Esto implicaba que se podía buscar mucho más rápido en la base de datos pequeña que en la grande. Su mejora de prestaciones llevó a la introducción de la indización, incorporado en la totalidad de los SGBD.

Sistemas NoSQL (2000)

El siglo xxi trajo una nueva tendencia en las bases de datos: el NoSQL. Esta tendencia introducía una línea no relacional significativamente diferentes de las clásicas. No requieren por lo general esquemas fijos, evitan las operaciones *join* almacenando datos desnormalizados y están diseñadas para escalar horizontalmente. La mayor parte de ellas pueden clasificarse como almacenes clave-valor o bases de datos orientadas a documentos.

Recientemente ha habido una gran demanda de bases de datos distribuidas con tolerancia a particiones, pero de acuerdo con el teorema CAP no es posible conseguir un sistema distribuido que simultáneamente proporcione consistencia, disponibilidad y tolerancia al particionado. Un sistema distribuido puede satisfacer solo dos de las tres restricciones a la vez. Por dicha razón muchas de las bases de datos NoSQL usan la llamada consistencia eventual para proporcionar disponibilidad y tolerancia al particionado, con un nivel máximo de consistencia de datos.

Entre las aplicaciones más populares encontramos MongoDB, MemcacheDB, Redis, CouchDB, Hazelcast, Apache Cassandra y HBase, todas ellas de código abierto.

Sistemas XML (2010)

Las bases de datos XML forman un subconjunto de las bases de datos NoSQL. Todas ellas usan el formato de almacenamiento XML, que está abierto, legible por humanos y máquinas y ampliamente usado para interoperabilidad.

En esta categoría encontramos: BaseX, eXist, MarkLogic Server, MonetDB/XQuery, Sedna.

Componentes

- El **motor de la base de datos** acepta peticiones lógicas de los otros subsistemas del SGBD, las convierte en su equivalente físico y accede a la base de datos y diccionario de datos en el dispositivo de almacenamiento.
- El **subsistema de definición de datos** ayuda a crear y mantener el diccionario de datos y define la estructura del fichero que soporta la base de datos.
- El **subsistema de manipulación** de datos ayuda al usuario a añadir, cambiar y borrar información de la base de datos y la consulta para extraer información. El subsistema de manipulación de datos suele ser la interfaz principal del usuario con la base de datos. Permite al usuario especificar sus requisitos de la información desde un punto de vista lógico.
- El **subsistema de generación de aplicaciones** contiene utilidades para ayudar a los usuarios en el desarrollo de aplicaciones. Usualmente proporciona pantallas de entrada de datos, lenguajes de programación e interfaces.
- El **subsistema de administración** ayuda a gestionar la base de datos ofreciendo funcionalidades como almacenamiento y recuperación, gestión de la seguridad, optimización de preguntas, control de concurrencia y gestión de cambios.

Lenguajes de modelación

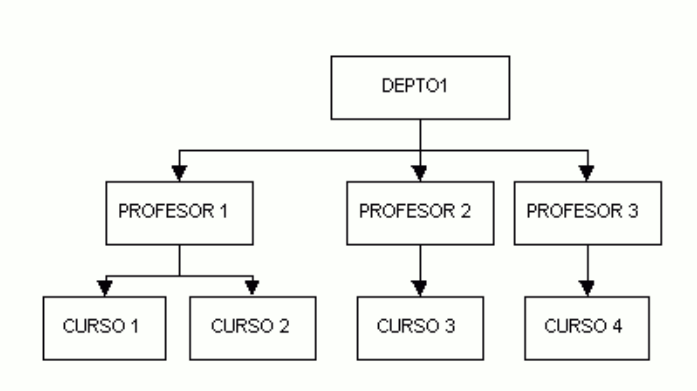
Toda base de datos soportada por un SGBD debe tener unos esquemas modelados adecuadamente. Coincidiendo con la evolución histórica de las bases de datos, estas han utilizado distintos modelos. Los SGBD esperan un modelo determinado para poder acceder de forma simple a la base de datos. Estos modelos son:

- Jerárquicos
- En red
- Relacionales
- Multidimensionales
- De objetos

También se han utilizado listas invertidas.

