# Recuperación

## Administración de Bases de Datos

Departamento de Lenguajes y Sistemas Informáticos

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

# Contenido

1) Introducción
2) Transacciones
3) El diario de recuperación
4) Algoritmos de recuperación
5) Estructura de InnoDB

# Introducción

- El SGBD debe asegurar la disponibilidad de los datos.

- El SGBD proporciona mecanismos para recuperar la BD frente a fallos lógicos o físicos que puedan destruir datos total o parcialmente.

# Introducción

- Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
  - Fragmento del artículo 9, primer punto:

*"El responsable del fichero, y … el encargado del tratamiento, deberán adoptar las medidas ... que … eviten su alteración, pérdida, … habida cuenta de … los riesgos a que están expuestos, ya provengan de la acción humana o del medio físico o natural."*

  - Riesgos del medio físico: fallos físicos o lógicos en el servidor.

# Introducción

- Dos situaciones a considerar:

- Recuperación a nivel de tabla/BD.
  - Mecanismos: Copias de seguridad y log binario.
  - *Se trabajará en el laboratorio 4.*

- Recuperación a nivel de transacción.
  - Mecanismo: Log de recuperación.
  - *Se trabajará en las sesiones magistrales.*

# Introducción

- Ejemplo:
  - Tabla “banco” con los datos bancarios de varias personas.

| id | nombre | numCuenta | saldo |
|---|---|---|---|
| 1 | Unai | 0123 3210 11 0110220330 | 600 |
| 2 | Mikel | 0123 3210 54 6547891235 | 800 |
| 3 | ... | | ... |

  - Queremos reflejar una transferencia de 100€ de Mikel a Unai.
    - ¿Qué operaciones son necesarias?

# Introducción

- Ejemplo:
  - Transferencia de 100€ de Mikel a Unai:

```
UPDATE banco SET saldo = saldo - 100 WHERE id = 2;
UPDATE banco SET saldo = saldo + 100 WHERE id = 1;
```

  - Problema: ambas operaciones deben realizarse correctamente.

  - ¿Qué pasa si hay un fallo entre la 1ª y 2ª operación?
    - ¿Cuál sería el estado de la BD entre la 1ª y 2ª sentencia?

# Introducción

- Ejemplo:
  - Una transacción es un mecanismo que asegura la ejecución de todas las operaciones que contiene.
  - Solución : crear una transacción con ambas operaciones:

```
< INICIO DE TRANSACCIÓN >
UPDATE banco SET saldo = saldo - 100 WHERE id = 2;
UPDATE banco SET saldo = saldo + 100 WHERE id = 1;
< FIN DE TRANSACCIÓN >
```

  - Si sucede un fallo entre operaciones, la transacción se aborta y no hay cambios en la BD.

# Transacciones

- Una transacción es un conjunto de operaciones que llevan la BD de un estado consistente a otro.

- Una transacción se considera:
  - Unidad lógica de procesamiento: Secuencia de instrucciones que incluye operaciones de acceso (inserción, borrado, modificación o consulta) a la base de datos.
  - Unidad lógica de integridad: Secuencia de operaciones que llevan la BD de un estado consistente a otro estado consistente.
  - Unidad lógica de recuperación.

# Transacciones

- Inicio de transacción.
  - Operación:

```
START TRANSACTION
```

- Fin de transacción.
  - Éxito: las modificaciones realizadas en la transacción se deben escribir en la BD.
    - Operación:

```
COMMIT
```

  - Fracaso: las modificaciones realizadas en la transacción se descartan (no se escriben en la BD).
    - Generalmente debido a un error en las operaciones de la transacción.
    - Operación:

```
ROLLBACK
```

# Transacciones en MySQL

- Iniciar una nueva transacción en MySQL:

```
mysql> start transaction;
```

  - Todos los comandos que se introduzcan después forman parte de la transacción.
  - Se finaliza con "commit" o "rollback".
    - Más información en [1].

- Ejemplo completo de transacción:

```
mysql> start transaction;
mysql> update cuenta set dinero = dinero - 100 where nombre = "Unai";
mysql> update cuenta set dinero = dinero + 100 where nombre = "Mikel";
mysql> commit;
```

[1] MySQL 8 Docs, START TRANSACTION, COMMIT, and ROLLBACK Statements: https://dev.mysql.com/doc/refman/8.0/en/commit.html

# Transacciones en MySQL

- Por defecto, MySQL funciona en modo AutoCommit[1].
  - Se introduce una sentencia "commit" automáticamente después de cada operación.
    - Crea una transacción por cada operación.
  - AutoCommit se puede desactivar para gestionar las transacciones de forma manual.

