**ARCHIVOS CONVENCIONALES**

Registro: están compuestos por una serie de caracteres o bytes. Es un conjunto de campos (atributos), donde cada uno de ellos contiene un único valor almacenado. Se alojan en MS y pueden almacenarse de dos formas diferentes: archivos convencionales o BD.

Archivos: es un elemento de información conformado por un conjunto de registros. Pueden organizarse como archivos secuenciales o archivos directos y pueden estar estructurados en varios niveles.

Puntero: es un objeto del lenguaje de programación que contiene la dirección de memoria de un dato o de otra variable que contiene al dato. La obtención del valor almacenado en esa ubicación se la conoce como desreferenciación del puntero. Se pueden utilizar para manipular estructuras de datos, referenciar bloques de memoria y proveer el paso de argumentos por referencias en las llamadas a funciones.

**PILAS**

Es la forma más primitiva de organización. Los datos se recolectan en el orden en que llegan. Cada registro consiste en una ráfaga de datos. El propósito de la pila es simplemente acumular la masa de datos y guardarlo.

Como no hay estructura para el archivo de la pila el acceso a registro es por búsqueda exhaustiva. Si se quiere todos los registros que contienen un campo particular o que tienen un valor determinado para ese campo, debe buscarse en el archivo entero.

Los archivos de pilas se aplican cuando los datos se recogen y almacenan antes de procesarlos o cuando no son fáciles de organizar. Este tipo de archivo usa bien el espacio cuando los datos almacenados varían en tamaño y en estructuras. Además, no se adapta a la mayoría de las aplicaciones.

* Fácil de programar.
* Permite la inserción más rápida, ya que se limita a insertar al final.
* Organización muy primitiva.
* El acceso a los datos es totalmente ineficiente ya que están desordenados.
* Puede ser que provoque redundancia y repetición en la información.

**ORGANIZACIÓN SECUENCIAL**

Lo registros quedan grabados consecutivamente. En la mayoría de los casos, un archivo puede ordenarse ascendente o descendentemente con base a algún campo de cada registro. Este campo es conocido como la llave del ordenamiento. Todos los registros se almacenan por posición: primer registro, segundo registro, etc.

Sólo puede leerse empezando desde el principio del archivo, leyendo un registro a la vez hasta llegar al registro deseado. La escritura de nuevos datos siempre se hace al final del archivo.

* Mejor utilización de espacio y.
* Son rápidos cuando los registros son accedidos secuencialmente.
* Son muy sencillos de usar y aplicar.
* Aplicables a archivos con poca volatilidad y tamaño variable.

* La localización de un determinado registro no se puede hacer individualmente ni de manera rápida.

**ARCHIVOS DIRECTOS**

Permiten acceder directamente a un registro sin tener que buscar uno a uno por todos los registros del archivo, utilizando una llave de acceso dentro del archivo.

**ORGANIZACIÓN RELATIVA**

Consiste en un conjunto de celdas identificadas con un número, las cuales están grabadas en forma contiguas en el disco e identifican al registro que contiene. Las celdas son de longitud fija y pueden o no contener información. En cada celda puede haber uno o más registros. Los registros tienen un tamaño máximo igual al tamaño de la celda.

Hay dos formas de acceder:

1. *Acceso directo*: especificando el número de celda.
2. *Acceso secuencial*: a partir de una determinada celda el siguiente.

**ORGANIZACIÓN DIRECTA**

Los registros se graban en posiciones que dependen del valor de una clave. Se le llama clave a una parte del contenido del registro que permita identificarlo unívocamente. Cuando tenemos un conjunto de registros tenemos un conjunto de claves.

Se utiliza la *función de hashing* para distribuir en forma pareja las claves en el espacio de direcciones. El espacio asignado a un archivo es fijo.

**ORGANIZACIÓN INDEXADA**

Incluye índices en el almacenamiento de los archivos, facilitando la búsqueda de algún registro en particular sin necesidad de ver todo el archivo. Es conveniente para archivos con mediana volatilidad y tamaño relativamente estable.

Cada archivo tiene sus bloques de índices y de datos. Un índice consiste en un listado de los campos claves del archivo, junto con la dirección correspondiente en disco.

* Búsquedas más rápidas.
* Para tener acceso a los datos sólo es necesario traer a memoria el bloque de índices donde está el apuntador a los datos.
* Ocupa más espacio en el disco que los archivos secuenciales.
* Sólo se puede grabar en soportes direccionales.
* Los registros deben ser de longitud fija.