Estructura jerárquica

La estructura jerárquica fue usada en los SGBD de los primeros *mainframe*. Las relaciones entre registros forman una estructura en árbol. Esta estructura es simple pero inflexible ya que las relaciones están confinadas al tipo 1:n. El sistema IMS de IBM y el RDM Mobile de Raima[5] son ejemplos de bases de datos con múltiples jerarquías sobre el mismo conjunto de datos. RDM Mobile es un nuevo diseño de base de datos imbuida para una red de ordenadores móviles. La estructura jerárquica es usada hoy en día para almacenar información geográfica principalmente.



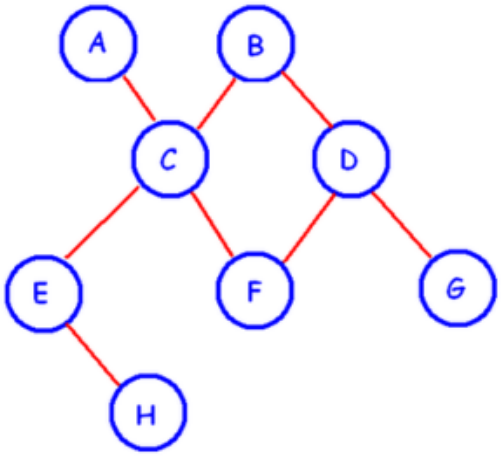
Ejemplo de un modelo de una base de datos jerárquica

El modelo de base de datos jerárquica tiene un esquema en el que los datos se organizan en una estructura arbórea. Esta estructura permite representar relaciones padre/hijo: cada padre puede tener varios hijos, pero cada hijo ha de venir de solo un padre (las conocidas como relaciones 1:N). Todos los atributos de un registro específico están asociados a un tipo de entidad. Este modelo fue creado por IBM en 1960.

En una base de datos una entidad tipo es el término genérico para tabla. Cada registro individual se representa como una fila, y cada atributo como una columna. Las entidades tipo se relacionan entre ellas usando correspondencias 1:N.

Actualmente las bases de datos jerárquicas más utilizadas son IMS de IBM y el Registro de Windows de Microsoft.

Estructura en red



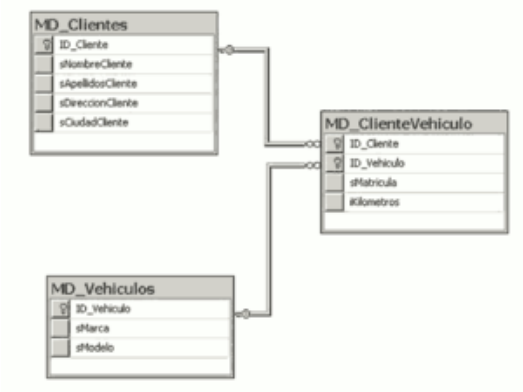
Modelo de base de datos en red

Esta estructura contiene relaciones más complejas que las jerárquicas. Admite relaciones de cada registro con varios que se pueden seguir por distintos caminos. En otras palabras, el modelo permite relaciones N:N.

El modelo en red está concebido como un modo flexible de representar objetos y sus relaciones. Su cualidad distintiva es que el esquema —visto como un conjunto de nodos conectados por arcos— no tiene ninguna restricción.

El inventor de este modelo fue Charles Bachman, y el estándar fue publicado en 1969 por CODASYL.

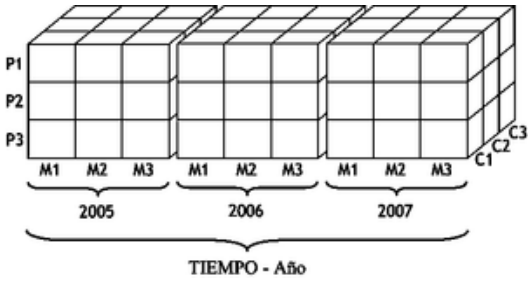
Estructura relacional



Ejemplo de tablas y relaciones.