  - Para activar/desactivar AutoCommit, asignar 1/0 a la variable "autocommit".
    - Desde la consola, para desactivar en la sesión en curso:

```
mysql> SET autocommit=0;
```

[1]MySQL 8 Docs, START TRANSACTION, COMMIT, and ROLLBACK Statements: https://dev.mysql.com/doc/refman/8.0/en/commit.html

# Ejercicio 1

- Crear una BD “BDTransac”, que contenga una tabla “TablaTransac” con 2 columnas: id (int), texto (text)
  - Después, introducir 2 líneas con datos aleatorios en TablaTransac.

- Iniciar manualmente una transacción:
  1) Realizar la inserción de 1 nueva línea en “TablaTransac”.
  2) Sin finalizar la transacción, provocar un fallo en la conexión a MySQL.
     - P.e. cerrar la terminal con la sesión activa.

- Abrir una nueva conexión y verificar qué datos hay en “TablaTransac”.

# Transacciones

- El SGBD se encarga de la ejecución de las transacciones.
- Para garantizar la validez de los datos frente a fallos, los SGBD cumplen las propiedades ACID para sus transacciones.
  - Propiedades establecidas en los 80s para describir las necesidades de un sistema transaccional fiable[1].

[1] "¿Qué es el modelo ACID en bases de datos?", KeepCoding.io: https://keepcoding.io/blog/que-es-acid-bases-datos/

# Transacciones

- Propiedades ACID:

- ***A**tomicity*
  - La transacción como unidad lógica de procesamiento.
  - Una transacción NO puede ejecutarse a medias.
    - O se ejecuta íntegramente ,o no se ejecuta (todo o nada).

- ***C**onsistency*
  - La transacción como unidad lógica de integridad.
  - Una transacción hace que la BD pase de un estado consistente a otro (antes y después de la transacción).

# Transacciones

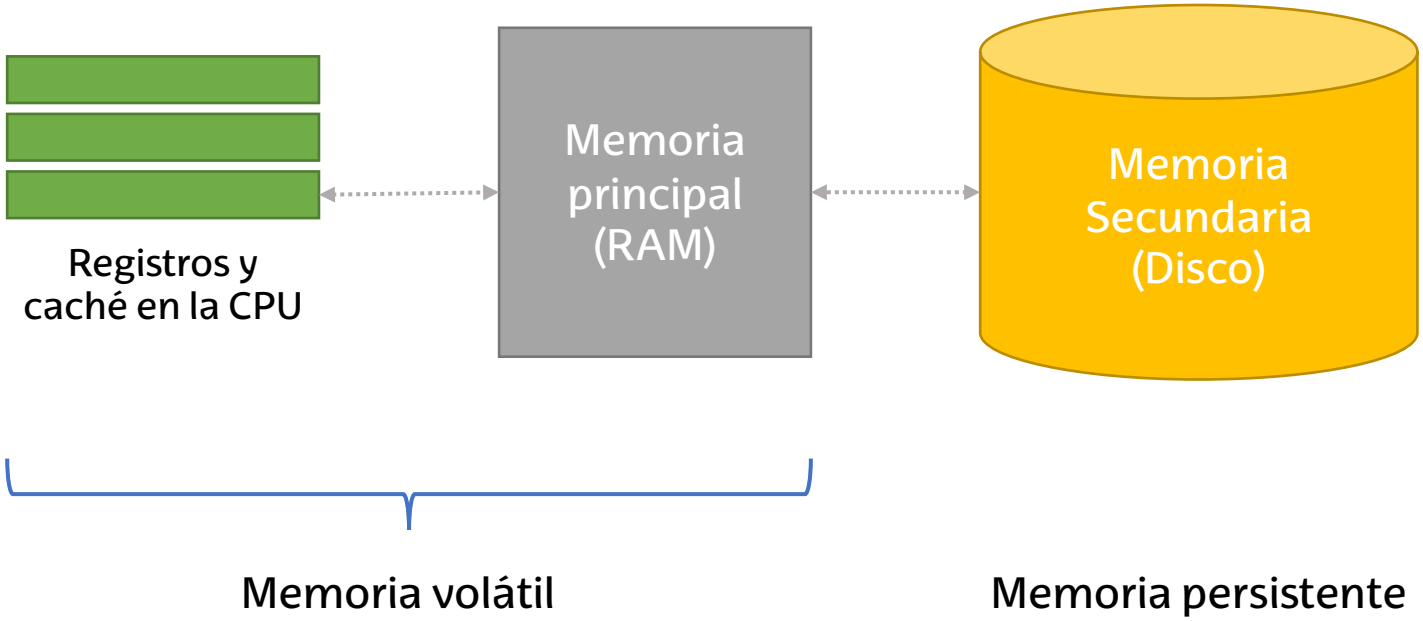
- Propiedades ACID:

