TEMA 2: ARQUITECTURA DE UN SGBD

ÍNDICE

- Una arquitectura con tres niveles
- Correspondencias entre niveles
- Lenguajes de una BD
- Enfoques para la arquitectura de un SGBD
- El administrador de la BD

1. Una Arquitectura con 3 Niveles

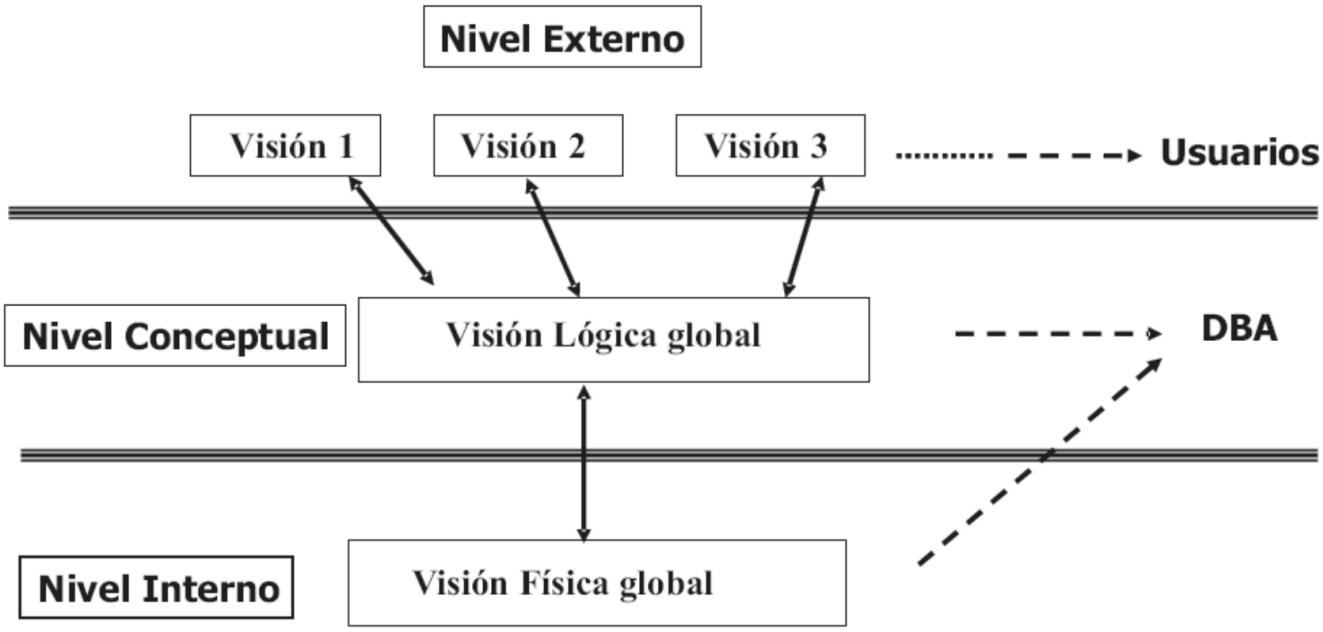
¿Por qué organizar en niveles?

- Los usuarios pueden acceder a los mismos datos, pero desde distintas **perspectivas**: Si un usuario cambia la forma de ver los datos no influye en el resto.
- La organización global de los datos puede cambiarse sin afectar a los usuarios -> **independencia lógica**.
- Los usuarios no tienen por qué gestionar **aspectos relativos a la representación física de los datos**: El administrador de la BD puede cambiar la forma de representar los datos sin influir en los usuarios.

