Bases de Datos - Conceptos

1- **Sistema de Bases de Datos**

Básicamente es un sistema cuya finalidad es almacenar información y permitir a los usuarios recuperar y actualizar esa información a partir de peticiones. Desde una visión muy simplificada podemos decir que una Base de datos nos permite:

- Crear archivos en la base de datos
- Eliminar archivos existentes en la base
- Incorporar nuevos datos a archivos existentes en la base
- Consultar datos de los archivos existentes
- Actualizar datos de los archivos existentes
- Eliminar datos de los archivos existentes

Que es una Base de Datos?

> Una BD es un conjunto de datos persistentes e interrelacionados que es utilizado por los sistemas de aplicación de una empresa, los mismos se encuentran almacenados en un conjunto independiente y sin redundancias.

2- **Componentes de un sistema de base de datos:**

2.1- **Los datos**

La información en una BD esta integrada y compartida. Integrada ya que es la unificación de distintos archivos de datos y en consecuencia elimina las redundancias. Compartida porque son varios los usuarios que acceden a la información en forma concurrente o no, con diversos objetivos.

2.2- **La tecnología**

El equipamiento constituido por las unidades de almacenamiento (discos rígidos, disks arrays, etc.), los procesadores y la memoria principal, más el sistema operativo sobre el que corre el sistema, componen el conjunto de recursos utilizados por el Sistema de Base de Datos.

2.3- **Los programas**

Se refiere a los programas que componen el concepto de DBMS (Data Base Manager System) o motor de base de datos; con el objeto de abstraer a los usuarios de los detalles tecnológicos, brindándoles los servicios que tiene como objetivo. El DBMS ofrece a los usuarios una percepción de la base de datos que está en cierto modo por encima del nivel del hardware y que maneja las operaciones del usuario expresadas en el nivel más alto de percepción.

El DBMS es el componente más importante del sistema de BD pero no el único. Los demás software son entre otros herramientas de desarrollo, librerías de funciones, frameworks de desarrollo, generadores de reportes, entre otros.

El término DBMS o Motor de Base de Datos se usa para referirse a un producto determinado de un fabricante en particular: IBM, Oracle, Sysbase, Microsoft, etc.

2.4- **Los usuarios**

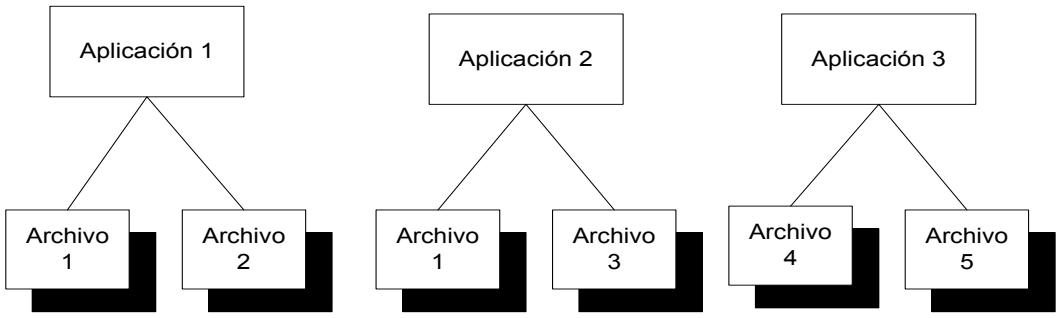
Podemos dividirlos en tres grupos:
El ***programador de aplicaciones***, que accede a los servicios brindados por el DBMS, mediante la inclusión de las solicitudes correspondientes en el código de los programas. Estos programas pueden ser batch u on-line.

El ***usuario***, que utiliza los programas on-line que les suministraron los desarrolladores, u otras interfaces donde puedan ingresar sus consultas en lenguaje SQL, por ejemplo. (SQL es Structured Query Language, aunque no necesariamente se hacen solo consultas).

