BLOQUE 4: Sistemas de Bases e Datos Relacionales
TEMA 10: Estructura de un Sistema de Bases de Datos Relacional
10.1 Componentes del entorno de la base de datos

- **Hardware:** Desde un simple PC ... a una red de ordenadores (frontend) con servidor central (backend)
- **Software:** SGBD + Aplicaciones + Sistema Operativo + Software de Red
- **Datos:** Datos + Metadatos
- **Procedimientos:** Instrucciones y reglas que gobiernan el diseño y uso de la base de datos y aplicaciones.
- **Personas (actores):**
  - Administrador de la base de datos (ABD). Persona (o grupo) responsable de administrar los recursos del SBD (a nivel técnico): BD + SGBD + Software de aplicaciones o programas de acceso
  - Diseñadores de bases de datos :
    - Recogen y comprenden las necesidades y objetivos de los futuros usuarios.
    - Construyen **Vista** (subconjunto de la información) que satisface requisitos de cada grupo de usuarios
    - Diseño final de BD que satisface necesidades de todos los usuarios
  - Desarrolladores de Software
    - Analistas: Determinan necesidades de procesamiento de los usuarios finales Especifican conjuntos de operaciones que satisfacen esas necesidades
    - Desarrolladores de aplicaciones : implementan dichas especificaciones, creando programas de aplicación
  - Usuarios finales: son los clientes
    - inexpertos: Accesos muy frecuentes y repetitivos. No son conscientes de la existencia del SGBD. Acceden a la BD mediante programas de aplicación que facilitan sus operaciones al máximo.
    - avanzados: familiarizados con la estructura de la base de datos y las funcionalidades ofrecidas por el SGBD. Pueden usar SQL para realizar sus operaciones.

10.2 Arquitectura de tres niveles ANSI-SPARC
Uno de los principales objetivos de un SBD es proporcionar a los usuarios una visión abstracta de los datos: ocultar detalles de cómo se almacenan y manipulan los datos, pero para la recuperacion de estos es necesario usar 1) estructuras de datos complejas para representar/almacenar esos datos en la BD y 2) mecanismos complejos para su manipulación. Distintos niveles de abstraccion.
Arquitectura ANSI/X3/SPARC

ANSI: American National Standards Institute
X3: Database subcommittee
SPARC: Standards Planning and Requirements Committee

Nivel Externo: Describe qué parte de los datos es relevante para cada (tipo de) usuario o aplicación Para ello usa múltiples Esquemas Externos. Cada Esquema Externo (Vista) define la porción de la BD que interesa a cada (tipo de) usuario o aplicación
Nivel Conceptual/Logico: Describe qué datos están almacenados en la BD y las relaciones entre ellos. Para ello usa el Esquema Conceptual o Lógico, que define la estructura lógica de la BD completa, tal y como la ve el ABD. Todo dato disponible en cada Esquema Externo debe estar contenido en, o ser derivado de, el nivel Conceptual/Lógico. No contiene ningún detalle relativo al almacenamiento
Actualmente se distinguen 2 esquemas en este nivel:

- **Esquema Conceptual:** organizativo, más cercano al usuario
- **Esquema Lógico:** más cercano al SGBD

Nivel interno: Describe cómo los datos están almacenados en la BD ⊙Para ello usa el Esquema Interno, el cual define la implementación física de la BD completa, con el fin de optimizar el rendimiento y el uso del espacio ■ Asignación de espacio de almacenamiento para datos e índices ■ Descripción de los tipos de registros almacenados, ■ Colocación de los registros (ordenados, no ordenados...) ■ Estructuras de almacenamiento (hashing,...), ■ Estructuras de acceso (indexación,...) ■ etc. ⊙Muy cercano al nivel físico, pero no trata con registros físicos (bloques, páginas, …) ni con unidades como cilindros o pistas.