Esta propuesta fue propuesta por el grupo de estudio en sistemas de administración de bases de datos de ANSI/SPARC. Los tres niveles que estipula la arquitectura **ANSI/SPARC** son:

- **Nivel interno:** Constituye la representación de la BD más cercana a la estructura de almacenamiento físico. Por tanto, es la capa donde se establece la forma en la que se implantan las estructuras de datos que organizan los niveles superiores. Además, es donde se realiza todo lo referido a la gestión de ficheros que tiene que haver el SGBD. Esto es, los datos se tienen que almacenar en ficheros que gestiona el nivel interno.
- **Nivel conceptual:** Supone una abstracción global de la BD que integra y aglutina todas las percepciones que los usuarios tienen de ella, cómo

representar los datos de una forma lógica, conceptual e inteligible por el humano. Aquí entra el diseño E/R, el paso a tablas.
- **Nivel externo**: A este nivel se definen todas las percepciones particulares de la BD por parte de los usuarios. Cada usuario puede tener si propia visión de la BD. Se usan los datos por parte de un usuario o una aplicación.



Nivel externo: Parte de la BD que es relevante para cada usuario -> Sólo aquellas entidades, relaciones y atributos que le son de interés. Están representadas de la forma que le interesa al usuario:

- Ejemplos:
  - Nombre completo o nombre y apellidos.
  - Fecha o día, mes y año, etc.

Datos calculados a partir de los que hay como la edad, las ventas totales, etc.

Nivel Conceptual: Visión global de los datos. Representa la estructura lógica de los datos -> qué datos están almacenados y qué relaciones hay entre ellos.

- Representa:
  - Todas las entidades, atributos y relaciones.
  - Las restricciones que afectan a los datos.
  - Información semántica sobre los datos.
  - Información de seguridad y de integridad.

- Da soporte a cada vista externa.
- No debe contener ningún detalle de almacenamiento.

Nivel interno: Representación física de la BD en el ordenador -> cómo están almacenados los datos. Busca el rendimiento óptimo del sistema.

- Representa:
  - Estructuras de datos.
  - Organizaciones en ficheros.
  - Comunicación con el SO para gestionar el uso de unidades de almacenamiento.
  - Compresión de datos, cifrado ...
- Parte de las responsabilidades de este nivel las realiza el SO:
  - Nivel físico.
  - No existe una división clara, depende de cada SGBD y de cada SO.

Ejemplo: Gestión docente universitaria

Item básico **PROFESOR**

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

Visión conceptual: tiene la siguiente estructura:

```
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:

Visión externa 1

- Gestión de personal.
- Lenguaje A.

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

Visión externa 2

- Ordenación académica.
- Lenguaje B.

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

Visión interna: la sintaxis del lenguaje interno podría ser:

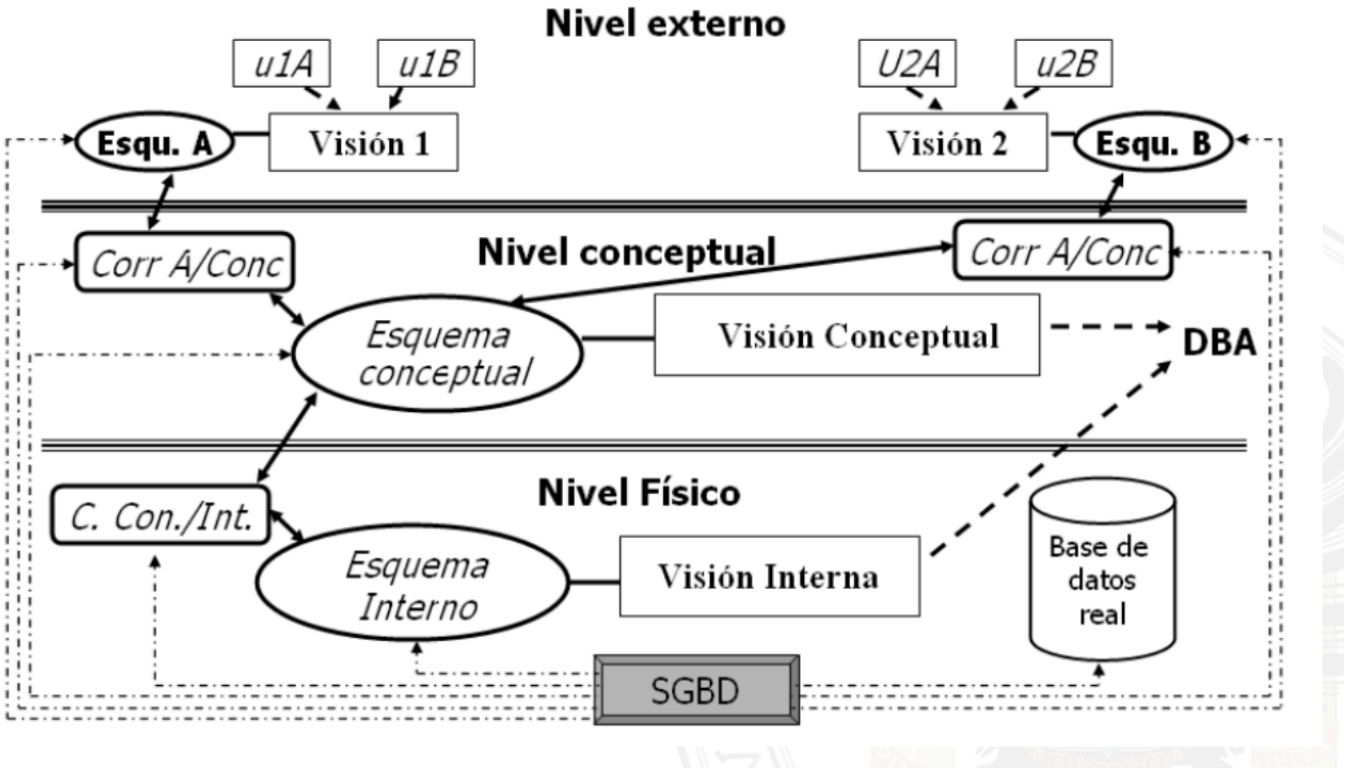
```
Profesor_interno BYTES=74