- *Isolation*
  - La transacción como unidad lógica de concurrencia.
  - La ejecución de una transacción no puede interferir en la ejecución de otra transacción.

- *Durability*
  - La transacción como unidad lógica de recuperación.
  - Los cambios realizados por una transacción confirmada (tras un COMMIT) deben persistir en la base de datos.

# Jerarquía de memoria

- Principales niveles de memoria en un computador:



# Jerarquía de memoria

- Comparativa orientativa de latencias[1]:

| Operación | Latencia (µs) |
|---|---|
| Leer 1 MB secuencialmente en memoria RAM | 250 |
| Leer 1 MB secuencialmente en un disco SDD | 1,000 |
| Leer 1 MB secuencialmente en un disco tradicional | 20,000 |

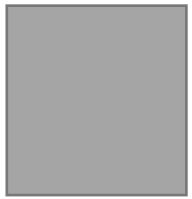
- Comparativa de ancho de banda[1]:

| Memoria | Ancho de banda aproximado (GB/s) |
|---|---|
| Caché L3 | ~100.0 |
| Memoria RAM DDR4 – 3200 Mhz | 25.6 |
| Disco SSD | 4.4 (lectura), 6.8 (escritura) |

[1]Ver fuentes de esta información en la última diapositiva.

# Jerarquía de memoria

- La memoria RAM contiene el proceso del SGBD en ejecución.
  - Todos los cálculos y cambios se realizan primero en la RAM.
    - Deben ser enviados a memoria secundaria para su almacenamiento persistente.



- La memoria secundaria contiene los datos a conservar de forma permanente.
  - Los datos se organizan en ficheros.
  - Es la memoria más lenta de la jerarquía.
    - Cuanto menos accesos, mejor rendimiento del SGBD.