Los 3 niveles de la A3N son **descripciones de datos**. Los datos reales sólo están en el **nivel físico** (en el disco) .
El SGBD es el responsable de mantener la correspondencia entre esquemas a través de los niveles y debe comprobar la consistencia entre ellos:
⊙Confirmar que cada Esquema Externo es derivable del Esquema Conceptual/Lógico (EC/L)
⊙Usar la información en el Esquema Conceptual/Lógico para establecer una correspondencia entre cada Esquema Externo y el Esquema Interno

10.3 Independencia de Datos
10.4 Funciones del SGBD

1. Almacenamiento, recuperación y actualización de datos

Proporcionar al usuario la capacidad de almacenar, recuperar y actualizar datos en la base de datos y de forma eficiente

2. Servicios para permitir la independencia de datos

Facilidades para soportar la independencia de los programas de la estructura física de la base de datos.

3. Metadatos accesibles para el usuario. **INFORMATION_SCHEMA**

Metadatos almacenados en la base de datos y accesibles para el usuario y para el SGBD, imprescindible para hacer posible la Arquitectura de Tres Niveles y para el funcionamiento de los subsistemas software que integran el SGBD.

4. Servicios de integridad

Mecanismos para asegurar que tanto los datos en la base de datos, como los cambios sobre los datos, siguen ciertas reglas. **Reglas de integridad.**

5. Soporte y procesamiento de transacciones

Una **Transacción** es una secuencia de acciones que lee y/o actualiza el contenido de la BD, y que debe ser ejecutada como una **unidad**. **Soporte de Transacciones:** mecanismo que asegura que todos los cambios hechos por una transacción se realizan correctamente y quedan almacenados en la BD, o bien que ninguno de esos cambios se ha realizado.

6. Servicios de control de la concurrencia

Mecanismo que garantiza que la BD es actualizada correctamente cuando múltiples usuarios/aplicaciones modifican los datos simultáneamente.

7. Servicios de recuperación de fallos

Mecanismo para recuperar la BD tras la ocurrencia de fallos en el sistema que quizá han dejado los datos inconsistentes: fallos del sistema (system crash), fallos en los medios de almacenamiento, errores hardware o software que han parado el SGBD, fallos provocados por el usuario (cancelación de ejecución), etc.
La recuperación consigue dejar la BD en un estado en el que los datos son consistentes

8. Servicios de autorización (seguridad)

Garantiza que sólo los usuarios autorizados pueden acceder a la base de datos. **Control de acceso selectivo:**

- Solo usuarios autorizados
- Solo a ciertas partes de la base de datos
- Solo para realizar ciertas operaciones con los datos

9. Soporte para la comunicación de datos

Un SGBD debe poder integrarse con software de comunicaciones (Data Communication Programs)
Garantiza que los usuarios accedan sin problemas a una BD centralizada desde ubicaciones remotas

10. Utilidades adicionales

El SGBD proporciona un conjunto de utilidades que ayudan al ABD a administrar la BD de manera efectiva:

- Herramientas de importación y exportación de datos
  - Carga de datos en la base de datos a partir de ficheros sin estructura
  - Descargar datos en ficheros de diversos tipos
  - Intercambio de información entre diferentes BD
- Utilidades de monitorización o supervisión, para controlar el uso, operación y rendimiento del sistema
- Programas de análisis estadístico para examinar las estadísticas de rendimiento y uso
- Facilidades de reorganización de ficheros o de índices
- Herramientas de backup
- Utilidades de ordenamiento, compactación y/o compresión de ficheros, etc.

10.5 ANEXOS (No entra en el examen)
10.5.1 Estructura general del SGBD
10.5.2 Ventajas y desventajas de los SGBD
10.5.3 Sistemas tradicionales de Procedimientos de Ficheros

TEMA 11: Aspectos basicos del Procesamiento de Transacciones
11.1 Soporte de transacciones

Una **transacción** es una acción, o una secuencia de acciones, llevada a cabo por un usuario o programa de aplicación, y que lee y/o actualiza el contenido de la base de datos.
Es una **UNIDAD LOGICA DE PROCESAMIENTO.** Puede ser un programa completo, una parte de un programa, una serie de sentencias SQL, o una unica sentencia.
Una transaccion es **atomica** o se ejecutan todas las operaciones que componen la transaccion o no se realiza ninguna. Toda transaccion acaba con éxito (**COMMIT**) o con fracaso (**ROLLBACK**).
Propiedades **ACID** de una transaccion:

- A → **atomicidad**: Subsistema de **Recuperacion** del SGBD

Ejecucion todo o nada

- C → **consistencia**: **Programadores** + Subsistema de **Integridad** del SGBD

Una transaccion T lleva la BD de un estado de consistencia a otro estado de consistencia. Al inicio de una transacción y justo al terminar, la BD está en un estado de consistencia: se cumplen todas las restricciones de integridad. Durante la ejecución de una transacción es posible que alguna restricción no se cumpla.