   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.
```

2. Correspondencia entre niveles

Definición: 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 lógica y física. Tenemos varios tipos:

- ***Transformación conceptual-interna*** : establece cómo se organizan las entidades lógicas del nivel conceptual en términos de registros y campos que se almacenan en el nivel interno. Tiene independencia física, esto es, cuando hay un cambio en el nivel interno o se cambia la correspondencia no varía en el nivel conceptual.

- ***Transformación exxtrena-conceptual:*** describe el esquema externo dependiendo del esquema conceptual subyacente. Tiene independencia lógica, es decir, cualquier cambio en el nivel conceptual o en la correspondencia no influye en el nivel externo. No siempre es posible.
- ***Transformación externa-externa***: algunos SGBDs permiten realizar esquemas externos según otros esquemas externos. Tiene independencia lógica, es decir, cualquier cambio de nivel externo subyacente o en la correspondencia no influye en el nivel externo del que depende.



3. Lenguajes de una BD

Las correspondencias deben escribirse en un lenguaje inteligible por el SGBD, que suele ser una variante del DSL. Se recomienda que la base de datos proporcione formas de guardar dichas correspondencias como metainformación para el SGBD.

Las correspondencias externa/conceptual y externa/externa suelen ser descritas por el usuario usando DDL, en cambio, la correspondencia conceptual/interna se suele construir a partir d elas definiciones del DBA y su construcción suele estar implementada en el SGBD.

Recomendación ANSI/SPARC

La arquitectura ANSI/SPARC recomienda que los SGBD tengan un lenguaje específico para trabajar, el DSL, que está implementado en el propio SGBD y se subdivide en:

- **Data Definition Language (DDL)**: sublenguaje de definición de datos, destinado a la deifinción de ED y esquemas en la BD.