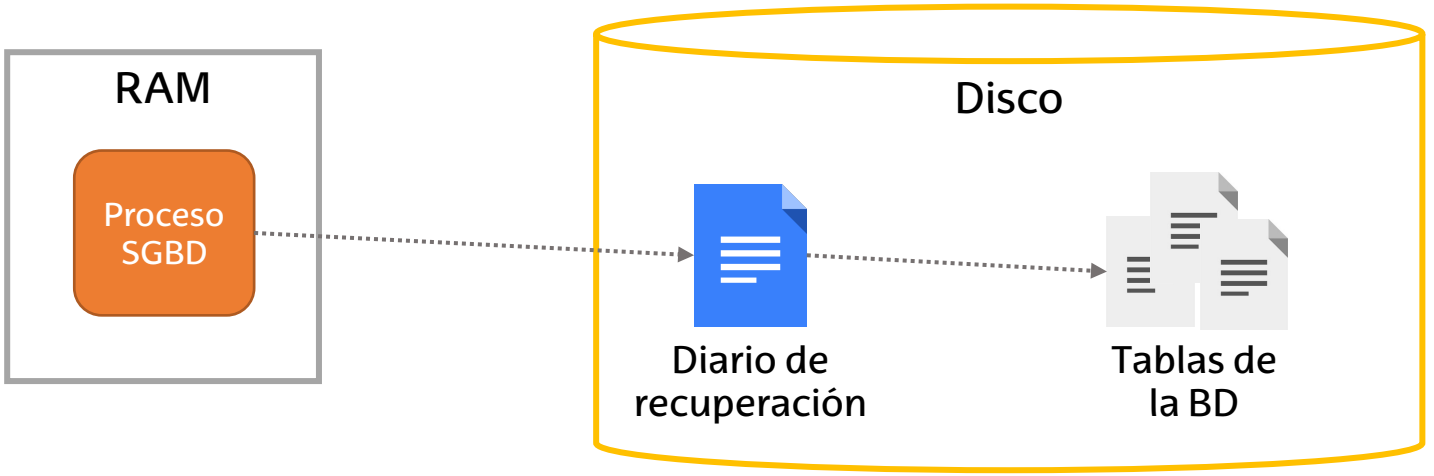
# Proceso de recuperación

- Existen diferentes tipos de fallos posibles para un SGBD:
  - Fallos físicos, p.e.:
    - Rotura de componentes, fallo de suministro eléctrico, ...
  - Fallos software, p.e.:
    - Fallo en el SO, en los drivers, ...
  - Fallos al ejecutar una transacción:
    - División por cero, excepciones del sistema, ...

- El SGBD debe asegurar que se cumplan las propiedades ACID, a pesar de los fallos.

# Proceso de recuperación

- Los SGBD* disponen de un diario en el que se escriben las operaciones realizadas por las transacciones.
  - Las operaciones se escriben en el diario antes de que los cambios se realicen en las tablas de la BD.
- El diario es un fichero en disco.




*La gran mayoría de SGBDs relacionales orientados a aplicaciones estándar.

# Proceso de recuperación

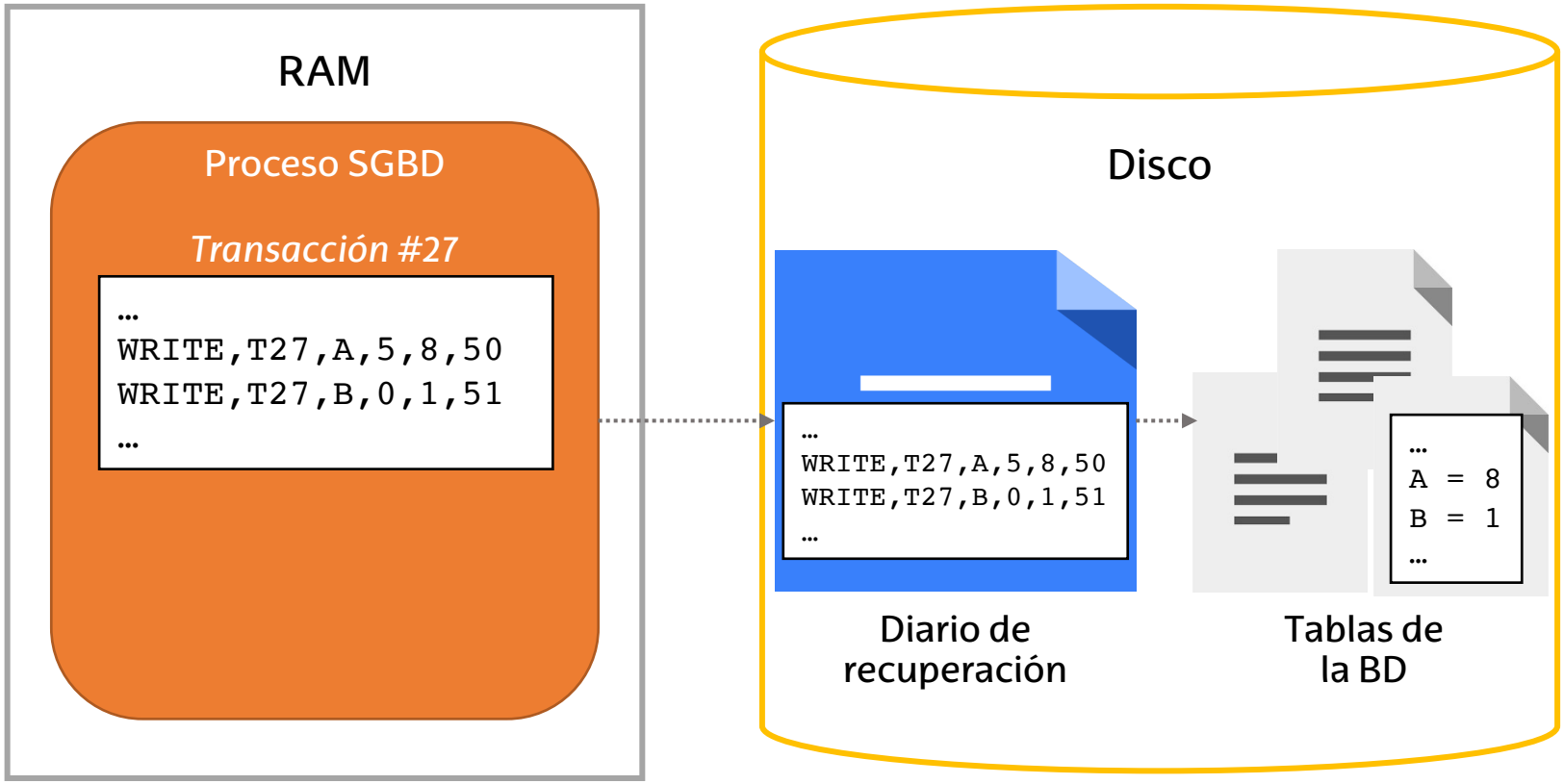
- El diario es la base del proceso de recuperación.
  - En caso de fallo, se utiliza para recuperar las operaciones de las transacciones que estaban en ejecución.
- Sirve para monitorizar la ejecución de las transacciones:
  - Inicio, fin, confirmación, cancelación, operaciones sobre datos.
- Los datos que se modifican deben escribirse en el diario antes que en la BD.
  - Este mecanismo se denomina "*Write ahead logging*".