La estructura relacional es la más extendida hoy en día. Se usa en *mainframes*, computadoras medias y microcomputadoras. Almacena los datos en filas (tuplas) y columnas (atributos). Estas tablas pueden estar conectadas entre sí por claves comunes. Mientras trabajaba en IBM en 1972, E. F. Codd concibió esta estructura. El modelo no resulta sencillo de consultar por el usuario ya que puede requerir una compleja combinación de tablas.

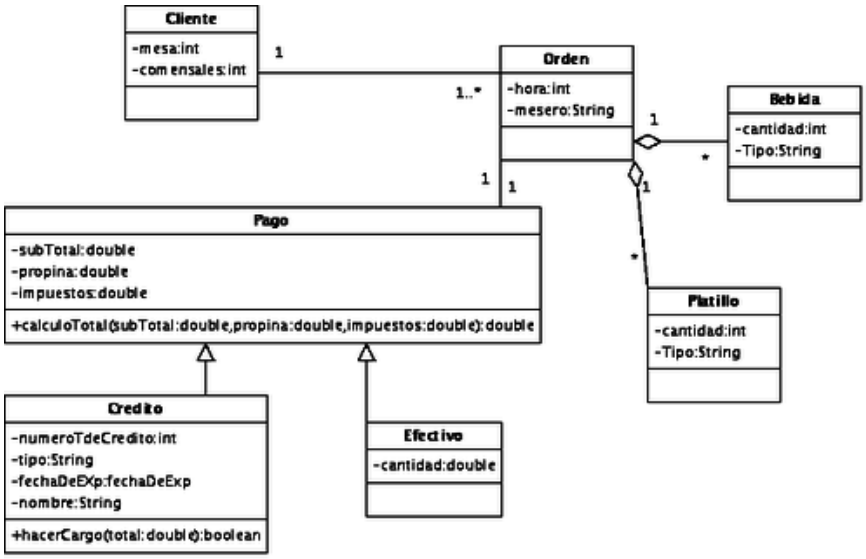
Estructura multidimensional



Cubos representando 4 dimensiones en base de datos multidimensional

La estructura multidimensional tiene parecidos a la del modelo relacional, pero en vez de las dos dimensiones filas-columnas, tiene N dimensiones. Esta estructura ofrece el aspecto de una hoja de cálculo. Es fácil de mantener y entender ya que los registros se almacenan del mismo modo como se ven. Sus altas prestaciones han hecho de ella la base de datos más popular para el proceso analítico de transacciones en línea (OLAP).

Estructura orientada a objetos



Ejemplo de base de datos conteniendo objetos y herencias

La estructura orientada a objetos está diseñada siguiendo el paradigma de los lenguajes orientados a objetos. De este modo soporta los tipos de datos gráficos, imágenes, voz y texto de manera natural. Esta estructura tiene gran difusión en aplicaciones web para aplicaciones multimedia.

Antes de la implantación de los SGBD con estructura orientada a objetos, el almacenamiento de datos multimedia se basaba en el sistema de ficheros para organizar, almacenar y procesar los datos. El proceso de ficheros es engorroso, costoso e inflexible. La redundancia de los datos es un inconveniente del proceso de ficheros ya que los ficheros independientes producen ficheros duplicados con su implicación en el espacio necesario. Otro inconveniente es la falta de integración, y la dificultad de mantenimiento. Esto fue encaminado aplicando la orientación a objetos a los datos.

Lenguajes de consulta

Los lenguajes de consulta de bases de datos y de generación de informes permiten consultar a la base de datos, analizar los datos y actualizarlos según los privilegios de cada usuario. También controla la seguridad de la base de datos para prevenir accesos no autorizados que vean, borren o cambien los datos. Mediante el uso de claves se

permite el acceso a toda la base de datos o a parte de ella. A modo de ejemplo, una base de datos de empleados puede contener todos los datos de los empleados, pero solo un grupo de usuarios puede estar autorizado a ver las nóminas mientras que otros pueden estar autorizados a ver solo las historias laborales y los datos médicos, o consultas.