**ORGANIZACIÓN SECUENCIAL INDEXADA**

Los registros se almacenan secuencialmente pero el archivo además tiene un índice. El registro tiene que tener un campo clave que va a servir de índice. Un mismo archivo puede tener más de un campo clave.

El archivo consta de tres áreas:

* Área de índices: archivo secuencial creado por el sistema, en el que cada registro contiene el campo clave y la dirección de comienzo de cada registro.
* Área de datos: contendrá los registros de datos clasificados en orden ascendente por su campo clave.
* Área de excedentes: Se reserva esta área para añadir nuevos registros que no pueden ser colocados en el área primaria cuando se produce una actualización del archivo.

Se puede leer la información de dos modos:

* Acceso directo: mediante el campo clave del registro a localizar.
* Acceso secuencial: los registros son leídos ordenados por el contenido del campo clave, independientemente del orden en que se fueron grabando.

Ventajas y desventajas:

* Dos modos de acceso.
* El sistema se encarga de relacionar la posición de cada registro con su contenido.
* Se pueden actualizar los registros en el mismo fichero.
* Necesidad de espacio adicional para el área de índices y desaprovechamiento de espacios intermedios libres.
* Solo se puede utilizar soportes direccionales.

**BASES DE DATOS**

Es un sistema computarizado cuya finalidad es almacenar información y permitir a los usuarios recuperar y actualizar esa información en base a peticiones.

**Componentes de una BD**

* Datos: Es una entrada del sistema de información. Pueden ser integrados y compartidos.

*Integrados* porque la BD es una unificación de varios archivos con redundancia mínima.

*Compartidos* porque las piezas individuales de la BD son utilizadas por diferentes usuarios al mismo tiempo.

* HW: Unidades de procesamiento y unidades de almacenamiento principal y secundario.
* SW: Abarca el SO y el DBMS.

*Motor de BD*(DBMS): Es una pieza de SW ubicada entre el SO y los programas de aplicación de usuarios. Procesa todas las peticiones ocultando a las aplicaciones los detalles de implementación.

* Usuarios:

*- Programadores*: Codifican aplicaciones para los usuarios finales, las cuales generan peticiones.

*- Usuario final*: Utilizan las aplicaciones desarrolladas para acceder en forma concurrente a la BD.

*- Administrador de BD* (DBA): Personal técnico responsable del funcionamiento y disponibilidad de la BD. Se encarga de manejar y mantener los diferentes esquemas.

*- Administrador de datos* (DA): Persona no técnica responsable de las necesidades de información de la BD. Decide qué datos hay que guardar en la BD.

Tareas del DBA

* *Respaldo*: Backup.
* *Integridad*: Ofrece reglas de integridad.
* *Seguridad*: Brinda controles de acceso físicos (servidores) y lógicos (usuarios) a los datos.
* *Disponibilidad*: Cumple los contratos de servicio definidos.
* *Desempeño*: Ofrece buena performance, rapidez.
* *Soporte*: Brinda asesoramiento de BD.

**Beneficios** con respecto a los archivos.

* Recuperación: Diferencia fundamental con los archivos convencionales. Recuperan la información y su estructura.
* Compactación: Los archivos son administrados por el DBMS que resuelve las problemáticas de duplicación de datos y administración del espacio vacío.
* Velocidad: Las BD son más rápidas para obtener la información. En el código se usarán pocas sentencias para acceder a los datos.
* Productividad: Los trabajadores se concentran en la información de negocio requerida y no en cómo obtenerla.
* Información actualizada: Los cambios se hacen visibles al instante para el usuario.
* Seguridad: Las BD son más potentes ya que permiten mayor granularidad en la regla de seguridad sobre los datos (se le puede dar seguridad a cada campo).
* Manejo centralizado de la información: Los modelos de datos se crean y administran en forma centralizada.
* Integridad: Es el grado de corrección de los datos, coherente entre los datos y con la realidad.
* Transaccionabilidad: Manejar transacciones como unidades atómicas de trabajo.
* Recuperación ante fallos: Las BD están preparadas para reponerse automáticamente ante cualquier fallo de un sistema computarizado.

**Independencia de datos**

Es la inmunidad o robustez que tienen las aplicaciones para soportar cambios en los datos (el usuario no se debe enterar). Hay 3 tipos:

* Independencia de dispositivos: cambios en los almacenamientos físicos. Lo da el HW y el SO.
* Independencia física: cambios en los archivos. Lo brinda el DBMS.
* Independencia lógica: cambios en el modelo de datos. Lo da el DBMS y hay que cumplir el modelo de datos.