El ***DBA*** *(DataBase Administrator) es la persona que tiene como responsabilidad Crear, alterar o borrar las distintas estructuras de una base de datos. Sus funciones se verán en las próximas clases.*

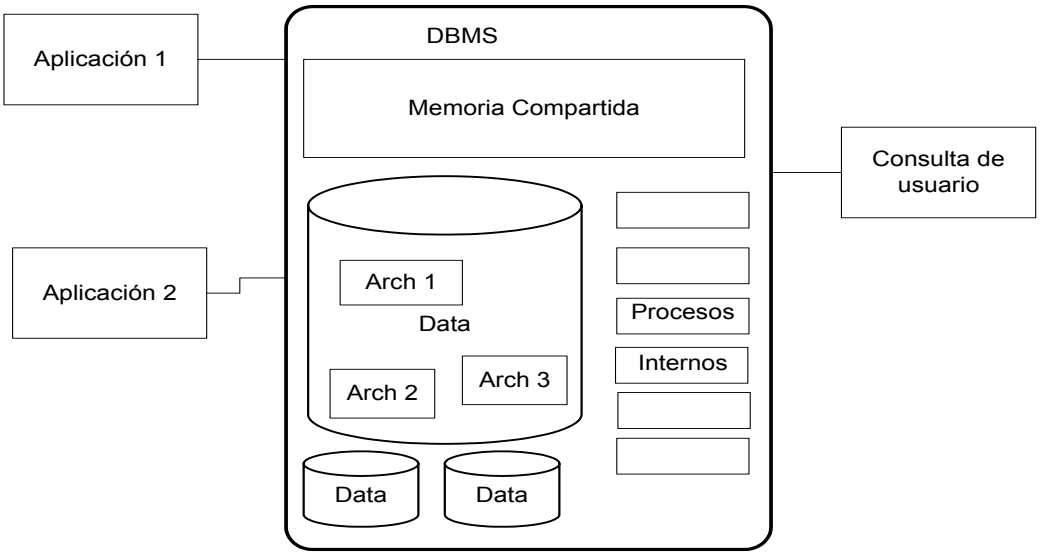
3- **Aplicaciones tradicionales vs. Enfoque de bases de datos**

Enfoque orientado a la aplicación



Cada aplicación es "dueña" de sus datos.
En el ejemplo, supongamos un sistema simple que consta de tres programas o aplicaciones a la vista del usuario. Cada programa manipula a nivel físico los archivos, peticionando al S.O. las operaciones de lectura/escritura sobre los mismos. El desarrollador debe tener en cuenta durante el diseño y construcción factores relativos tales como: concurrencia entre la aplicación 1 y 2, ya que ambas utilizan el Archivo 1.

Enfoque orientado a los datos o enfoque de base de datos.



En este ejemplo los archivos de datos (tablas) residen el la Base de Datos, y los programas o consultas de usuarios son enviadas mediante algún mecanismo de comunicación estándar al DBMS, que es quién realiza realmente la consulta/actualización y devuelve el resultado al programa. El DBMS se encarga entre otras cosas de la seguridad, concurrencia, etc. Las peticiones no se hacen al S.O. sino al DBMS mediante instrucciones SQL.

Una persona (que en realidad representa a la empresa) es "dueña" de los datos. Este es el administrador de datos. Es quién define que datos se van a colocar en la base y que políticas de seguridad se pueden aplicar a los datos. Esta política será aplicada por el DBA, que es el técnico responsable por la operación y mantenimiento de la base.

