Soledad Ruiz Gómez, 3DGIIM

FUND. BASES DE DATOS - TEMA 2

ARQUITECTURA DE UN SGBD

NIVELES GENERALES DE LA ARQUITECTURA

¿Por qué organizar en niveles?

1. Permite a los usuarios acceder a los mismos datos, pero desde distintas perspectivas. Si un usuario decide cambiar la forma de ver los datos, esto no influye en el resto.

2. Además, facilita la independencia lógica, pues la organización global de los datos puede modificarse sin afectar a los usuarios.

3. Por último, los usuarios no tienen por qué gestionar aspectos relativos a la representación física de los datos. Esto es, el BDM puede cambiar la forma de representar los datos sin influir en los usuarios.

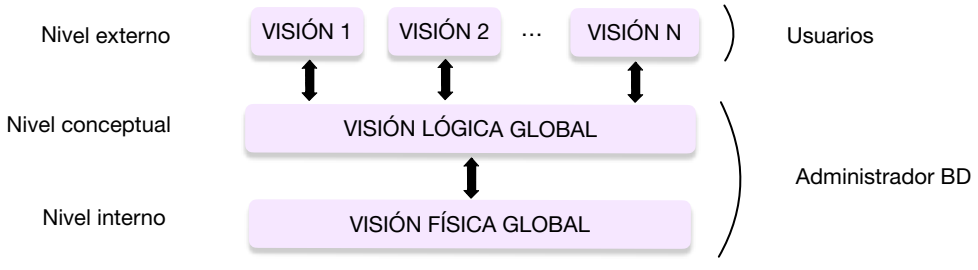
Arquitectura ANSI/SPARC

Consiste en que la percepción de los datos en un SGBD puede hacerse siguiendo tres niveles de abstracción: nivel interno, nivel conceptual y nivel externo.

- **Nivel interno**: consta de una única visión física global. Es la representación de la base de datos más cercana a la estructura de almacenamiento físico (a la máquina). Los niveles superiores se sustentan sobre sus estructuras de datos. A este nivel accede el DBM.
Equivalente a observar a una persona como un conjunto de huesos, músculos y órganos. Sólo personal cualificado es capaz de entender ese nivel.

- **Nivel conceptual**: consta de una única visión lógica global. Se encarga de la abstracción global de la base de datos (ve los datos desde un punto de vista lógico). Así, integra todas las percepciones que los distintos usuarios tienen de la BD. A este nivel accede el DBM.
Equivalente a observar a una persona tal y como la percibimos habitualmente, conociendo todas sus facetas.

- **Nivel externo**: consta de varias visiones distintas entre sí. Ello se debe a que aquí se definen las distintas percepciones particulares que los usuarios tienen de la BD. Naturalmente, a este nivel acceden los usuarios.
Equivalente al hecho de que una persona puede ser percibida de distintas maneras (su madre puede pensar que es muy guapa y que tiene que cuidarla, alguien a quien le cae mal puede pensar que es egoísta, una persona cercana puede tenerle cariño…)



Correspondencia entre niveles: una transformación o correspondencia entre niveles es un conjunto de normas que establece cómo se definen los datos de un nivel en términos de otro. Es un mecanismo fundamental para el establecimiento de la independencia (tanto física como lógica). Los distintos tipos son:

- **Transformación conceptual/interna**: indica cómo se organizan las entidades lógicas del nivel conceptual en términos de registros y campos almacenados en el nivel interno (almacenamiento físico).
A la hora de asegurar la independencia física, si se realiza un cambio en el nivel interno, *no* varía el nivel conceptual, sino la correspondencia.

Ejemplo: en el NI, tengo almacenados los datos de los alumnos a partir de una dirección de memoria X. Cuando en el NC se necesita información de los alumnos, hay una correspondencia que recoge en qué direcciones físicas de memoria están esos datos. Si necesito reubicar en memoria esa información (NI), claramente, la necesidad del NC de conocer los datos de los alumnos no va a variar. Lo que hay que modificar es la correspondencia, que ahora indicará al NC las nuevas posiciones de memoria donde podrá hallar esos datos.

- **Transformación externa/conceptual**: describe un esquema externo en términos del esquema conceptual subyacente. Para cada visión, he de ir al modelo lógico y seleccionar los datos que necesite. De ahí que haya una transformación externa/conceptual para cada una de las visiones del nivel externo.
A la hora de asegurar la independencia lógica, paralelamente a la transformación anterior, si se realiza un cambio en el nivel conceptual, *no* varía el nivel externo, sino la correspondencia. Aún así, recordar que no siempre es posible establecer esta independencia.