- I → **isolation** (aislamiento): Subsistema de **Concurrencia** del SGBD

Las transacciones son procesos o threads concurrentes: puede haber conflictos entre ellos si acceden a los mismos datos. La ejecucion de T es independiente de la del resto de transacciones. Los

cambios producidos por T durante su ejecucion no deberian ser visibles para otras transacciones hasta que T finalice.

- D → **durabilidad**: Subsistema de **Recuperacion** del SGBD

Una vez que T finaliza con éxito y es confirmada, sus cambios están almacenados en la BD de forma permanente y perduran, aunque el sistema falle justo después.

Inicion de transaccion en Oracle:
Cuando no hay ya una transacción en progreso, y se ejecuta una sentencia LMD. Sentencias LMD: INSERT, UPDATE, DELETE, SELECT.
Cuando se ejecuta una sentencia LDD.Sentencias LDD: CREATE, ALTER, DROP, RENAME, COMMENT, si ya existe una transacción en ejecución, Oracle la finaliza con COMMIT y comienza una nueva para esta sentencia LDD.
Finalizacion de transaccion en Oracle:

- **COMMIT**: Finaliza la transacción actual y confirma (hace permanentes) los cambios realizados
  - COMMIT explícito
  - COMMIT implícito (por parte del SGBD) cuando.el programa finaliza de forma normal y sin errores, se sale de la herramienta (SQLDeveloper, ...) correctamente o se ejecuta una sentencia LDD **Cada sentencia LDD es tratada como una transacción en sí misma.**
- **ROLLBACK**: Finaliza la transacción actual y deshace (anula) los cambios realizados
  - ROLLBACK explícito (por parte del programa o usuario).
  - ROLLBACK implícito (por parte del SGBD) cuando el programa finaliza de forma anormal, inesperada o errónea o se sale de la herramienta (SQLDeveloper, ...) apagando el PC, o cerrando la ventana directamente, sin desconectar de la BD.

Diapositiva 19 examen

11.2 Concurrencia de transacciones
Los sistemas de procesamiento de transacciones son sistemas con grandes bases de datos y cientos de usuarios concurrentes que ejecutan transacciones. Requieren alta disponibilidad (24/7/365), y respuesta rápida para sus cientos de usuarios. Las transacciones emitidas por diferentes usuarios suelen ejecutarse concurrentemente, y pueden acceder a y actualizar los mismos elementos de BD.
Control de la concurrencia: garantiza la actualizacion correcta de la base de datos cuando hay multiples usuarios modificando de manera concurrente la base de datos. Tecnicas de control de la concurrencia:
El **bloqueo (locking)** uso de candados/cerrojos/bloqueos para controlar el acceso concurrente o simultáneo a los datos de la BD. Tipos de bloqueos:

- Bloqueo Compartido (lectura)
  - Si T tiene un bloqueo compartido sobre un elemento de datos, T puede leer el elemento
  - Otras transacciones pueden leer el mismo elemento al mismo tiempo
  - Ni la transacción que lee, ni ninguna otra, pueden escribir el elemento
- Bloqueo exclusivo (escritura)
  - Si T tiene un bloqueo exclusivo sobre un elemento de datos, sólo dicha transacción puede leer y escribir el elemento
  - El resto de las transacciones no pueden leer ni escribir ese elemento.

Reglas de uso:
Si una transacción necesita leer un elemento X, antes debe obtener un bloqueo compartido sobre X (si no lo consigue, espera)
Si una transacción desea escribir un elemento Y, antes debe conseguir un bloqueo exclusivo sobre Y (si no lo consigue, espera)

Una vez realizada la operación, la transacción debe desbloquear el elemento de datos
Una operación de COMMIT o ROLLBACK libera todos los bloqueos adquiridos por la transacción