- **Data Manipulation Language (DML)**: sublenguaje de manipulación de datos, mediante el que podemos introducir datos en los esquemas, modificarlos, eliminarlos y consultarlos, además de permitir consultar la estructura de los esquemas definidos en la BD.
- **Data Control Language (DCL)**: sublenguaje de control de datos, que permite gestionar los requisitos de acceso a los datos y otras tareas administrativas sobre la BD.

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

Frente a las recomendaciones de ANSI/SPARC, la industria ha seguido caminos diferentes → se ha favorecido la adopción de lenguajes de datos diferentes que no incorporan una separación estricta entre niveles ni de sublenguajes. Ejemplo: SQL y sus estandarizaciones: SQL89, SQL92, SQL3. Los SGBD han ido proporcionando soporte.

Los lenguajes anfitriones son los que sirven para hacer aplicaciones para la BD. Pueden ser lenguajes de propósito general (C, Java, Pascal,...) o herramientas específicas para bases de datos (Developer de Oracle). Proporcionan procesamiento avanzado de datos y gestión de la interfaz de usuario.

Estos lenguajes deben disponer de herramientas para trasladar la estructura de datos del DSL a la ED propia de cada lenguaje de aplicación y lo mismo para las operaciones DSL y las soportadas por cada lenguaje anfitrión. Esta característica se llama acoplamiento y hay dos categorias: acoplamiento débil y acoplamiento fuerte.

- **Acoplamiento débil:** Son aquellos que están compuestos generalmente por lenguajes de propósito general en los que el programador distingue entre sentencias del lenguaje anfitrión y sentencias dispuestas para acceder a la BD a través del DSL. Algunas alternativas para implementar el acoplamiento débil son:
    - ***Utilización de APIs de acceso a BD:*** el SGBD proporciona una biblioteca binaria y una API para acceder a la BD desde el código fuente del programa.
    - ***DSL inmerso en el código fuente del lenguaje anfitrión***: el programador escribe un código en híbrido, una mezcla de sentencias de lenguaje anfitrión con sentencias DSL, un preprocesador lo transforma a código