Ventajas del enfoque de bases de datos

- Es posible aplicar una **política de seguridad**: ofrece a la organización un sistema de control centralizado de su información. Al tener a su mando la totalidad de los datos , el DBA puede asegurarse de que solo se acceda a los datos según lo establecido.
- Es posible hacer cumplir **normas** aplicables a la representación de los datos : tienen que ver son los tipos de datos utilizados, las normativas para nombrar y documentar datos.
- Es posible disminuir la **redundancia** : en el otro enfoque, cada aplicación tiene sus propios archivos, por lo que el mismo dato real puede estar replicado varias veces en distintos archivos. Esto genera un costo de espacio adicional. Sin embargo, hay veces que no se desea eliminar toda la redundancia por razones de performance. Por ejemplo, un campo ‘TOTAL_FACTURA’. Otro ejemplo, una base replicada totalmente en un segundo servidor.
- Es posible **evitar la inconsistencia**. En realidad es una consecuencia del punto anterior. Al estar un dato almacenado en distintos lugares, voy a necesitar mantener actualizadas todas las copias existentes del mismo dato. Si no lo hago así, genero inconsistencias. En un enfoque de base de datos centralizado, es posible (hasta cierto punto), eliminar las inconsistencias haciendo que exista una única entrada para un dato determinado, o que si bien el dato se encuentre redundante por motivos de performance, la actualización se realice en forma consistente.
- Es posible **compartir los datos**. No solo las aplicaciones existentes comparten la información (debido a las ventajas del control, la no redundancia y la consistencia), sino que también puedo crear nuevas aplicaciones que utilicen esos datos.
- Es posible **mantener la integridad**. Aunque se eliminen las redundancias, aun la base de datos puede contener información errónea. A esto se refiere el concepto de integridad : por ejemplo, que no se encuentren registradas ventas

de artículos que no existen, entradas de alumnos sin legajo, etc. El DBA puede definir reglas para que la base mantenga automáticamente la integridad.
- Es posible establecer un **criterio para la organización de los datos**, de manera de favorecer al rendimiento de ciertas aplicaciones, al considerarse más importantes que otras.
- Es posible contar con la **independencia de los datos**. Este ventaja se constituye de por sí en un objetivo de los sistemas de bases de datos.

4- **Independencia de los datos**

En el enfoque de aplicaciones tradicionales, los datos y programas están fuertemente acoplados. Los requerimientos de un programa determinan la forma en la que se van a organizar los datos en disco y el método de acceso más apropiado. Además, hay un conocimiento total de la organización de datos y de método de acceso al momento de programar la aplicación, lo que determinará su lógica interna.
Ejemplo 1: una aplicación XBASE, hay un archivo EMPLEADO y quiero hacer un listado por nombre. Debido a ello creo un índice por nombre (en consecuencia, la aplicación me determinó el método de acceso). Luego en el programa, abro el archivo y digo que índice quiero usar: “SET INDEX TO 2”. O sea que tengo que conocer la organización física.
Ejemplo 2: un programa COBOL con la siguiente declaración de estructura de un archivo:

```
01 REG-CLIENTE
  03 COD-CLIENTE  PIC 99999
  03 NOM-CLIENTE  PIC X(30)
  03 NCUIT        PIC 9(11)
  03 SALDO        PIC 9999999.99
```

Si el archivo de datos requiere de nuevos campos, el programa debe ser modificado use o no los nuevos datos, ya que intentará leer un registro de 56 bytes. Por más que se sea lo suficientemente precavido como para dejar espacio adicional, por ejemplo hasta 100 bytes:

```
01 REG-CLIENTE
  03 COD-CLIENTE  PIC 99999
  03 NOM-CLIENTE  PIC X(30)
  03 NCUIT        PIC 9(11)
  03 SALDO        PIC 9999999.99
  03 FILLER       PIC X(54)
```

en algún momento puede no ser suficiente, o bien puede ser que cambie el código de cliente por un PIC más grande.

En los ejemplos precedentes, decimos que la aplicación es dependiente de los datos. Es imposible alterar la estructura de los datos o la técnica de acceso sin afectar los programas.