Los SGBD implementan los **niveles de aislamiento utilizando bloqueos.** (Sentencia del ANSI SQL que permite definir el nivel de aislamiento para una transacción: SET TRANSACTION ISOLATION LEVEL nivel;)
Debe ser la 1ª sentencia de la transacción. Valores posibles para "nivel":
⊙ **READ UNCOMMITED** Es posible que se lean datos modificados por transacciones no confirmadas. Los bloqueos de lectura y de escritura son liberados en cuanto finaliza la sentencia
⊙ **READ COMMITTED** Siempre se leen datos modificados por transacciones confirmadas .Cada sentencia dentro de la transacción sólo ve los datos confirmados antes del inicio de la sentencia (no de la transacción). Se consigue haciendo que los bloqueos de lectura sean liberados en cuanto finaliza la SELECT y los bloqueos de escritura no sean liberados hasta el final de la transacción
⊙ **REPEATABLE READ** Se realizan siempre lecturas 'repetibles': dentro de una transacción siempre se obtienen los mismos valores en diferentes lecturas sucesivas de los mismos datos. Se consigue haciendo que los bloqueos de lectura y escritura no sean liberados hasta el final de la transacción
⊙ **SERIALIZABLE → Aislamiento absoluto**. Cada sentencia dentro de la transacción sólo ve los datos confirmados antes del inicio de la transacción y sus propios cambios Se consigue haciendo que los bloqueos de lectura y escritura no sean liberados hasta el final de la transacción y, además, realizando 'bloqueos de rango': se bloquean los datos seleccionados con cada SELECT...FROM...WHERE

Nivel de Aislamiento	Lectura sucia	Lectura no repetible	Lectura Fantasma
READ UNCOMMITED	Posible	Posible	Posible
READ COMMITED	No	Posible	Posible
REPEATABLE READ	No	No	Posible
SERIALIZABLE	No	No	No

☐ **Lectura sucia (Dirty Read)**
⊙ Una transacción T1 actualiza ciertos datos, después otra transacción T2 lee dichos datos (ya modificados)
⊙ T1 ejecuta ROLLBACK y los datos vuelven al estado original
⊙ En este momento, los datos leídos por T2 no son válidos: son fruto de una lectura sucia
☐ **Lectura no repetible (Non-repeatable Read)**
⊙ Después de que una transacción T1 lea ciertos datos de la BD, otra transacción T2 los actualiza y se confirma
⊙ Cuando T1 lee de nuevo los mismos datos encontrará que han cambiado: es una lectura no repetible, los mismos datos leídos 2 veces son diferentes
☐ **Lectura fantasma (Phantom Read)**
⊙ Una transacción T1 ejecuta una consulta, otras transacciones insertan nuevas filas y se confirman
⊙ La transacción T1 ejecuta de nuevo la misma consulta y el resultado muestra las nuevas filas (fantasmas)

11.3 Recuperacion tras fallos

El SGBD ofrece un mecanismo para recuperar la base de datos en caso de que resulte afectada de alguna forma por un fallo que afecte a su procesamiento. La ocurrencia de fallos puede provocar la pérdida de datos en los medios de almacenamiento. **El SGBD puede devolver la BD a un estado coherente**

- **Fallos locales:** Sólo afecta a la transacción T fallida. Pérdida de datos correspondientes a T en memoria principal
  - **fallo local previsto:** Cancelación 'programada', excepción tratada
  - **fallo local no previsto:** Error de programación (bug), división por cero, desbordamiento, interrupción provocada por el usuario
  - **fallo por imposicion del control de la concurrencia:** El método de control de concurrencia decide abortar T porque incumple la serialización, o bien para romper un interbloqueo
- **Fallos globales:** Afecta a todas las transacciones en ejecución. Pérdida de datos en memoria principal, y quizá en el almacenamiento secundario
  - **fallo del sistema:** Mal funcionamiento hardware, software (SGBD, SO), o red. No daña el disco: el contenido de la BD queda intacto
  - **fallo de los medios de almacenamiento:** 'Aterrizaje' de cabezas lectoras de disco, o soportes no legibles, etc. Algunos bloques del disco pueden perder sus datos: BD corrupta
  - **fallos fisicos y catastrofes:** Desastre natural: inundación, terremoto, incendio, apagón... Robo, sabotaje, destrucción o corrupción de datos, de hardware, de software o de las instalaciones

Tras un fallo hay que considerar 2 efectos:

- Perdida de la memoria principal incluyendo la cache y otros buffers del SGBD
- Perdida de datos de la BD en el almacenamiento secundario

GESTOR DE RECUPERACION: garantiza la atomicidad y durabilidad incluso cuando ocurren fallos. Debe asegurarse que aun en presencia de fallos cada transaccion T ejecuta todas sus operaciones con éxito y su efecto que permanente en la BD o bien no tiene ningun efecto en la BD.