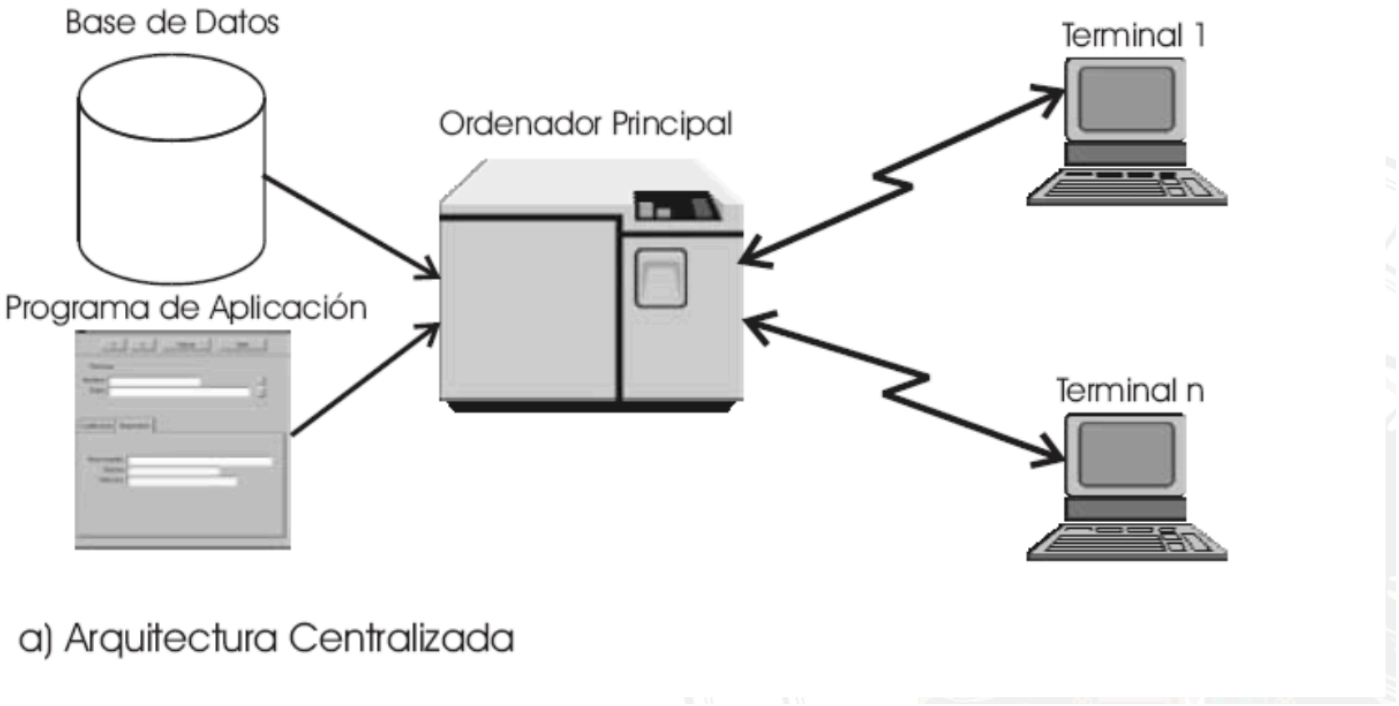
fuente con llamadas a la API, se compila y se enlaza con la biblioteca de acceso a la BD. Como ejemplos, tenemos SQL inmerso en C, SQLJ, etc.

- **Acoplamiento fuerte:** Aquellos que tienen lenguajes y herramientas de propósito específico (Developer Oracle, Oracle APEX). Se parte del DSL como elemento central y se le incorporan características procedimentales para facilitar el desarrollo de apliaciones. Para implementar este acoplamiento__
  - Diversas propuestas (la mayoría propietarias) como PL/SQL de Oracle.
  - Podemos ejecutar Java sobre una máquina virtual implantada en el propio SGBD.
  - Han aparecido numerosos entornos de desarrollo específicos para el desarrollo de aplicaciones de gestión: diseñadores de informes, de formularios.

4 . Enfoques para la arquitectura de un SGBD (falta por ver)

Arquitectura centralizada

Inicialmente, el usuario a través del terminal enviaba peticiones al servidor. El servidor contenía los programas de aplicación y las bases de datos necesarias para cubrir dichas peticiones. En el servidor era donde se realizaba la carga tanto de procesamiento de informacion como de ejecución, mientras que el terminal sólo enviaba peticiones.



a) Arquitectura Centralizada

Problema:

- Elevado coste de los ordenadores principales.
- Aparece el PC con precios contenidos.

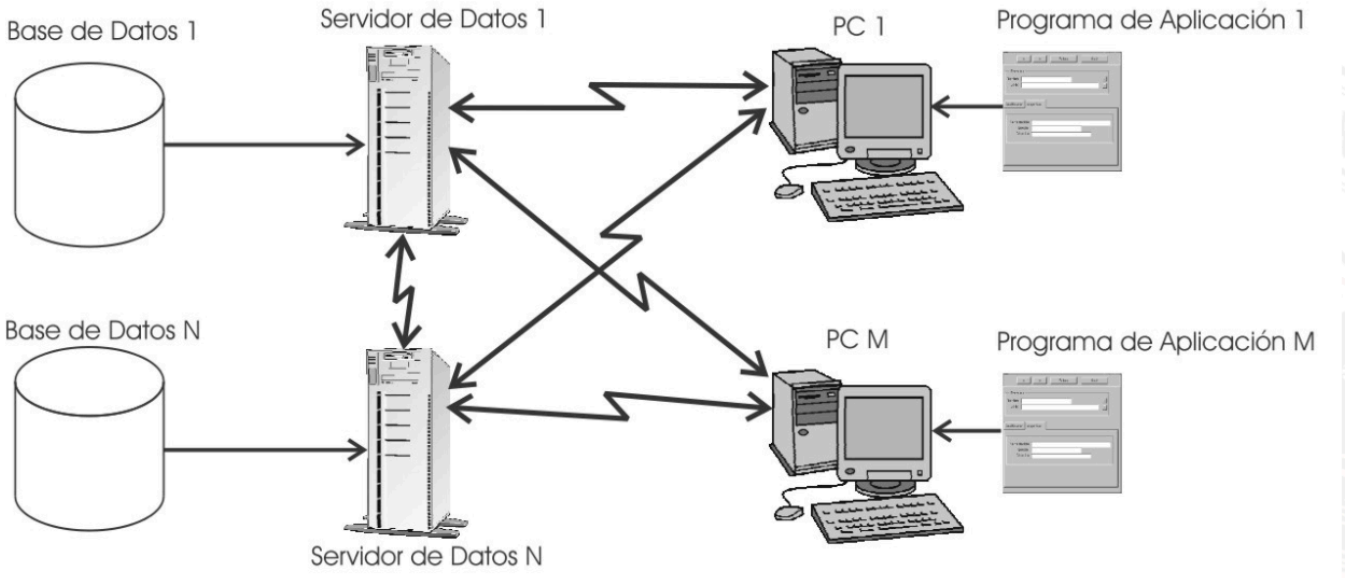
Solución:

- Desplazar la ejecución de los programas de usuario y la interacción hasta los PCs.
    - Reducción de costes en hardware.
    - Disposición de terminales con mayor capacidad.

Arquitectura distribuida

La arquitectura cliente/servidor se divide en 2 partes:

- *Servidor*: se compone de múltiples servidores que contienen las bases de datos. Los servidores escuchan las peticiones de los programas de apliación y se comunican con ellos, enviándoles los datos necesarios.