**MODELOS DE BD TRADICIONALES**

El modelo determina el modo de organizar, manipular y almacenar los datos. Tiene 3 aspectos:

* *Estructural*: La estructura para la representación de los datos.
* *Manipulación*: Cómo se procesan los datos para obtener la información.
* *Integridad*: Garantizar la corrección de los datos.

**Bases de Datos Jerárquicas**

Almacena la información en una estructura jerárquica que enlaza los registros en forma de árbol, en donde un nodo padre puede tener varios nodos hijo, son relaciones uno a muchos unidireccionales.

Un conjunto de registros con los mismos campos se denomina archivo. Cada nodo del árbol representa un tipo de registro conceptual. A su vez, cada registro está constituido por un número de campos que los describen (propiedades o atributos). Las relaciones entre registros se establecen mediante punteros y están representadas por las ramas.

Ventaja sobre las BD relacionales: el rendimiento. El acceso de un registro a otro es prácticamente inmediato.

Desventajas: El control sobre los datos queda en manos de las aplicaciones. Por lo tanto no se garantiza la inexistencia de registros duplicados y no existe garantía de que un registro hijo esté relacionado con un registro padre válido.

**Bases de Datos de Red**

Los registros están conectados entre sí por medio de enlaces en una red. El enlace es exclusivamente la asociación entre dos registros y cada registro puede tener más de un enlace permitiendo la relación de muchos a muchos.

Las relaciones se representan mediante conjuntos, los cuales son una colección de dos o más tipos de registros vinculados entre ellos.

**MODELO DE BD RELACIONALES**

Sus 3 aspectos son:

* Estructural: Los datos se estructuran solamente en tablas. Son la estructura lógica de la BD, una representación. Se demuestra con la teoría de conjuntos. Una relación representa una tabla, una tupla es una fila de dicha tabla y un atributo es una columna. No existen apuntadores que conecten una tabla con otra. Las conexiones están representadas por la aparición de un valor en dos o más tablas.
* Manipulación: Define operaciones (peticiones) cuyos operando son tablas y sus resultados también, lo que permite encadenar operaciones formando lo que se conoce como “expresiones anidadas”.

Se demuestra con el álgebra relacional y el cálculo relacional.

* Integridad: Define reglas de integridad implementadas como restricciones sobre las tablas. (restricciones de dominio, clave primaria)

**Operaciones** que define el álgebra relacional, en donde todos los resultados son tablas nuevas:

* Proyección: misma tabla pero con menos campos. Requiere un parámetro que es una lista de campos.
* Selección: misma tabla pero con menos registros. Posee un parámetro que es una condición.
* Producto cartesiano: tabla con la suma de los campos de los operandos y que contiene todas las combinaciones posibles entre los registros.
* Reunión: producto cartesiano de los operandos seguido de una selección en donde se igualan los campos del parámetro. (Une las tablas de costado)
* Unión: tabla con los mismos campos que sus operandos pero con la sumatoria de sus registros. (Une las tablas una arriba de otra)
* Intersección: tabla con los mismos campos que sus operandos pero con los registros comunes a ambos.
* Diferencia: tabla con los mismos campos que sus operandos pero con los registros que pertenecen al primero y no pertenecen al segundo.
* División: división de conjuntos de sus operandos.

El DBMS tienen la inteligencia para simplificar las operaciones que se codifiquen, por lo tanto, se puede escribir de la forma que mejor se entienda.

**SQL** (Standard Query Language – Lenguaje de consulta estándar)

Es el lenguaje estándar para trabajar con BD relacionales y es soportado prácticamente por todos los productos en el mercado. Surge en los 70’ y se estandarizó en el año 1992: *SQL 92 (Estándar Internacional de Lenguaje de BD SQL 1992)*, el cual permitió programa para cualquier DBMS.

Las operaciones de SQL son *set at time* que trabaja sobre conjuntos sin recorrer nada. SQL es un lenguaje *host*: está dentro de un lenguaje de programación cualquiera y general del sistema.

**Sentencias**

* DDL (Data Definition Language): Define el modelo de datos.

. Create table

. Alter table

. Drop table

* DML (Data Manipulation Language): Manipula los registros de la tabla.

. Insert

. Update

. Delete

**Orden de ejecución completo:**

6 → SELECT

7 → INTO

1 → FROM

2 → WHERE

3 → GROUP BY

4 → HAVING

5 → ORDER BY

. Select

**Sentencia ‘SELECT’**

Tiene 3 cláusulas fundamentales:

* *SELECT* = Proyección
* *FROM* = Producto cartesiano x
* *WHERE* = Selección

Otras cláusulas:

* **Select DISTINCT**: Elimina duplicados considerando todos los campos.
* **Comodín**: muestra todos los campos.
* **Alias**: Para llamar por otro nombre. Mejora la visualización.
* **Constantes**: Pone un valor fijo para todos los registros.

