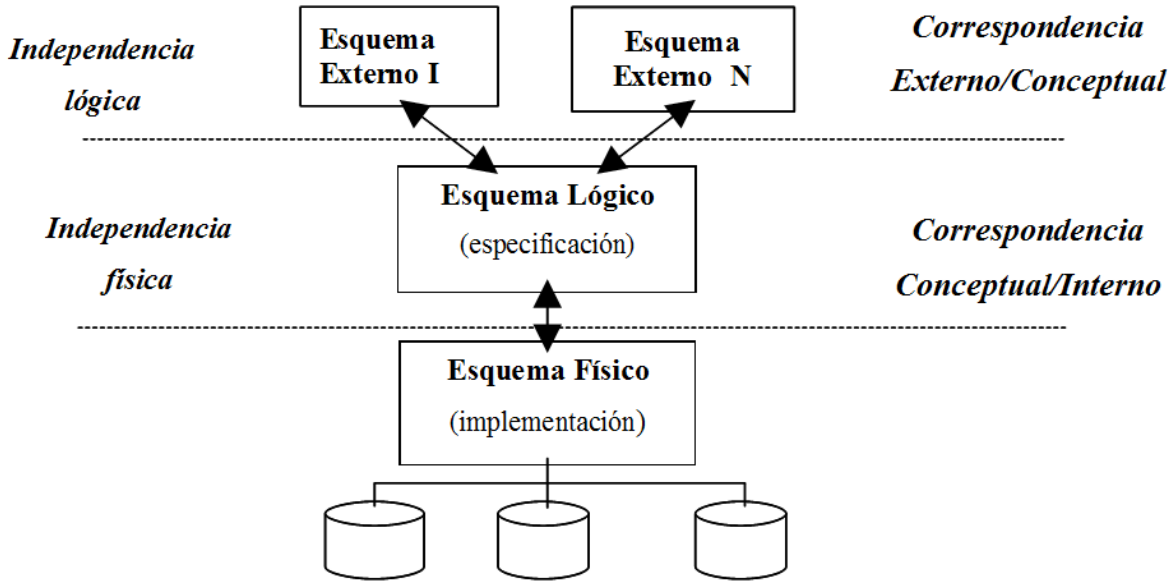
Sistemas Gestores de bases de datos libres y comerciales

Arquitectura de niveles del SGBD.

Para asegurar que las características más importantes de una BD se cumplan es necesario que los SGBD presenten una arquitectura que efectivamente garanticen la independencia de los datos y aplicaciones, posibiliten la creación de múltiples vistas y además de un catálogo para la descripción de la BD. Se trata, entre otras cosas, de que cumplan con el mayor número posible de reglas de Codd.

El comité ANSI/SPARC propuso la arquitectura de los SGBD de tres niveles que finalmente se ha estandarizado.



Un SGBD permite la definición de BD en tres niveles de abstracción: lógico, físico y externo. La definición de la bases de datos en cada uno de estos niveles se denomina **esquema.**

- En el **nivel externo** se definen las vistas parciales de la base de datos para distintos grupos de usuarios.
- En el **nivel lógico** se realiza la definición de las estructuras de datos que constituyen la base de datos. A esta definición se le denomina **esquema lógico.**
- En el **nivel físico** (o interno) se elige una implementación para cada una de las estructuras definidas en el esquema lógico. A esta definición se le denomina **esquema físico.**

Los tres esquemas son distintas maneras de describir los datos aunque estos realmente existen en el nivel físico. Sin embargo, el sistema físico ha de ser totalmente transparente a los usuarios y éstos tan solo han de ver su esquema externo. Cualquier referencia al esquema externo se traducirá en una petición, por parte del sistema operativo a referencia a los datos oportunos del esquema lógico. Posteriormente, se deberá de traducir en una solicitud al esquema físico.

Por ejemplo, supongamos una vista donde tenemos el nombre y la edad de los empleados. Una solicitud de toda la vista externa se deberá traducir en una solicitud de los datos correctos del esquema lógico, nombre y fecha de nacimiento. Posteriormente se traducirá al esquema físico donde se sabrá exactamente donde están los datos, si hay un índice para hacer más rápido el acceso, etc. Después, cuando se consigan los datos, el proceso de traducción irá a la inversa, es decir, de abajo a arriba en los niveles de la arquitectura.

El proceso de transformación solicitudes y resultados de un nivel a otro se denomina **correspondencia o transformación (*mapping*)**. Este proceso consumirá tiempo pero facilitará lo que se pretendía:

- **Independencia lógica** respecto a los datos: Es la capacidad de modificar el esquema lógico sin que ello afecte a los esquemas externos ni programas de aplicación. Por ejemplo, en el esquema lógico se puede incorporar un nuevo campo para los empleados que sea la fecha de entrada en la empresa. Esto no afectará para nada la vista externa de empleado y edad correspondiente.
- **Independencia física** respecto a los datos: se puede modificar el esquema físico sin que ello tenga que afectar al esquema lógico y mucho menos a los esquemas externos. Así por ejemplo, se puede decidir añadir más espacio para la BD incorporando un nuevo fichero donde se guarden las cosas (Oracle) o se decide incluir un nuevo índice para ir más rápido en el acceso a los datos en un determinado orden.