- *Cliente:* Los PCs contienen los programas de aplicación, están conectados en red con el servidor de datos e interactúan con ellos. En los PCs es donde se realiza la ejecución de los programas y sólo se comunica con los servidores para el envío o reciibmiento de datos, es en los PCs donde recae ahora toda la carga de ejecución y procesamiento de datos.



b) BD Distribuida y programas de aplicación en arquitectura Cliente/Servidor

Problema:

- Alto coste de mantenimiento de los PCs:
    - Instalación.
    - Configuración.

    - Actualización.

Solución:

- Separar en las aplicaciones:
    - Parte que interactúa con el usuario: interfaz de usuario.
    - Parte de ejecución lógica del programa.

Actualmente:

Arquitectura articulada en tres niveles de procesamiento

Surge esta solución con 3 niveles de procesamiento:

Nivel de servidor de datos:

- Posiblemente distribuido geográficamente.
- 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.

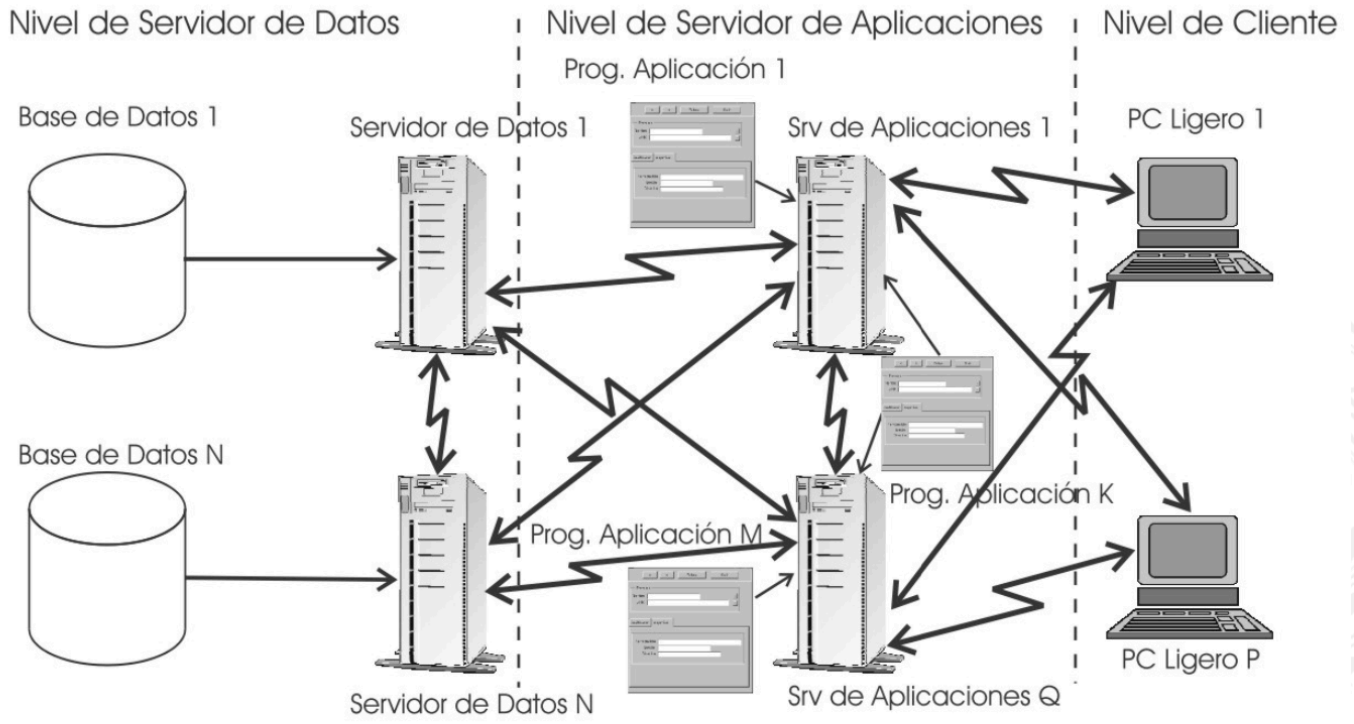
Nivel de servidor de aplicaciones:

- Son evoluciones de servidores Web que proporcionan los programas de aplicación a clientes ligeros, que disponen de entornos de ejecución de aplicaciones:
    - Usando estándares.
    - Protocolos de red TCP/IP.
    - Protocolo HTTP.
    - Despliegue de Applets Java a ejecutar en navegadores con

  soporte de máquina virtual Java.
    - Servlets, JSP, ASP, etc.

Nivel de cliente:

- 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.
- Basados en el carácter portable con que se distribuyen las aplicaciones desde los servidores de aplicaciones.
- Menos dependencia del hardware y del SO a la hora de abordar la ejecución de las aplicaciones.



c) BD Distribuida y programas de aplicación en arquitectura de tres Capas

Ventajas:

- Reducción significativa de costes en cuanto al mantenimiento de los clientes: instalación, configuración y actualización de las aplicaciones realizada en el servidor, no en cada cliente.