Ejemplo: en el NC, tengo un campo con el nombre y apellidos de cada alumno. Una vista del NE necesita el nombre completo de cada alumno, por lo que la transformación coge directamente el campo nombre_y_apellidos para la vista. Sin embargo, supongamos que realizamos un cambio en el NC y ahora almacenamos nombre, primer apellido y segundo apellido en tres campos diferentes. Como la necesidad de la vista no ha cambiado, ahora la transformación tendrá que saber que tiene que entregar la concatenación nombre+apellido1+apellido2 a la vista.

- **Transformación externa/externa**: algunos SGBD permiten describir esquemas externos en términos de otros esquemas externos.
De nuevo, a la hora de asegurar la independencia lógica, si hay un cambio en el esquema externo subyacente, *no* varía el esquema externo dependiente, sino la correspondencia. Volver a recordar que no siempre podemos asegurar esta independencia.

Lenguajes de una BD

1. En lo que respecta a este aspecto, la recomendación de ANSI/SPARC es disponer de un lenguaje específico orientado a la definición, control y manipulación de datos. Dicho lenguaje se conoce como un **DSL** (Domain Specific Language), que estará implementado en el propio SGBD. Este consta de varias partes:
- DDL (Data Definition Language): subconjunto del DSL para estructuras de datos y esquemas de la BD.
- DML (Data Manipulation Language): subconjunto del DSL para introducción, modificación, eliminación y consulta de datos. También debe permitir consultar los esquemas definidos en la BD.
- DCL (Data Control Language): subconjunto del DSL para gestionar los requisitos de acceso a los datos y otras tareas de administración de la BD.

ANSI/SPARC recomienda disponer de un DDL, un DML y un DCL para cada nivel de la arquitectura. En la práctica, todos estos sublenguajes se presentan bajo una única implementación: cada sentencia trabaja sobre uno o varios niveles y un sistema de privilegios discrimina quién puede ejecutar qué y en qué nivel.

A pesar de esta recomendación, que plantea lo que sería una situación "idílica", la realidad es otra: en la industria han aparecido distintos lenguajes, aunque se intenta estandarizar el uso y diseño de estos. Un ejemplo destacado es el SQL.

2. Por otro lado, también necesitamos un **lenguaje anfitrión** o **lenguaje de aplicación**. Este sirve para desarrollar las aplicaciones en el SO que trabajarán sobre la BD. Pueden ser lenguajes de propósito general (C/C++, Java…) o herramientas de desarrollo específicas (Developer de Oracle, Oracle APEX…). Este nos proporciona un procesamiento avanzado de datos, además de la gestión de la interfaz de usuario.

Una de las necesidades que se nos genera es que hay que poder establecer un mecanismo para trasladar de la BD al entorno de procesamiento de la aplicación (esto es, trasladar tanto las estructuras de datos como las operaciones).

Esta "traducción" BD-Aplicación tiene una característica llamada **acoplamiento**, que depende de cómo se compenetren el DSLy el lenguaje anfitrión. Estas parejas pueden clasificarse como:

- **Débilmente acoplados**: cuando como lenguaje anfitrión usamos lenguajes de propósito general. En este caso, las operaciones que podemos hacer nos sirven para estrategias más bien sencillas. El programador podrá distinguir sentencias propias del lenguaje anfitrión y sentencias dispuestas para acceder a la BD a través del DSL (me doy cuenta de cuándo estoy usando un lenguaje u otro).
¿Cómo podemos implementarlo?
+ APIs de acceso a BD: como Java Database Connectivity (JDBC) u Open Database Connectivity (ODBC).

+ DSL inmerso en el código fuente del lenguaje anfitrión: el programador escribe un código híbrido, mezclando sentencias del lenguaje anfitrión con sentencias DSL. Hay un preprocesador que transforma el código fuente del lenguaje anfitrión con llamadas API de acceso a la BD. Finalmente, esto se compila y se enlaza con la biblioteca de acceso a la BD. (ej: SQL inmerso en C)

- **Fuertemente acoplados**: cuando como lenguaje anfitrión usamos lenguajes y herramientas de propósito específico. Las operaciones que podemos hacer nos sirven para estrategias más complejas. Se parte del DSL como elemento central y se le incorporan características procedimentales para facilitar el desarrollo de aplicaciones. ¿Cómo podemos implementarlo? Hay varias opciones, bastante diversas pero la mayoría de ellas propietarias. Sobre todo, son extensiones del DSL para una mayor capacidad de procesamiento (ej: PL/SQL de Oracle, que es una extensión procedural de SQL). También se puede ejecutar Java sobre una máquina virtual implantada en el SGBD.