**Conectores**

*Lógicos*: Unen dos condiciones. [*And, not, or*]

*De inclusión*: Evalúan true/false. [in - not in], [= > < ], [all / any]

**SUBCONSULTA**

Es una sentencia SELECT completa embebida en la cláusula WHERE de otra sentencia SELECT completa utilizando conectores de inclusión. La tabla que genera la subconsulta se desecha. Puede haber varios niveles de subconsulta. Tiene mejor desempeño que el producto cartesiano porque genera tablas más chicas.

Tipos de subconsulta:

* *Anidadas*: no hacen referencia a ningún objeto de la consulta exterior. Primero se ejecuta la subconsulta una vez y luego la consulta exterior una vez. Son independientes y es más eficiente.
* *Correlacionadas*: hacen referencia a objetos de la consulta externa. Primero se ejecuta la consulta externa una vez, y por cada registro a validar en su cláusula WHERE se ejecuta la subconsulta. No son independientes.

**GROUP BY**

Agrupa un conjunto de registros que tienen valores iguales en uno o más campos. Cada grupo representa una fila de la tabla resultante. El objetivo es poder sacar medidas resumen. Es más lento que el distinct.

Funciones de resumen: (count, sum, max, min, avg)

Para extraer medidas resumen de cada grupo. Se utilizan en las cláusulas posteriores en el orden del procesamiento lógico del GROUP BY. El DBMS reconoce las funciones resumen que se escriben en todas las sentencias y las calcula al momento de agrupar.

**EVOLUCIÓN DE LIBRERÍAS DE ACCESO A DATOS**

Mapeadores

DAO

ADO

OBDC

OLEDB

Drivers

Providers

Evolución

3°

2°

1°

* DBLibrary: librerías que proporcionaba directamente el fabricante del DBMS. Cada librería tenía su forma particular de usarse y las aplicaciones no tenían portabilidad.
* ODBC: fabricada por Microsoft cuyo objetivo es la portabilidad en las aplicaciones cliente. Se basa en una capa de librerías genéricas conectadas a una serie de drivers. Estos drivers son los provistos por los fabricantes de DBMS. La programación era de muy bajo nivel.
  + JDBC: (después de ADO) mismas características de ODBC pero para Java y con interfaces de alto nivel. Es un lenguaje interpretado.
* OLEDB: fabricado por Microsoft igual a ODBC pero incorpora la programación orientada a objetos. La programación sigue siendo de bajo nivel. Los drivers se llaman providers pero son lo mismo.
* DAO (Data Access Object): fabricada por Microsoft que encapsula ODBC permitiéndole al programador programar en un nivel más alto.
  + RDO (Remote Data Object): evolución de DAO agregándole características como por ejemplo soporte para BD distribuidas.
* ADO (Active Data Object): Encapsula OLEDB brindándole al programador una interface de alto nivel.
  + ADO.NET: evolución de ADO para la plataforma .NET de Microsoft.
* Mapeadores (objeto relacional): son librerías que funcionan por encima de las librerías tradicionales de alto nivel. Su objetivo es mapear automáticamente los objetos de negocio del programa a las tablas de la BD relacional. Los mapeadores generan automáticamente la sentencia de SQL. Ejemplos: NHibernet (para .net), Hibernet (java), Linq (Microsoft), Entity framework (Microsoft), Data nucleus.

**ARQUITECTURA ANSI-SPARC**

Estándar de diseño. Divide la estructura de la BD en 3 niveles:

* Externo: Administra las diferentes visiones que tienen los diferentes usuarios o roles que utilizan las aplicaciones. Pueden existir varias vistas y cada una abarca solamente una porción del modelo lógico que corresponde al perfil del usuario interesado. Esto brinda seguridad en el acceso a datos. Este nivel es relacional porque trabaja con tablas.
  + MAPPING Externo-Conceptual: Administra la relación entre el modelo de datos conceptual y las diferentes vistas del nivel externo. Garantiza la independencia lógica de datos.
* Conceptual: Administra la estructura lógica de los datos. Mantiene la representación del *modelo de datos* lógico, manejando cuestiones de estructuras (modelo), seguridad (permisos), manipulación (consultas) e integridad. Este nivel también es relacional.
  + MAPPING Interno-Conceptual: Administra la relación entre el nivel conceptual y el interno, es decir, como se distribuyen los registros de las tablas en los diferentes archivos físicos (si se dividen en diferentes discos, se aumenta la velocidad de búsqueda). Garantiza la independencia física de datos.
* Interno: Es el de mayor relación con el SO y contiene la administración de los *archivos convencionales* en MS. Administra la estructura física o interna de los archivos, la organización y los métodos de acceso. No es un nivel relacional ya que trabaja con archivos y no con tablas.