-Estructuras de memoria

- **Cache de la BD:**

⊙ Uno o más búferes en memoria principal que se utilizan para transferir (bloques de) datos desde y hacia el disco
⊙ Es un área común a todas las transacciones
  ■ Una vez que un bloque es copiado a la caché de BD, el bloque queda accesible a todas las transacciones en ejecución
  ■ Por tanto, si una T escribe un elemento X, el nuevo valor queda almacenado en la caché de BD en memoria y el resto de transacciones pueden verlo
  ■ De hecho, toda operación leer y escribir busca primero el bloque en la caché de BD en memoria, y si no está ahí, entonces lo busca en el disco
⊙ Sólo cuando se vuelcan los bloques de datos modificados desde la caché de BD a disco, se puede considerar permanente esa modificación

- **Areas de trabajo privadas**

⊙ Cada transacción T tiene asignada un área en memoria principal donde guarda todo elemento de datos que lee y/o escribe
⊙ Es un espacio en memoria principal y local a la transacción T
⊙ Se crea al iniciarse T y se elimina cuando T finaliza

-Lectura de la BD
-Escritura de la BD

RECUPERACION DE FALLOS: REDUNDANCIA + ACCIONES

- REDUNDANCIA
  - Facilidades de **backup**, para realizar copias periódicas de la base de datos
  - Mecanismo de **bitácora**, para mantener la pista del estado actual de las transacciones y los cambios de los datos

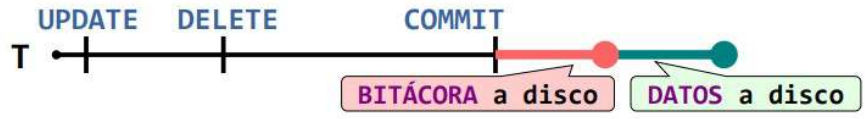
Fichero especial que almacena detalles sobre las operaciones efectuadas por las transacciones. Se mantiene en el almacenamiento secundario.
Y en la memoria principal, obviamente, se mantiene el búfer de bitácora (~caché). Se va rellenando con entradas hasta que se llena, momento en el que se vuelca a disco. Entradas fichero bitacora:

- **<INICIAR, T>**
  - Indica que la transacción T ha comenzado su ejecución
- **<LEER, T, x>**
  - Indica que T ha leído el valor del elemento x de la base de datos
- **<ESCRIBIR, T, x, valor_anterior, valor_nuevo>**
  - Indica que T ha modificado el valor del elemento x
- **<COMMIT, T>**
  - Indica que T ha finalizado con éxito y su efecto puede ser confirmado en la BD en disco: sus cambios pueden quedar permanentes
- **<ROLLBACK, T>**
  - Indica que la transacción T ha sido anulada de forma que ninguna de sus operaciones tendrá efecto sobre la BD: la transacción será revertida, todas sus operaciones serán deshechas
- **<PUNTO DE CONTROL, ...>** → La veremos más adelante