# Proceso de recuperación

- El diario recoge las operaciones que realizan las transacciones:
  - `START_TRANSACTION, <T>, <time>`
    - Indica el inicio de transacción T.
  - `READ, <T>, <X>, <V>, <time>`
    - La transacción T ha leído el elemento X de la BD y su valor es V.
  - `WRITE, <T>, <X>,<oldV>, <newV>, <time>`
    - La transacción T ha modificado el elemento X de la BD. Su valor antes era oldV y tras la modificación es newV.
  - `COMMIT, <T>, <time>`
    - La transacción T ha finalizado correctamente y sus cambios deben escribirse en la BD.
  - `ROLLBACK, <T>, <time>`
    - La transacción T ha finalizado, sus cambios no deben escribirse en la BD.

# Proceso de recuperación

- Ejemplo:



# Proceso de recuperación

- En caso de fallo, el sistema de recuperación revisa el diario y aplica las siguientes operaciones para cada transacción T:
  - `REDO(T)`: se re-escriben valores nuevos en la BD (`newV`).
  - `UNDO(T)`: se escriben los valores antiguos en la BD (`oldV`).

- Recuperarse de un fallo consistirá en deshacer (`UNDO`) o rehacer (`REDO`) algunas operaciones a partir del contenido del diario.

# Recuperación en MySQL

- En MySQL, el diario de recuperación se llama "Redo Log"[1].
  - Es utilizado por el sistema de recuperación de InnoDB
    - InnoDB es el motor de almacenamiento por defecto de MySQL.
  - Por defecto, se encuentra activado.
    - Podemos comprobarlo con el siguiente comando:

```
mysql> SHOW GLOBAL STATUS LIKE 'Innodb_redo_log_enabled';
```

    - No es recomendable desactivarlo.

[1]MySQL 8 Docs, Redo Log: https://dev.mysql.com/doc/refman/8.0/en/innodb-redo-log.html

# Recuperación en MySQL

- En MySQL, el diario de recuperación se llama “Redo Log”
  - Por defecto, el diario se almacena en varios ficheros que ocupan un total de 100 MB.
  - Mostrar los ficheros que contienen el Redo Log:
    - Por defecto, en la carpeta /var/lib/mysql/#innodb_redo.

```
mysql> select file_name from performance_schema.innodb_redo_log_files;
```

  - Mostrar el tamaño total del Redo Log:
    - Se muestra en Bytes.

```
mysql> select @@global.innodb_redo_log_capacity;
```

# Recuperación en MySQL

- En MySQL, el diario de recuperación se llama “Redo Log”.
  - Son ficheros escritos en binario, no pensados para su manipulación manual.
  - Para leer su contenido, utilizar el comando “strings” de Linux.
    - Ejemplo para mostrar las últimas 20 líneas.

```
# strings ib_redo6 | tail –n 20
```

    - Salida de ejemplo:

```
id=159;root=4;space_id=8;table_id=1069;trx_id=6947;
InnoDB
$M`,
```