Al definir este nivel se debe tener en cuenta el espacio de almacenamiento:

* Tamaño.
* Tasa de crecimiento.
* Velocidad de acceso: velocidad de discos, concurrencia de discos, distribución de los datos en los archivos (mapping i-c), RAID.

Es responsabilidad del DBA definir y mantener los distintos niveles y mappings, y del DBMA administrar el correcto funcionamiento.

**RAID**: Arreglo de discos físicos. Puedo elegir usar varios discos físicos de modo que el SO vea todo como uno solo. Puede ser de Software (a través del SO) o de Hardware (placas especiales a las que se enchufan los discos).

*Tipos*:

* RAID 0: distribuye los archivos en todos los discos disponibles.
* Duplica la velocidad de acceso.
* No tiene redundancia. Si se rompe un disco, se pierde el archivo.
* RAID 1: todo se almacena en todos los discos (duplicado).
* Redundancia. Mejora el desempeño de lecturas.
* Pérdida de la mitad del almacenamiento físico. Pequeña penalidad en la escritura.
* RAID 5: Como mínimo tener tres discos. Guarda un componente en uno, el otro componente en otro y en el tercero guarda la paridad.
* Mejora en 2/3 la velocidad de acceso. Tiene redundancia.
* Mínimo 3 discos para armar el RAID. Pierdo 1/3 de espacio de almacenamiento.
* RAID 10: Mínimo 4 discos. Combinación entre 0 y 1. Divide los discos a la mitad: arriba usa RAID 0 y abajo 1
* Duplica la velocidad de acceso. Tiene redundancia.
* Mínimo 4 discos para armar el RAID. Se pierde el 50% del espacio de almacenamiento.

**SISTEMA DE ADMINISTRACIÓN DE BD - DMBS**

El DBMS es el software que maneja todo acceso a la BD. La finalidad general del DBMS consiste en proporcionar una interfaz de usuario para el sistema de BD. Cuando un usuario emite una petición de acceso, el DBMS intercepta esa petición y la analiza, inspecciona el esquema completo de la BD y ejecuta las operaciones necesarias sobre ella.

**Funciones del DBMS**:

* Definición de datos: debe ser capaz de aceptar definiciones de datos en la forma fuente y convertirlas a la forma objeto correspondiente. Debe incluir un procesador DDL.
* Manipulación de datos: debe ser capaz de manejar peticiones sobre la BD. Debe incluir un procesador DML.
* Optimización y ejecución: las peticiones DML deben ser procesadas por el componente *optimizador* de la BD
  + Optimización

Es el proceso del que se encarga el optimizador, el cual selecciona la forma más eficiente de implementar las peticiones del usuario. Esto permite que el sistema realice una navegación automática sobre los datos para satisfacer las peticiones.

Dada una petición en particular, el optimizador elegirá la estrategia para implementarla basándose en las tablas que usa, su tamaño, la cantidad de índices, cómo están agrupados físicamente los datos y qué operadores relacionales están involucrados.

* Seguridad e integridad de los datos: debe vigilar las peticiones del usuario y rechazar todo intento de violar las restricciones de seguridad y de integridad definidas por el DBA.
* Recuperación de datos y concurrencia: debe imponer ciertos controles de recuperación y concurrencia.
* Diccionario de datos: debe proporcionar una función de diccionario de datos.
  + Catálogo o diccionario

Contiene información detallada (*metadatos*) sobre los diversos esquemas de la BD con su respectivos mappings, entre otras cosas. El propio catálogo consiste en tablas (tablas del sistema).

* Rendimiento: debe realizar todas las tareas de la manera más eficiente posible.

**Comparación entre los sistemas de administración de BD y los sistemas de administración de archivos**

El administrador de archivos es el componente del SO subyacente que administra los archivos almacenados; está "más cerca del disco" de lo que lo está el DBMS. El DBMS es construido generalmente sobre algún tipo de administrador de archivos. El usuario de un sistema de administración de archivos no tiene restricciones de acceso a los datos y se encarga de su mantenimiento.