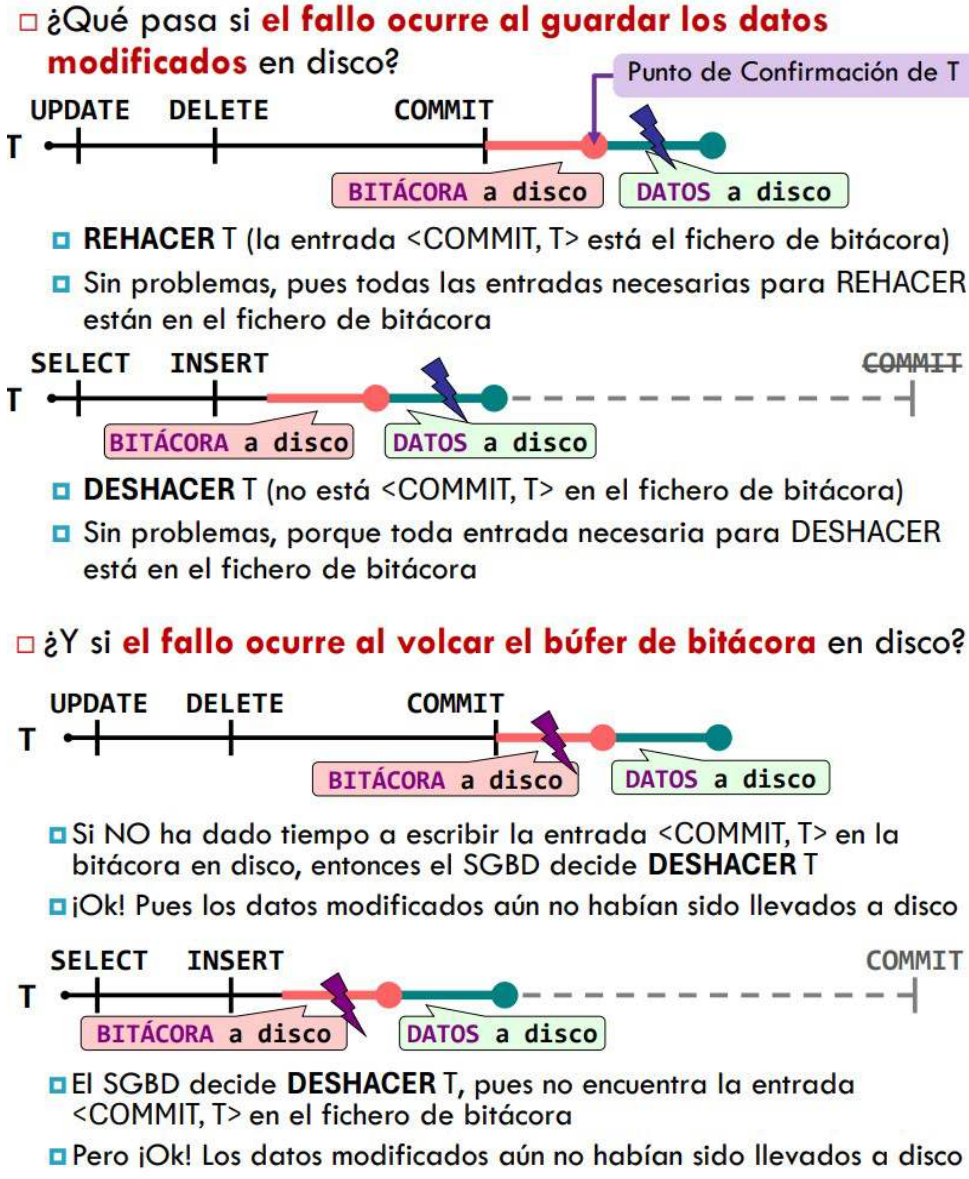
Si en la bitácora NO está la entrada es porque T estaba en curso de ejecución **Se debe DESHACER T, s**i en la bitácora SÍ está la entrada es que T había finalizado correctamente **Se debe REHACER T.** Hay que reacer T porque no es seguro que todos sus cambionse llevasen a disco.
-Bitacora adelantada: Si ocurre un fallo global cuando el búfer de bitácora está incompleto, su contenido se pierde al igual que el resto de la memoria principal, dichas entradas no serán consideradas en el proceso de recuperación, pues el SGBD acude al fichero bitácora para aplicar las acciones de recuperación. Esto puede impedir la restauración correcta tras el fallo de una transacción.
No se puede guardar en disco los datos modificados por T hasta que se haya escrito en disco toda entrada de bitácora para T hasta el momento actual,antes de guardar en disco datos modificados por T, es necesario escribir en el fichero de bitácora todas las entradas de bitácora para T hasta el momento.

- Si el SGBD decide **llevar a disco cambios** realizados por T **mientras que T** aún **está en ejecución...**

- Si el SGBD decide **llevar a disco cambios** realizados por T **cuando** T ya **ha ejecutado COMMIT...**



Por tanto, el COMMIT de T se completa una vez que se ha escrito en disco toda entrada de bitácora pendiente para T. En ese momento, T alcanza su '**punto de confirmación**' esto quiere decir que T ha pasado al estado CONFIRMADA. Está asegurado que sus modificaciones sobre los datos serán permanentes, sea cual sea el momento en el que el SGBD vuelque dichos cambios al disco e incluso si no da tiempo a guardarlos debido a un fallo.