  - Más información: https://dev.mysql.com/doc/refman/8.0/en/innodb-redo-log.html

# Ejercicio 2

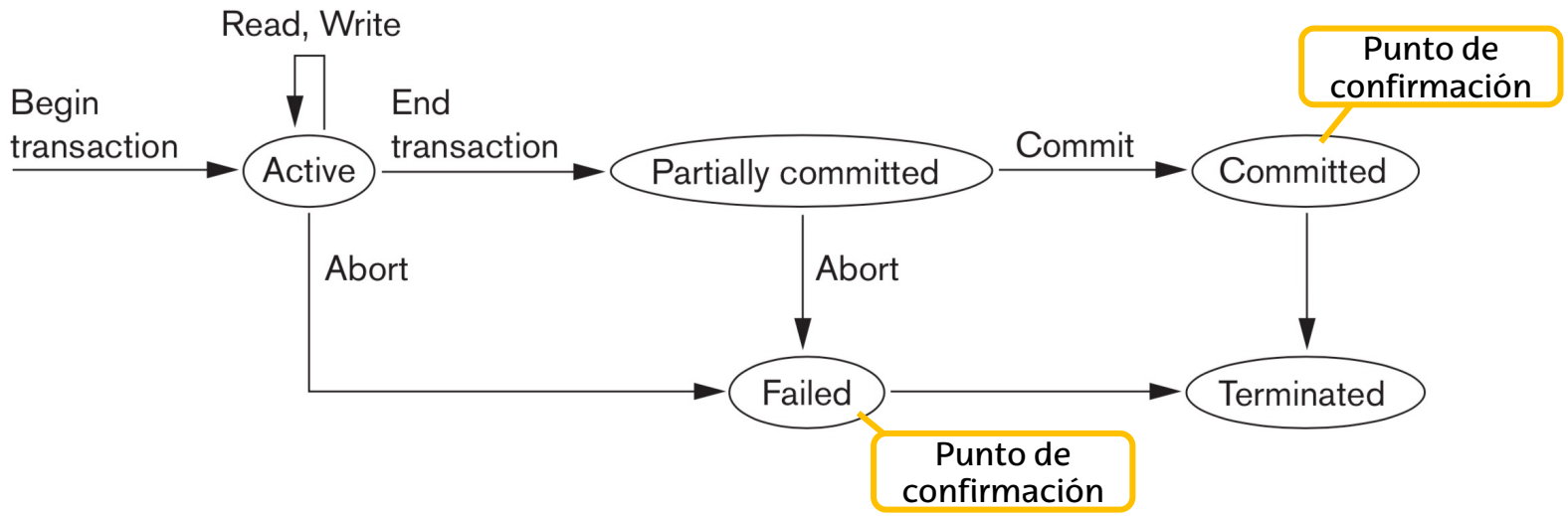
- Crear una BD “BDTransac”, que contenga una tabla “TablaTransac” con 2 columnas: id (int), texto (text)
  - Después, introducir 2 líneas con datos aleatorios en TablaTransac.

- Iniciar manualmente una transacción:
  1) Realizar la inserción de 1 nueva línea en TablaTransac.
  2) Sin finalizar la transacción, detener el proceso MySQL.
     - Usar “service mysql stop” en una terminal y sesión SSH aparte.

- Buscar en el Redo Log indicios de la transacción interrumpida.
- Iniciar el proceso MySQL y verificar qué datos hay en “TablaTransac”.

# Proceso de recuperación

- Ciclo de vida de una transacción.




- *Commit* implica una finalización exitosa.
- *Abort* implica un error y, por tanto, una operación Rollback.

[1]Imagen: "Fundamentals of Database Systems", 7ª Edición

# Proceso de recuperación

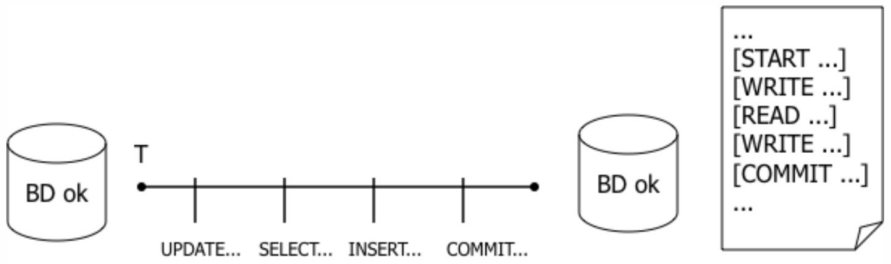
- En este tema vamos a usar una versión simplificada del proceso de recuperación.
  - Consideramos 1 único fichero para el diaro.