En un sistema de base de datos, por el contrario nos brinda:

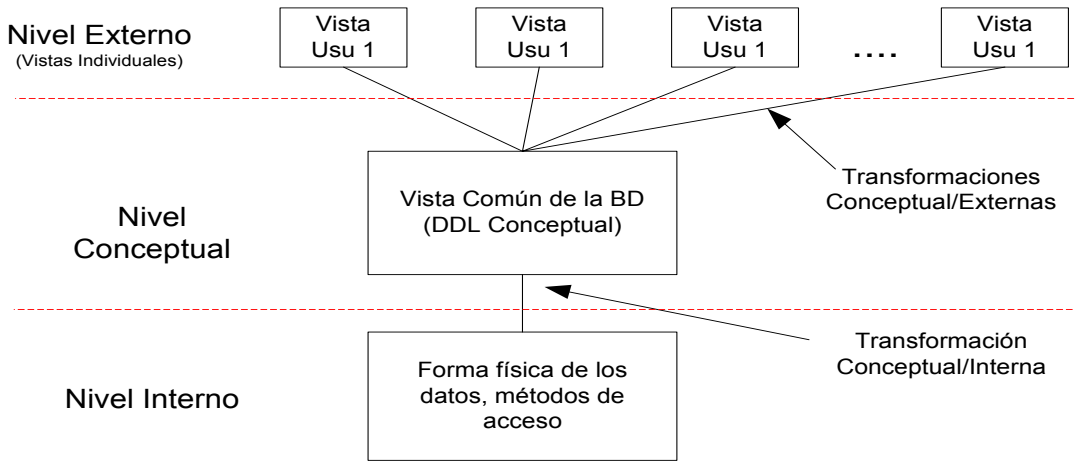
- ***Independencia lógica***. Cada aplicación requiere una vista diferente de los mismos datos. Por ejemplo la aplicación A que requiere un listado de todos los clientes (código y nombre solamente) ordenados por nombre lo solicitara al DBMS. El DBMS resuelve la consulta mediante la lectura de los campos y los registros apropiados y lo entrega a la aplicación "cliente". El programa "A", ni se enteró de la existencia de otros campos en el archivo, ni siquiera como hizo el motor para resolver la consulta en el orden indicado. Puede existir otra aplicación B que requiera la actualización del saldo de una cuenta determinada. A su vez, B también desconoce cual es el orden de los campos, si existen o no otros clientes, y como se llevo a cabo la actualización.
- ***Independencia física***. Es posible modificar la estructura de almacenamiento, la distribución física o la técnica de acceso sin afectar las aplicaciones. En el ejemplo del programa A, puedo crear un índice por nombre para que se resuelva el listado más rápidamente ; en el ejemplo B, puedo crear una tabla de hash. En ningún caso los programas son afectados.

La independencia lograda nunca es absoluta.

5- **Arquitectura de un sistema de bases de datos (ANSI/SPARC)**

Esta Arquitectura fue propuesta por el Grupo de Estudio en Sistemas de Administración de Bases de Datos de ANSI/SPARC, la misma se ajusta bastante bien a la mayoría de los sistemas de BD.

Se divide en tres niveles: interno, conceptual, y externo.



La arquitectura ANSI/SPARC

El nivel interno (FISICO), el más cercano al almacenamiento físico, es el que se ocupa de la forma como se almacenan físicamente los datos.

El nivel externo (VISTAS), es el más cercano al usuario. Se ocupa de la forma como los usuarios individuales percibirán los datos.

El nivel conceptual (LOGICO) es un nivel de mediación entre interno y externo.

Como el nivel externo se ocupa de las vistas individuales de los usuarios, el conceptual se ocupa de una sola vista que las agrupe a todas. Este nivel conceptual es también una representación 'abstracta' de la base de datos, es decir, implica estructuras orientadas hacia el usuario (archivo, registro, campo) y no hacia la máquina (byte, palabra, tamaño de página, tamaño de buffer, etc)

5.1 Nivel externo o VISTAS

Esta es la percepción que tienen los usuarios respecto de la BD. Los usuarios pueden ser programadores, usuarios finales, o el DBA.