Los administradores de archivos no están al tanto de la estructura interna de los registros. Ofrecen poco soporte para las restricciones de seguridad y de integridad y para los controles de recuperación y concurrencia. No hay un concepto real de diccionario de datos. Proporcionan mucho menos independencia de datos que el DBMS. Normalmente los archivos son exclusivos de cierto usuario o aplicación en particular.

**Variables de relación base y vistas**

Las tablas existentes se denominan *variables de relación base* y a sus valores de relación se les llama *relaciones base*; a una relación que puede ser obtenida a partir de dichas relaciones base se la denomina *relación derivada*. Las *vistas* son una *relación derivada*.

Las variables de relación base representan datos que en realidad están almacenados en la BD. Las vistas sólo proporcionan diferentes formas de ver a los datos reales en un instante dado.

Las vistas permiten centrar, simplificar y personalizar la percepción de la BD para cada usuario. Son mecanismos de seguridad ya que los usuarios no acceden directamente a las tablas base.

**TRANSACCIÓN**

Unidad de trabajo lógica que comprende por lo regular a varias sentencias SQL. Es una unidad de trabajo porque todas las operaciones dentro de la transacción se toman como un todo y también es una unidad de recuperación porque si algunas de las operaciones falla, se vuelve toda la transacción atrás. Las transacciones empiezan con un BEGIN TRANSACTION y pueden finalizar con un COMMIT si la transacción terminó correctamente o un ROLLBACK si la transacción debe volverse atrás o deshacerse. Una sola sentencia se considera transacción porque puede que tenga que actualizar varios registros y por si hay algún fallo de energía.

Propiedades:

* *Atomicidad*: se ejecuta todo o nada.
* *Consistencia*: dejan siempre a las BD en un estado consistente antes y después de la transacción.
* *Aislamiento*: una transacción no interfiere en el procesamiento de otra y ambas mantienen la consistencia de la BD. (concurrencia)
* *Durabilidad*: Una vez que una transacción es confirmada, las modificaciones que realizó en la BD son permanentes más allá de que ocurra una caída posterior.

Recuperación de transacciones: Cuando una transacción es confirmada, se establece un COMMIT POINT o punto de confirmación, y corresponde al final de una unidad lógica de trabajo y a un estado de consistencia de la BD. En cambio, cuando una transacción es revertida (ROLLBACK) todas las modificaciones deben volverse al estado en que se encontraban en el COMMIT POINT anterior.

Log de transacciones (T-Log): El DBMS, para cada BD, mantiene en disco un T-Log en donde va almacenando secuencialmente todas las operaciones que se realizan y la información necesaria para revertirlas. Se utiliza el protocolo de escritura adelantada que dice que antes de realizar cualquier modificación en la BD se debe persistir la operación en el T-Log.

* Protocolo de escritura adelantada: Escribe primero en T-Log que está en disco, después los pasa a RAM donde están las tablas y después por detrás se actualizan las tablas en disco (check point). Esto garantiza la recuperación transacciones (recovery).

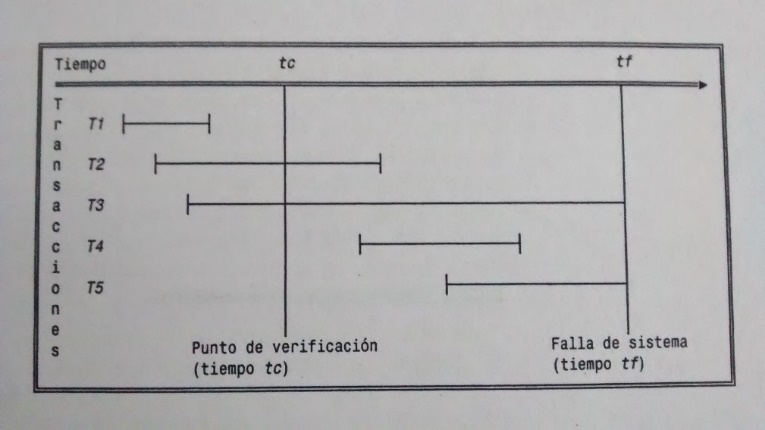
El Log tiene una parte activa (contiene las operaciones de las transacciones que aún no finalizaron) y otra pasiva (contiene las operaciones de transacciones que ya finalizaron).

CHECK POINT: Son momentos regulares en el tiempo en que el DBMS graba en disco todas las modificaciones existentes en el T-Log y registra en él el momento en que se realizó ese check point.