Si el SGBD proporciona un modo de acceder y actualizar la base de datos, así como de consultarla, este posibilitará la creación de bases de datos personales. Sin embargo, le faltaría la capacidad de dejar trazas de las acciones o los controles necesarios que necesita la base de datos de una gran organización. Estos controles están solo disponibles cuando un conjunto de programas auxiliares supervisan los accesos y actualizaciones de los datos.

Arquitectura

La arquitectura de un SGBD especifica sus componentes (incluyendo su descripción funcional) y sus interfaces. Trata de conceptos distintos que la arquitectura de la base de datos. Los componentes principales de un SGBD son:

- **Interfaces externas**: medios para comunicarse con el SGDB en ambos sentidos (E/S) y explotar a todas sus funciones. Pueden afectar a la BD o a la operación del SGBD, por ejemplo:
  - operaciones directas con la base de datos: definición de tipos, asignación de niveles de seguridad, actualización de datos, consulta de la base de datos...
  - operaciones relativas a la operación del SGBD: copia de seguridad y restauración, recuperación tras una caída, monitoreo de seguridad, gestión del almacenamiento, reserva de espacio, monitoreo de la configuración, monitoreo de prestaciones, afinado...
  - las interfaces externas bien pueden ser utilizadas por usuarios (p. e. administradores) o bien por programas que se comunican a través de una API.
- **Intérprete o procesador del lenguaje**: la mayor parte de las operaciones se efectúan mediante un lenguaje de base de datos. Existen lenguajes para definición de datos, manipulación de datos (p. e. SQL), para especificar aspectos de la seguridad y más. Las sentencias en ese lenguaje se introducen en el SGBD mediante la interfaz adecuada. Se procesan las expresiones en dicho lenguaje (ya sea compilado o interpretado) para extraer las operaciones de modo que puedan ser ejecutadas por el SGBD.
- **Optimizador de consultas**: realiza la optimización de cada pregunta y escoge el plan de actuación más eficiente para ejecutarlo.
- **Motor de la base de datos**: realiza las operaciones requeridas sobre la base de datos, típicamente representándolo a alto nivel.
- **Mecanismo de almacenamiento**: traduce las operaciones a lenguaje de bajo nivel para acceder a los datos. En algunas arquitecturas el mecanismo de almacenamiento está integrado en el motor de la base de datos.
- **Motor de transacciones**: para conseguir corrección y fiabilidad, la mayoría de las operaciones internas del SGBD, se realizan encapsuladas dentro de transacciones. Las transacciones pueden ser especificadas externamente al SGBD para encapsular un grupo de operaciones. El motor de transacciones sigue la ejecución de las transacciones y gestiona su ejecución de acuerdo con las reglas que tiene establecidas (p. ej., control de concurrencia y su ejecución o cancelación).
- **Gestión y operación de SGBD**: comprende muchos otros componentes que tratan de aspectos de gestión y operativos del SGBD como monitoreo de prestaciones, gestión del almacenamiento, mapas de almacenamiento.

Véase también

- Base de datos
- Almacén de datos
- Anexo:Comparación de sistemas administradores de bases de datos relacionales

Referencias

1. «Introducción al sistema gestor de base de datos (SGBD)» (https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/sistema-gestor-de-base-de-datos-sgbd/). *IONOS Digitalguide*. Consultado el 27 de abril de 2022.
2. Bachman, Charles W. «*The programmer as navigator*» (http://dl.acm.org/citation.cfm?id=362534) (en inglés). Consultado el 17 de febrero de 2013.
3. Codd, E. F. (1970). *"A Relational Model of Data for Large Shared Data Banks"* (http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf). In: *Communications of the ACM* 13 (6): 377-387.