Los usuarios utilizarán algún tipo de lenguaje para acceder a los datos. En el caso de los programadores puede ser 4GL, PL/SQL, C, COBOL, VB, etc. Los usuarios finales en su lugar pueden usar algún tipo de interfase especial para acceder a los datos. Sea cual fuere la forma, dentro de estos lenguajes debe haber un *sublenguaje* (por ejemplo embebiendo uno dentro del otro) que se ocupa específicamente de las operaciones con la base de datos.

Estas operaciones son: *conexión*, *definición de datos* (DDL), *manipulación de datos* (DML), *desconexión*.
El ejemplo más común y que hoy se ha impuesto como estándar es el SQL.

La percepción que tenemos como usuarios de una base de datos es solamente una vista externa. El DBA es quien nos define esas vistas. Estas vistas están compuestas por registros lógicos que no necesariamente tienen algún tipo de correspondencia con cada registro físico de un archivo o tabla en el nivel conceptual. Para que esto ocurra, hay una correspondencia entre el esquema conceptual y el esquema externo.

Una vista externa es el contenido de una Base de datos como lo ve algún usuario en particular (para este usuario la vista es la Base de Datos).

5.2 Nivel conceptual o LOGICO

Representa de una forma 'entendible' de toda la información contenida en una base de datos. En lugar de describir datos personales, describe a los datos de toda la organización. Debe ofrecer un panorama de los datos como realmente son, a diferencia de como los usuarios lo ven debido a las limitaciones existentes (lenguaje, hardware, interface, etc.).

La vista conceptual se define mediante un esquema conceptual. Este esquema conceptual se escribe en DDL. Contiene definiciones del contenido de la base, , tipos de datos, restricciones, reglas de integridad, etc.

5.3 Nivel interno o FISICO

En este nivel se define como se almacenan los datos en disco, es una representación de bajo nivel de toda la base de datos. Por ejemplo, se especifican las estructuras de los registros, se definen índices y métodos de acceso, en que secuencia física se encuentran los registros, etc.

Además de los tres niveles la arquitectura comprende ciertas transformaciones:

- Una Transformación conceptual / interna.
- Y Varias Transformaciones Externas / Conceptual.

Transformación Conceptual / Interna.

La transformación conceptual / interna define la correspondencia entre la vista conceptual y la base de datos almacenada, y especifica la representación en el nivel inerno de las filas y columnas del modelo concepcual.

Transformación Externa/Conceptua

Dada una determinada vista externa, esta transformación define la correspondencia entre dicha vista externa y la vista conceptual.
Por ej.: se puede cambiar el tipo de dato a obtener, combinar varios campos conceptuales en un unico registro externo.

6- **Funciones del motor de base de datos y del DBMS en su conjunto**

6.1 Diccionario de datos. Es una base de datos del sistema, que define a los objetos dentro del mismo. Es lo que se llama 'metadatos'. Allí se almacenan las diversas vistas (externas, conceptuales, internas). El diccionario tiene que estar integrado a la base.

6.2 Control de la seguridad. La seguridad implica controlar que los usuarios estén autorizados para hacer lo que intentan. Para ello los DBMS poseen lo que se

llama el catálogo. El catalogo mismo esta formado por entidades e interrelaciones (por lo que en un esquema relacional van a ser tablas). En ellos se almacenan datos acerca de los objetos de la base (tablas, indices, vistas, usuarios, permisos, etc). El DBA puede alterar estos catálogos de manera de otorgar/revocar los permisos necesarios a los usuarios/grupos de usuarios creados en la base. Otro mecanismo que se emplea para la seguridad son las vistas.

6.3 Mecanismos para garantizar la integridad y la consistencia
Constraints: son controles de integridad que se les pueden incorporar a la base.
Por ejemplo, constraints de PRIMARY KEY, integridad referencial, etc.
Triggers: un procedimiento que se ejecuta ante un determinado evento sobre un objeto. Antes o después de UPDATE/DELETE/INSERT.
Transacciones: son una unidad lógica de trabajo. Partiendo de que una transacción lleva la base de datos de un estado correcto a otro estado correcto, el motor posee mecanismos de manera de garantizar que la operación completa se ejecute o falle.
Logical Logs: Es un registro donde el motor almacena la información de cada operación llevada a cabo con los datos.