Aparte de todo esto, es necesario recalcar que también han aparecido numerosos entornos de desarrollo específicos para el desarrollo de aplicaciones de gestión (como informes, formularios…).

EL NIVEL EXTERNO

Es la parte de la BD que es relevante para cada usuario. En la vista de usuario aparecerán sólo aquellas entidades, relaciones y atributos que le sean de interés. Esta información estará representada de forma que le interese al usuario (ej: fecha vs. día/mes/año), por lo que cada usuario (aplicación) percibirá la información de manera distinta. Además, también se podrán acceder datos que no estén en la BD pero que se calculen a partir de los que sí hay (ej: edad, ventas totales…).

EL NIVEL CONCEPTUAL

Proporciona una visión global de los datos. Para ello, hace uso de una estructura lógica: *qué datos* están almacenados y *qué relaciones* hay entre ellos.
Este nivel representa a) todas las entidades, atributos y relaciones (contenedores de información), b) las restricciones que afectan a los datos (para garantizar la consistencia) (ej: precio>0) c) información *semántica* sobre los datos y d) información de seguridad e integridad.
Además, da soporte a la vista externa.
Este nivel *no* debe contener ningún detalle de almacenamiento (función del nivel interno).

NIVEL INTERNO

Es la representación física de a BD en el ordenador. Recoge cómo están almacenados los datos y trata de hallar el rendimiento óptimo del sistema.

Este nivel representa a) estructuras de datos (de SGBD), b) organizaciones en ficheros, c) comunicación con el SO para gestionar el uso de unidades de almacenamiento y d) compresión de datos, cifrado…

Notar que parte de las responsabilidades de este nivel las realiza el SO. Esto se debe a que, a su vez, el nivel interno se subdivide en nivel físico (gestionado por el SO) y en nivel interno (gestionado por SGBD). No hay una divisó clara entre estos dos subniveles, pues depende de cada SGB y de cada SO.

EJEMPLO: GESTIÓN DOCENTE UNIVERSITARIA

Ítem básico: PROFESOR

- Identificado por: Número de registro personal (NPR)
- Caracterizado por: Nombre y apellidos, Sueldo, Departamento al que pertenece

Visión conceptual:

```
Profesor = registro de
  NRP campo alfanumérico de 10 caracteres,
  Apellidos campo alfanumérico de 30 caracteres,
  Nombre campo alfanumérico de 20 caracteres,
  Sueldo campo decimal de 8+2 dígitos,
  Departamento campo alfanumérico de 30 caracteres
fin Profesor.
```

Visión externa 1 (sin depto):

```
TYPE Profesor IS RECORD (
   NRP VARCHAR2(10),
   Apellidos VARCHAR2(30),
   Nombre VARCHAR2(20),
   Sueldo NUMBER(8,2)
);
```

Visión externa 2 (sin sueldo):

```
TYPE Profesor = RECORD
   NRP : STRING[10];
   Apellidos : STRING[30];
   Nombre : STRING[20];
   Departamento : STRING[30];
END;
```

Visión interna:

```
Profesor_interno BYTES=74
   NRP TYPE=BYTES(10),OFFSET=0
   Apellidos TYPE=BYTES(30),OFFSET=10
   Nombre TYPE=BYTES(20),OFFSET=40
   Sueldo TYPE=WORD(2),OFFSET=60
   Departamento TYPE=BYTES(10),OFFSET=64.
```

EL ADMINISTRADOR DE LA BASE DE DATOS (DBM/DBA)

El DBM es una figura de primordial relevancia en el contexto de los SGBDs. Sus responsabilidades son, y constan de:

1. Elaboración del esquema conceptual:
- Análisis de las necesidades de información de la empresa
- Identificación de los datos operativos
- Elaboración del esquema lógico
- Implantación del esquema conceptual
- El DBM suele delegar esta tarea en equipos de consultoría, aunque echa el último vistazo

2. Decidir la estructura de almacenamiento en el nivel interno:
- Realizar el esquema interno
- Diseñar e implementar las correspondencias conceptual/interno asociadas
- El DBM suele ocuparse de esta tarea