- ACCIONES DE RECUPERACION
  - Un protocolo de **checkpoints**

En el proceso de recuperación no se revisa TODA la bitácora en disco: sería muy costoso y poco práctico. Para limitar la porción de la bitácora examinada y el coste de su procesamiento, se usan checkpoints

El SGBD marca automáticamente un punto de control, cada m segundos, o tras escribir n entradas < COMMIT, Ti > en la bitácora desde el último punto de control. Marcar un punto de control significa:

1. **Suspender** la **ejecución** de las **transacciones**
2. **Forzar** la **escritura en disco del búfer de bitácora**
   - Volcado de todas las entradas **en el fichero bitácora**
3. **Forzar** la **escritura en disco de todo bloque modificado de la caché de BD**
   - Volcado de todos los bloques modificados **a los ficheros de datos**
4. **Escribir** una **entrada** <PUNTO DE CONTROL> en el fichero de bitácora en disco
5. **Escribir** en un ***Fichero Especial de Arranque*** la dirección de la entrada < **PUNTO DE CONTROL** > dentro del fichero de bitácora
6. **Reanudar** la **ejecución de las transacciones**

☐ La entrada en el fichero de bitácora de tipo contiene:
⦿Lista de identificadores de las transacciones activas (en curso de ejecución) en ese instante
⦿Dirección en el fichero bitácora de la 1ª y última entradas para cada Ti activa
☐ Un punto de control sincroniza memoria y disco
⦿búfer de bitácora → fichero de bitácora
⦿caché de BD → ficheros de datos
En el proceso de recuperación el uso de puntos de control permite Recorrer la bitácora a partir del último punto de control e ignorar las Ti confirmadas antes del último punto de control

- Un Gestor de Recuperación, que permita al sistema restaurar la BD a un estado consistente tras un fallo, mediante **técnicas y estrategias de recuperación**.

- Recuperación tras un **fallo de tipo 5 o 6**, que produjo daños físicos en la BD...
  - **Restaurar** la última **copia de seguridad** de la BD, en un nuevo disco
  - Reconstruir un estado más actual usando la **bitácora** en disco: **Rehacer operaciones escribir** de las transacciones **confirmadas** hasta el momento de la caída
- Recuperación tras un **fallo de tipo 1 a 4**, que no ha dañó la BD físicamente, pero la dejó inconsistente...
  - **Deshacer operaciones escribir** de las **transacciones** que estaban **en curso** y no se confirmaron
  - **Rehacer**, si es necesario, **operaciones escribir** de las **transacciones confirmadas**, para asegurar que sus cambios están en disco
  - Para ello, usar la **bitácora** en disco y un algoritmo de recuperación

-Algoritmo Deshacer/Rehacer

1. Se accede a la ultima entrada <PUNTO DE CONTROL>
2. Esta entrada contiene la lista A de transacciones activas A = { T2, T1, T3 }
3. Se crea la lista C de confirmadas C = CJT VACION
4. Recorrer el fichero de bitácora desde ahí
5. Cada <INICIAR, T> añade T a A

6. Cada <COMMIT, T> mueve T de A a C
7. Al terminar de recorrer la bitácora A = {T1, T3} y C = {T2}

Recuperación: ❖ Deshacer las de A: T1 y T3 ❖ Rehacer las de C: T2 ❖ Ignorar el resto: T4

11.4 Procesamiento y optimizacion de consultas

El SGBD garantiza la obtención eficiente de datos

- Búferes y cachés de datos, para mantener en memoria y procesar datos extraídos de la BD en disco
- Estructuras de datos auxiliares y técnicas de búsqueda para acelerar la búsqueda en disco de los registros deseados
  - Índices (ficheros auxiliares)
  - Elegir qué índices crear y mantener es parte de la etapa de Diseño Físico y Ajuste del esquema de la BD
- Técnicas de optimización y estrategias de procesamiento para acelerar la ejecución de las consultas

El SGBD siempre ejecuta las sentencias SQL de la forma más eficiente posible