**RECOVERY**: Recuperación de transacciones.

* **Falla Local**: Se ejecuta cuando una transacción en particular hizo rollback e implica deshacer los cambios únicamente de esa transacción. Para eso debe retroceder en el T-Log a partir de la ejecución del rollback e ir revirtiendo tas las ejecuciones de esa transacción hasta el comienzo de la misma. Deshace todos los cambios en memoria.
* **Falla Global**: Es la que afecta al DBMS completo en donde hay que deshacer los comandos de las transacciones que aún no finalizaron y rehacer las transacciones que finalizaron correctamente pero que sus modificaciones no fueron persistidas en disco.
* *Soft Crash* (caídas suaves): Son caídas que no comprometen los medios de almacenamiento secundario y únicamente se pierde el contenido de la RAM.

Ejemplo:



DBMS al iniciar, ejecuta un proceso de recovery general (global) para dejar la BD en un estado consistente. Durante el recovery, los datos no son accesibles. Luego se posiciona (en el T-Log) en el momento del último Check Point ejecutado y procede de la siguiente manera:

1. Comienza con dos listas de transacciones, la lista DESHACER y la lista REHACER. Iguala la lista DESHACER con la lista de todas las transacciones dadas en el registro de punto de verificación más reciente y deja vacía la lista REHACER.
2. Busca hacia delante en la bitácora comenzando en el registro del punto de verificación.
3. Si encuentra una entrada BEGIN TRANSACTION en la bitácora para la transacción T, añade T a la lista DESHACER.
4. Si encuentra una entrada COMMIT para la transacción T, mueve a T de la lista DESHACER a la lista REHACER.
5. Por último, elimina de la tabla DESHACER las transacciones que empezaron luego del CHEK POINT y no terminaron.

Ahora el sistema trabaja hacia atrás en el T-Log, deshaciendo las transacciones que están en la lista DESHACER, y luego vuelve a trabajar hacia delante volviendo a hacer las transacciones que están en la lista REHACER teniendo en cuenta la necesidad de ejecutar Rollbacks parciales (Se deshace a partir del Check Point hacia atrás y se rehace toda la transacción).

Una vez terminada toda la actividad de recuperación, es entonces (y sólo entonces) que el sistema está listo para aceptar más trabajo.

* *Hard Crash* (caídas fuertes): Son caídas en donde hay compromiso en los medios secundarios. Ej: Se rompe un disco.

Recuperarse de esto implica restaurar las copias de seguridad que se han realizado. Una vez restaurada, si el medio de almacenamiento en donde estaba el T-Log no fue comprometido, se lo utiliza para ejecutar el proceso de recovery normal con la salvedad de que se inicia en el último check point anterior de la copia de seguridad. De esta forma no hay pérdida de datos.

En cambio, si el medio del almacenamiento del Log sí se vio comprometido, los datos cargados en horarios sin copia de seguridad se perdieron.

**Confirmación (COMMIT) en dos fases**

Es importante siempre que una transacción dada pueda interactuar con varios “administradores de recursos” independientes, donde cada uno administra su propio conjunto de recursos recuperables y mantiene su propio T-Log.

No tiene sentido que la transacción emita, un COMMIT para la BD1 y un ROLLBACK para la BD2, o viceversa; o que emita la misma instrucción para ambos porque el sistema aún podría fallar entre una y otra. En vez de ello, la transacción debe emitir un solo COMMIT (o ROLLBACK) a *nivel sistema*.

Ese COMMIT o ROLLBACK global es manejado por un componente del sistema llamado *coordinador*, cuya tarea es garantizar que ambos administradores de recursos confirmen o deshagan al unísono sus actualizaciones mediante el *protocolo de confirmación de dos fases*:

* El coordinador da instrucciones a todos los administradores de recursos para que estén listos para manejar la transacción. Luego, cada administrador de recursos debe forzar todos los registros de T-Log de los recursos locales usados por la transacción, hacia su propio T-Log físico; esto con el fin de que el administrador de recursos tenga ahora un registro permanente del trabajo que hizo a nombre de la transacción. Si la escritura forzada es satisfactoria, el administrador responde un "OK" al coordinador, en caso contrario responde "No OK".
* Cuando el coordinador ha recibido las respuestas de todos los participantes, fuerza una entrada en su propia bitácora física registrando su decisión con respecto a la transacción. Si todas las respuestas fueron "OK", esa decisión es "confirmar", en caso contrario la decisión es "deshacer". El coordinador informa después su decisión a cada participante y luego cada participante debe confirmar o deshacer la transacción localmente según se le indica.

**Concurrencia**

Los DBMSs permiten que muchas transacciones accedan a una misma BD a la vez. Se necesita algún tipo de *mecanismo de control de concurrencia* para asegurar que las transacciones concurrentes no interfieran entre sí.