3. Conexión con usuarios:
- Análisis de requerimientos por cada uno de los usuarios
- Diseñó lógico
- Codificación del esquema externo y de las correspondencias externo/conceptual
- El DBM suele ocuparse de esta tarea

4. Definir las restricciones de seguridad
- Establecer reglas, tanto genéricas como específicas
- Incluir, si es posible, la integridad en el esquema conceptual
- El DBM suele ocuparse de esta tarea: las restricciones de seguridad son tan importantes como lo son los contenedores
- El DBM suele ocuparse de esta tarea

5. Definir e implantar la política de seguridad:
- Gestión de los distintos usuarios
- Gestión de sus privilegios
- El DBM suele ocuparse de esta tarea

6. Definir e implantar la estrategia de recuperación frente a fallos:
- Los SOs y los SGDBs suelen incorporar facilidades para afrontar los fallos, como SGDBs redundantes o RAID (Redundant Array of Inexpensive Disks).
- El DBM puede y debe realizar copias de seguridad de la BD, en función de lo críticos que sean los datos que gestiona
- También ha de desarrollar políticas de gestión de transacciones (ej: qué pasa si una no se va la luz en medio de una)
- A veces, los datos se almacenan por replicado en distintas zonas geográficas
- El DBM suele ocuparse de esta tarea

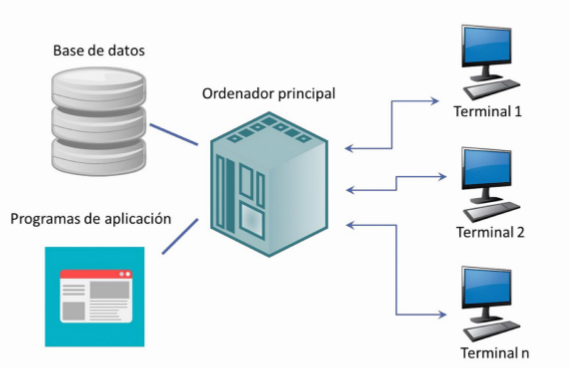
7. Optimización del rendimiento
- Organizar los datos físicamente para mejorar los tiempos de acceso y consulta
- Liberar espacio no utilizado
- Reorganizar las operaciones para que se ejecuten de forma más rápida
- Determinar la necesidad de nuevos recursos hardware
- Establecer prioridades en el uso de los recursos
- El DBM suele ocuparse de esta tarea

8. Monitorizar el SGBD:
- Seguimiento continuo de la actividad del sistema para comprobar que todo funciona correctamente. Para ello:
  + Revisa el acceso de los usuarios a los diversos recursos de la BD
  + Comprueba los niveles de uso de los sistemas de almacenamiento
  + Evalúa la eficiencia con la que se realizan las operaciones
- El DBM suele ocuparse de esta tarea

TIPOS DE ARQUITECTURA DE IMPLANTACIÓN

El concepto de SBD ha evolucionado bastante, paralelamente al desarrollo de la informática, en términos de: forma de gestionar la información, forma de ejecutar los programas y forma de interactuar con el usuario. Estudiamos los distintos planteamientos que se han dado a lo largo de la historia:

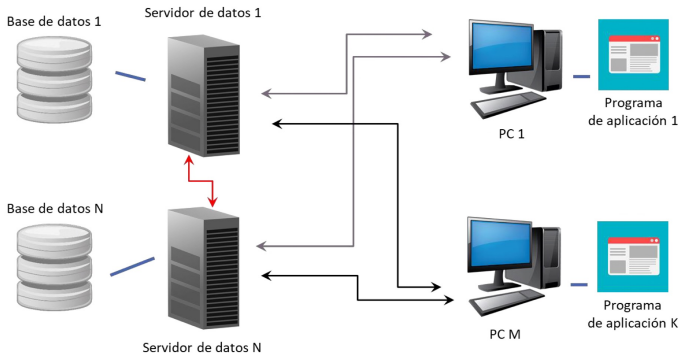
Inicialmente: Arquitectura centralizada
- Toda carga de gestión y procesamiento de información recaía en servidores centrales.
- El usuario accedía mediante terminales (que pedían al ordenador principal que ejecutase ciertas aplicaciones y que gestionase la BD)
- El ordenador principal albergaba el SGBD y los programas de aplicación (era un súper ordenador donde se instalaba todo)
- El problema de esta propuesta es que el súper ordenador es un bicharraco y es muy caro, por lo que era necesaria otra arquitectura (aparte de que es difícil de configurar)



Cuando aparece el PC, la solución que se idea es desplazar la ejecución de los programas usuario y la interacción hasta los PCs. Esto reduce considerablemente la complejidad y, por tanto, los costes en hardware (nos despedimos del súper ordenador).