El conseguir la independencia tanto física como lógica es una de las metas fundamentales de los SGBD

Ejemplo Clarificador:[1]

Vamos a utilizar una base de datos para llevar la información asociada a una empresa mediana.

Esquema externo:

El departamento comercial de la empresa debe saber en cada momento la región en la que trabaja cada empleado e incluso podría reasignarlos. ¿Debe conocer las nóminas de los empleados? Probablemente no. Esa información está oculta para el departamento comercial. Tenemos un esquema externo.

El departamento de nóminas debe conocer los días y horas trabajados y dietas y otros gastos. ¿Debe conocer qué ordenador utiliza cada empleado? Probablemente no. Tenemos otro esquema externo.

El departamento de informática debe conocer los usuarios que utilizan cada

1 En los ejemplos clarificadores intento, con lo que se ha visto hasta ahora, poner un ejemplo que ayude a hacernos una idea de los nuevos conceptos. Estos ejemplos no deben tomarse como definiciones correctas ni como ejemplos reales. Sé que se podrían poner muchas pegas a estos ejemplos, por lo que si a alguien no le parecen correctos, que los ignore.

ordenador físico por motivos de seguridad. ¿Debe conocer cuanto cobran esos empleados? Tenemos otro esquema externo.

Esquema lógico:

Cuando unimos todos los esquemas externos, obtenemos el esquema lógico que además de unir todos esos esquemas, utiliza un modelo de datos (el modelo relacional en nuestro caso) para representar toda esa información. ¿Hemos elegido ya el tipo de servidor en el que vamos a guardar los datos y el SGBD que utilizaremos? No. Estamos en el modelo lógico y este modelo podría valer (en teoría, en la práctica es difícil) para cualquier SGBD relacional.

Esquema físico:

A partir del esquema lógico y utilizando un SGBD real (un programa real, Oracle, Mysql u otro) obtenemos la base de datos física, la que está guardada en uno o más servidores, en uno o más discos duros, con sus ficheros, directorios, etc.

SGBD Actuales

Información general

	Creador	Fecha de primera versión	Licencia de software
DB2	IBM	1982	Propietario
Informix	Informix Sofware	1985	Propietario
Microsoft SQL Server	Microsoft	1989	Propietario
MySQL	MySQL AB	1996	Gratuito (por ahora)
Oracle	Oracle Corporation	1977	Propietario
PostgreSQL	PostgreSQL	1989	Gratuito

Sistemas Operativos Soportados

	S.O. Soportados
DB2	Windows, Linux, Unix
Informix	Windows, Linux, Mac OS X, Unix
Microsoft SQL Server	Windows
MySQL	Windows, Linux, Mac OS X,Unix
Oracle	Windows, Linux, Mac OS X, Unix
PostgreSQL	Windows, Linux, Mac OS X, Unix

Todos estos SGBD soportan las siguientes características:

ACID:

En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción (es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.). Así pues, si un sistema de gestión de bases de datos es ACID compliant quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID.

En concreto ACID es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.

Atomicidad: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

Consistencia: Integridad. Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.

Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.

Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Sacado de la Wikipedia

Ejemplo Clarificador:

Una persona va a sacar dinero a un cajero.

Atomicidad: o saca dinero o no, pero lo que no se debe permitir es que la

persona obtenga el dinero y no se apunte en su cuenta. Las dos operaciones, dar el dinero y restar de su cuenta serían la transacción. O se hace entera (las dos operaciones) o no se hace nada (niguna de las dos operaciones).

Consistencia: El sistema seguirá en un estado correcto. Después de sacar el dinero, la cuenta del banco será correcta teniendo en cuenta el dinero que queda en el cajero y el dinero que hay en la cuenta del cliente. Es una propiedad compleja de alcanzar

Aislamiento: Si un pariente del cliente saca exactamente a la misma vez dinero con la cartilla, no se producirá ningún error.

Durabilidad: Aunque nada más tener el dinero en la mano, el cajero se queda sin electricidad, la operación queda registrada.

MySQL solo soporta ACID en un tipo especial de tablas (se tiene que especificar).

Integridad referencial:

La integridad referencial significa que la clave externa de una tabla de referencia siempre debe aludir a una fila válida de la tabla a la que se haga referencia. La integridad referencial garantiza que la relación entre dos tablas permanezca sincronizada durante las operaciones de actualización y eliminación.