6.4 Backup y recovery
Utilitarios para realizar backups y recuperación ante caidas. Backups incrementales (solo se guardan las modificaciones a partir de cierta fecha), backup en caliente.

6.5 Optimización del acceso a los datos. Módulos que previo a la resolución de una consulta u otra operación, evalúan cual es la forma más adecuada para llevarla a cabo (cual índice utilizar, si se va a crear una tabla de hash, si se van a examinar todos los registros de un archivo, etc). Para ello se basa en la información estadística existente en los catálogos.

6.6 Administración de la organización física de los datos en disco y/o memoria. Los motores de bases de datos poseen su propia (y compleja) forma de hacer uso de los recursos de disco que el sistema le brinda. Es por ello que entran en juego una serie de consideraciones como ser:

- El tamaño de página.
- El mecanismo de paginación del disco a memoria.
- Los tamaños de los buffers para acceder al disco.
- El tamaño de memoria compartida que va a alocar.
- La administración de esos recursos de memoria.

Si se van a colocar los datos dentro de 'archivos' del sistema operativo (haciendo uso de los FILE SYSTEM de usuario) o si se van a utilizar directamente los recursos a nivel máquina (RAW DEVICE).
Definición de los 'extents', tamaños que se alocan al requerir espacio para un nuevo objeto o la ampliación de uno ya existente.
Distribución física de la base de datos en 'DATAFILES'.

Particionamiento o Fragmentación de las tablas según su lógica interna.

6.7 Mecanismo de conectividad. El motor otorga y lleva un control de sesiones de usuarios. Estos usuarios pueden ser usuarios finales o programas. La conexión a la base puede ser a través del mecanismo de conectividad (red) existente (DEC, TCP/IP), o por SHARED MEMORY.

6.8 Administración y chequeo de los recursos de la base
Poseen una serie de utilitarios para monitorear el uso de los recursos de la base, determinar quienes están accediendo a la base en un momento dado, que se encuentran haciendo, etc.

6.9 Concurrencia en lecturas y actualizaciones
Los distintos usuarios que acceden en un momento dado a la base, deben percibir a la misma en forma consistente, a través de su 'vista'. Como este acceso es simultáneo, los DBMS permiten la concurrencia mediante mecanismos de recuperación de transacciones y bloqueos. Debe permitir detectar (e eliminar) deadlocks. También puede permitir transacciones en forma distribuida.

6.10 Facilidades de auditoría. A los DBMS se les puede activar la opción de guardar en un log un registro del uso de los recursos de la base para auditar posteriormente.

6.11 Logs del sistema. Mediante este tipo de logs, el DBA puede llegar a determinar cual fue, por ejemplo, el problema que produjo una caída del sistema.

6.12 Acomodarse a los cambios, crecimientos, modificaciones del esquema. El motor permite realizar cambios a las tablas constantemente, casi en el mismo momento en que la tabla está siendo consultada.

6.13 Creación de triggers y stored procedures. Con el objeto de fortalecer aún más el concepto de independencia entre datos y programas de aplicación, los motores nos dan la facilidad de incorporar los llamados STORED PROCEDURES. Básicamente, son un conjunto de instrucciones en un lenguaje que el motor entiende, que pueden ser llamados en forma externa desde las aplicaciones programa o usuarios. De esta manera se pueden cambiar los programas, por ejemplo, por una decisión de mejorar la interfase visual, utilizando las reglas del negocio, que están definidas dentro de la base. Poseen la desventaja de ser propietarios.

7- **Algunos ejemplos de enfoques no relacionales.**

- De lista invertida: DATACOM
- Jerárquico : IMS
- Red (CODASYL): IDMS, TOTAL