- Mayor facilidad y flexibilidad para el usuario. Puede acceder desde casi cualquier puesto y a veces desde distintos dispositivos: móviles, tablets, portátil, PC, etc.

Inconvenientes:

- Mayor complejidad en:
    - La configuración y administración de los servidores de aplicaciones.
    - El desarrollo de las aplicaciones conforme a este modelo distribuido.

5. El administrador de la BD

El DBA es una figura de primordial relevancia en el contexto de los SGBDs

- Elaboración del esquema conceptual, que conlleva:
    - Análisis de las necesidades de información.
    - Identificación de los datos operativos.

    - Elaboración del esquema lógico.
    - Implantación del esquema conceptual.
- Decidir la estructura de almacenamiento en el nivel interno:
    - Esquema interno.
    - Correspondencia conceptual/interna asociada.
- Conexión con usuarios:
    - Análisis de requerimientos.
    - Diseño lógico.
    - Apoyo al desarrollador de aplicaciones: Codificación del esquema externo, correspondencias externo/conceptual.
- Definir las restricciones de integridad:
    - Establecer reglas: genéricas y específicas.
    - Incluir, si es posible, la integridad en el esquema conceptual.
- Definir e implantar la política de seguridad:
    - Gestión de usuarios.
    - Gestión de privilegios.
- Definir e implantar la estrategia de recuperación frente a fallos:
    - Los SOs y los SGBDs suelen incorporar facilidades para afrontar los fallos:
        - SGBDs redundantess
        - RAID - Redundant Array of Inexpensive Disks.
    - El DBA puede y debe realizar copias de seguridad de la BD.
    - Políticas de gestión de transacciones.
- Optimización del rendimiento:
    - 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.
- Monitorizar el SGBD:
    - Seguimiento continuo de la actividad del sistema:
        - Auditar el acceso de los usuarios a los diversos recursos de la BD.
        - Comprobar los niveles de uso de los sistemas de almacenamiento.
        - Evaluar la eficiencia con que se realizan las operaciones.