Ejemplo Clarificador:

Guardamos de los clientes el DNI, el nombre y el teléfono y es muy importante tener esos datos.

Cuando hacemos una factura a un cliente, sólo ponemos el DNI

La integridad referencial obliga a que **todos los DNI** que pongamos en las facturas tengan asociados un nombre y un teléfono. Es decir, si pongo un DNI me tengo que asegurar de que ya he metido antes el nombre y el teléfono. Además esas comprobaciones deben correr a cargo del SGBD.

Otra vez, MySQL solo soporta Integridad Referencial en un tipo especial de tablas (se tiene que especificar)

Unicode

Es un estándar para utilizar caracteres en varias lenguas (desde la ñ española hasta los ideogramas chinos).

Triggers (disparador)

Un trigger es un procedimiento que se ejecuta automáticamente cuando se modifica una base de datos.

> Ejemplo clarificador:
>
> Se guarda en la base de datos la cuenta de los clientes que tengo.
>
> Añado un cliente (Automáticamente, salta el trigger y le suma uno a la cuenta de clientes, no tengo que hacerlo yo, lo hace el SGBD)
>
> Quito un cliente (Salta el trigger y le resta uno a la cuenta de clientes)

Además todos estos SGBD soportan índices, vistas, guiones y procedimientos.

Breve recorrido por la historia de los SGBD[2]

DB2

1970:Origen de DB2 por parte de IBM
1983:Se empieza a comercializar
1994:DB2 pasa a ser DB2 UDB (DB2 Universal Database), ampliado con funciones avanzadas.

Informix

1980:Origen de Informix Tiene su origen esta a RDBMS
1990: INFORMIX era muy popular después de ORACLE.
2000: Decae financieramente la compañia INFORMIX.
2001: IBM compra este SGBD a INFORMIX.

Microsoft SQL Server

1988: Origen de SQL Server mediante la colaboración entre Microsoft, Sybase e IBM. Orientado a OS/2
1995: Microsoft lanza SQL Server 6.0 independientemente y como propietario
2005: La versión de SQL Server ya es competidora de Oracle y DB2
2008: Última versión de SQL Server

MySQL

1995: Origen de MySQL (opensource)
2008: La empresa propietaria de MySQL es subsidiaria de Sun Microsystems
2009: Sun Microsystems es subsidiaria de Oracle y, por tanto, también MySql

Oracle

1977: Origen de Oracle. Continúa siendo el paradigma de SGBD relacional.

2 No hay que estudiársela, sólo comparar los años de aparición y observar los movimientos entre empresas que se han sucedido.

PostgreSQL

1982: Origen de Ingres (intento de implementar un SGBD relacional en la universidad de Berkley)

1985: Nuevo proyecto post-Ingres o simplemente POSTGRES.

1989: Se publica la versión 1 para una pequeña comunidad de usuarios.

1993: crecimiento importante de la comunidad de usuarios, la cual demandaba más características.

1996: Se cambia el nombre a PostgreSQL El proyecto PostgreSQL continúa haciendo lanzamientos principales anualmente y lanzamientos menores de reparación de bugs, todos disponibles bajo la licencia BSD, y basados en contribuciones de proveedores comerciales, empresas aportantes y programadores de código abierto mayormente.

Uso de los SGBD

Oracle:

Más de 1500 administraciones del sector público utilizan aplicaciones Oracle
los 20 primeros de la lista de los mejores bancos utilizan Oracle
las 20 primeras compañias de telecomunicaciones utilizan Oracle
las 10 primeras universidades de todo el mundo utilizan Oracle
(sacado de la página web de Oracle)

MS SQL Server

Hay algunos ejemplos en
http://www.microsoft.com/spain/sql/productinfo/casestudies/default.mspx

DB2

En la actualidad la tecnología de gestión de datos de IBM es utilizada por más de 40 millones de usuarios de 300.000 empresas en todo el mundo. Mientras que la evolución del DB2, Universal Data Base dispone de más de 6 millones de usuarios y 1.300.000 licencias instaladas.

MySQL

Dentro de las empresas que utilizan Mysql en sus ambientes deproducción tenemos las siguientes: Sonny, Suzuki, Lycos, Yahoo, Dell, PortaOne, Nasa, Unicef, Mcafee, Aizawa Securities,Google

Y en http://www.mysql.com/customers/ hay más ejemplos

PostgreSQL

La American Chemical Society, BASF, IMDb, Skype, Sony Online, U.S. Departamento de Trabajo

Conclusión: De los SGBD mencionados, IBM es propietaria de dos, IBM e Informix, Oracle es propietaria de otros dos, Oracle y MySQL, Microsoft ofrece SQL Server sólo para sus Sistemas Operativos y PostgreSQL es una alternativa no comercial seria.