Posteriormente: Cliente/Servidor

- El papel de servidor lo ejecuta, naturalmente, el servidor de la BD, que ejecuta un servicio de escucha de peticiones
- Los clientes son los PCs conectados en red con el servidor, que albergan los programas de aplicación y que disponen de un servicio de enlace cliente que interactúa con el servicio de escucha del servidor
- Con el desarrollo de redes de comunicaciones, se implementa un enfoque distribuido para los servidores
- El problema de esta propuesta es el alto coste de mantenimiento de los PCs: para instalar, configurar o actualizar una aplicación, tengo que ir PC por PC



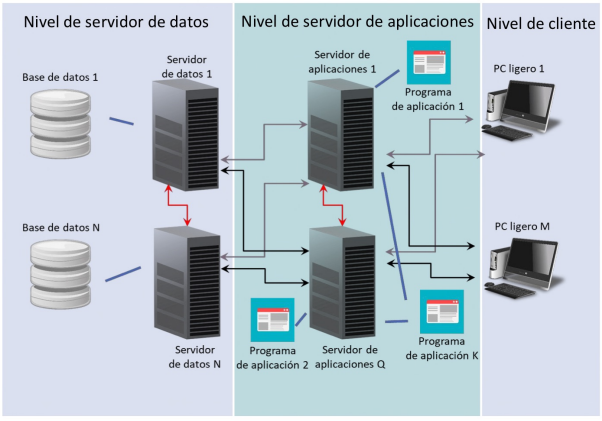
La solución a esto consiste en establecer una separación en las aplicaciones: parte que interactúa con el usuario (interfaz de usuario) vs. parte de ejecución lógica del programa.

Actualmente: Arquitectura articulada en tres niveles de procesamiento

1. Nivel de servidor de datos:
+ Puede estar distribuido
+ El SGBD permite organizar la información de la empresa como una BD global
+ Las peticiones de datos formuladas desde una sede se traducen de forma transparente a peticiones en las sedes donde se encuentran esos datos

2. Nivel de servidor de aplicaciones: son evoluciones de Servidores Web que proporcionan los programas de aplicación a clientes ligeros, que disponen de ejecución de aplicaciones. Para ello, hacen uso de estándares, protocolos de red TCP/IP, protocolo HTTP… . La parte más potente de la aplicación se ejecuta en el servidor y la otra en el PC.

3. Nivel de cliente: consta de PCs ligeros dotados de configuraciones basadas en estándares abiertos. En muchos casos se pueden ejecutar las aplicaciones desplegadas en un navegador web con soporte de ejecución de código JavaScript y html avanzado.
El concepto de este nivel está basado en el carácter portable con el que se distribuyen las aplicaciones desde los servidores de aplicaciones. Consigue generar menos dependencia del hardware y del SO a la hora de ejecutar aplicaciones.



Ventajas de esta arquitectura:

+ Se reduce significativamente el mantenimiento de los clientes: la instalación, configuración y actualización de las aplicaciones se realiza en el servidor, no en cada cliente

+ Mayor facilidad y flexibilidad para el usuario: puede accedes desde casi cualquier puesto y/o dispositivo

Inconvenientes de esta arquitectura:

+ Se desarrolla una mayor complejidad tanto en la configuración y administración de los servidores de aplicaciones, como en el desarrollo de las aplicaciones conforme a este modelo distribuido.

Ejemplo:

1. Usuario del PC invoca desde el navegador la ejecución de una aplicación a través de una URL.

- La parte de interfaz de usuario de la aplicación se puede distribuir como:
  + Un applet Java que se ejecuta en la máquina virtual del navegador.
  + Una serie de paginas HTML generadas desde el servidor de aplicaciones (Servlets, JSP, ASP)

2. La interacción del usuario a través de esta interfaz determina la interacción con la parte lógica de la aplicación que se ejecuta en el servidor de aplicaciones:

- Peticiones de procesamiento.
- Acceso a datos de la BD.
- Generación de nuevas páginas o evolución del applet que ofrecen la respuesta al usuario a través de la interfaz de usuario.

- FIN DEL TEMA 2 -