* El problema de la actualización perdida

La actualización de una transacción A se pierde ya que una transacción B la sobrescribe.

* El problema de la dependencia no confirmada

Una transacción A recupera o actualiza una tupla que ha sido actualizada por otra transacción pero que aún no ha sido confirmada por esa misma transacción. A está operando sobre una suposición falsa y dicho cambio se pierde.

* El problema del análisis inconsistente

**Bloqueo**

Es una *técnica de control de concurrencia*. Cuando una transacción debe asegurarse de que algún objeto en el que está interesada no cambiará mientras lo esté usando, adquiere un bloqueo sobre ese objeto. El efecto del bloqueo es inhibir todas las demás transacciones en ese objeto y, por lo tanto, impedir que lo cambien.

El sistema soporta dos tipos de bloqueos, *bloqueo de escritura* (se rechaza una petición de cualquier tipo) y *bloqueo de lectura* (Se rechaza una petición de escritura y se permite una de lectura).

Bloqueo por aproximación: *Protocolo de bloqueos por aproximación*: ninguna transacción puede adquirir un bloqueo sobre una tupla sin antes poseer un bloqueo por aproximación sobre la tabla que lo contiene. Tres tipos de bloqueos: aproximación de bloqueo compartido, aproximación de bloqueo exclusivo y aproximación de bloqueo compartido-exclusivo.

Bloqueo Mortal: Cuando dos o más transacciones se encuentran en estados simultáneos de espera, cada una de ellas esperando que alguna de las demás libere un bloqueo para poder continuar. Si ocurre, es preferible que el sistema lo detecte (detección de un ciclo en el grafo de espera) y lo rompa (seleccionar una de las transacciones bloqueadas mortalmente y deshacerla). Algunos sistemas volverán a iniciar esta transacción desde el principio.

**Seriabilidad**: Un conjunto de transacciones es seriable si el resultado de su ejecución concurrente es igual al resultado de su ejecución en serie. Se cumple mediante el protocolo de bloqueo en dos fases, el cual consiste en que una transacción debe adquirir un bloqueo sobre el objeto antes de utilizarlo y después debe liberar ese objeto.

Niveles de aislamiento: Es el *grado de interferencia* que una transacción dada es capaz de tolerar por parte de las transacciones concurrentes. Si se quiere garantizar la seriabilidad, no se puede aceptar ninguna cantidad de interferencia. Entre mayor sea el nivel de aislamiento, menor será la interferencia y la concurrencia.

**INTEGRIDAD**

Es la exactitud o corrección de los datos en la BD. Una BD podría estar sujeta a cualquier cantidad de restricciones de integridad.

Restricciones de Tipo: Es una sola enumeración de los valores válidos del tipo.

Restricciones de Atributo: Es sólo una declaración para que un atributo especificado sea de un tipo en particular.

Restricciones de Tablas: Impuesta a una tabla individual.

Restricciones de BD: Relaciona dos o más tablas distintas dentro de la misma BD.

Restricciones de Estado: Se ocupan de los estados correctos de la BD.

Restricciones de transición: Determinan que transiciones de un estado a otro son válidas.

**CLAVES**

Claves candidatas: Deben cumplir con la propiedad de unicidad. Toda tabla tiene por lo menos una clave candidata, la cual es la única forma garantizada por el sistema para señalar una tupla específica. Si una tabla tiene más de una clave candidata, una será *clave primaria* y las otras serán *claves alternas****.***

Claves externas: Representa una referenciaa la tupla que contiene el valor de clave candidata coincidente. Una clave externa es un conjunto de atributos de una tabla 2 cuyos valores tienen que coincidir con los valores de cierta clave candidata para una tabla 1. Cada atributo de una clave externa dada debe tener el mismo nombre y tipo que los componentes correspondientes de la clave candidata con la que coincide.

**Procedimientos almacenados**

Es un grupo de una o varias transacciones, los cuales pueden aceptar parámetros de entrada y devolver parámetros de salida, contener instrucciones que realicen operaciones en la BD y llamadas a otros procedimientos y devolver un valor de estado a un programa.

Ventajas: Tráfico de red reducido entre el cliente y el servidor, mayor seguridad (controla procesos y actividades), reutilización del código, mantenimiento más sencillo, rendimiento mejorado (se compila la primera vez que se ejecuta y crea un plan de ejecución que vuelve a usarse).

**Procedimientos activados (triggers)**: Son procedimientos definidos por el usuario. Es invocado automáticamente al ocurrir algún evento o condición de activación.

**Check**: Es una restricción para especificar los valores de datos aceptables en una o más columnas.