- Cada SGDB gestiona el diario recuperación de forma diferente:
  - P.e. MySQL utiliza 2 diarios:
    - Redo Log: https://dev.mysql.com/doc/refman/8.0/en/innodb-redo-log.html
    - Undo Log: https://dev.mysql.com/doc/refman/8.0/en/innodb-undo-logs.html

# Proceso de recuperación

- Cuando una transacción T realiza `COMMIT`:
  - Todas las operaciones de la transacción se ejecutaron con éxito.
  - El efecto de las operaciones se anotó en el diario, incluyendo COMMIT.
  - T ha llegado a un punto de confirmación.

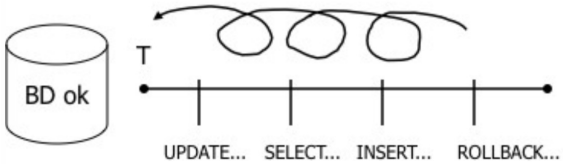
- Después de `COMMIT`:
  - T está confirmada.
  - Sus cambios son permanentes en la BD.



# Proceso de recuperación

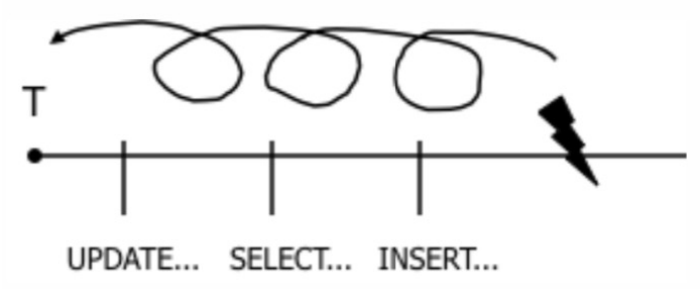
- Cuando una transacción T realiza `ROLLBACK`:
  - T ha resultado fallida y sus operaciones no deben tener efecto.
  - El efecto de las operaciones se anotó en el diario, incluyendo `ROLLBACK`.
  - T ha llegado a un punto de confirmación.

- Después de `ROLLBACK`:
  - T está cancelada.
  - Sus operaciones no deben escribirse en la BD.



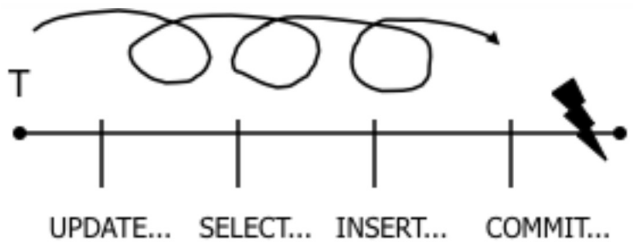

# Proceso de recuperación

- Si sucede un fallo cuando una transacción T está en ejecución, sus operaciones se deben deshacer.
  - T no alcanzó un punto de confirmación
  - El diario puede contener algunas de sus operaciones, pero no está la operación `COMMIT`.
  - El proceso de recuperación aplica `UNDO(T)`.



# Proceso de recuperación

- Si sucede un fallo cuando una transacción T ha sido confirmada, entonces se debe rehacer.
  - Las operaciones y el COMMIT asociados a la transacción se encuentran en el diario.
  - No es seguro que todos sus cambios se hayan escrito en la BD.
  - El proceso de recuperación aplica `REDO(T)`.



# Proceso de recuperación

- `UNDO(T)` implica deshacer cada una de las operaciones a partir de las anotaciones en el diario.
  - Se empieza por la última y en orden inverso.
  - Después de ejecutar: `UNDO( [WRITE,<T>,<X>,<oldV>, <newV>,<time>] )` en la BD se cumple: `<X> = <oldV>`

- `REDO(T)` implica rehacer cada una de las operaciones a partir de las anotaciones en el diario.
  - Se empieza por la primera y en el mismo orden.
  - Después de ejecutar: `REDO( [WRITE,<T>,<X>,<oldV>, <newV>,<time>] )` en la BD se cumple: `<X> = <newV>`

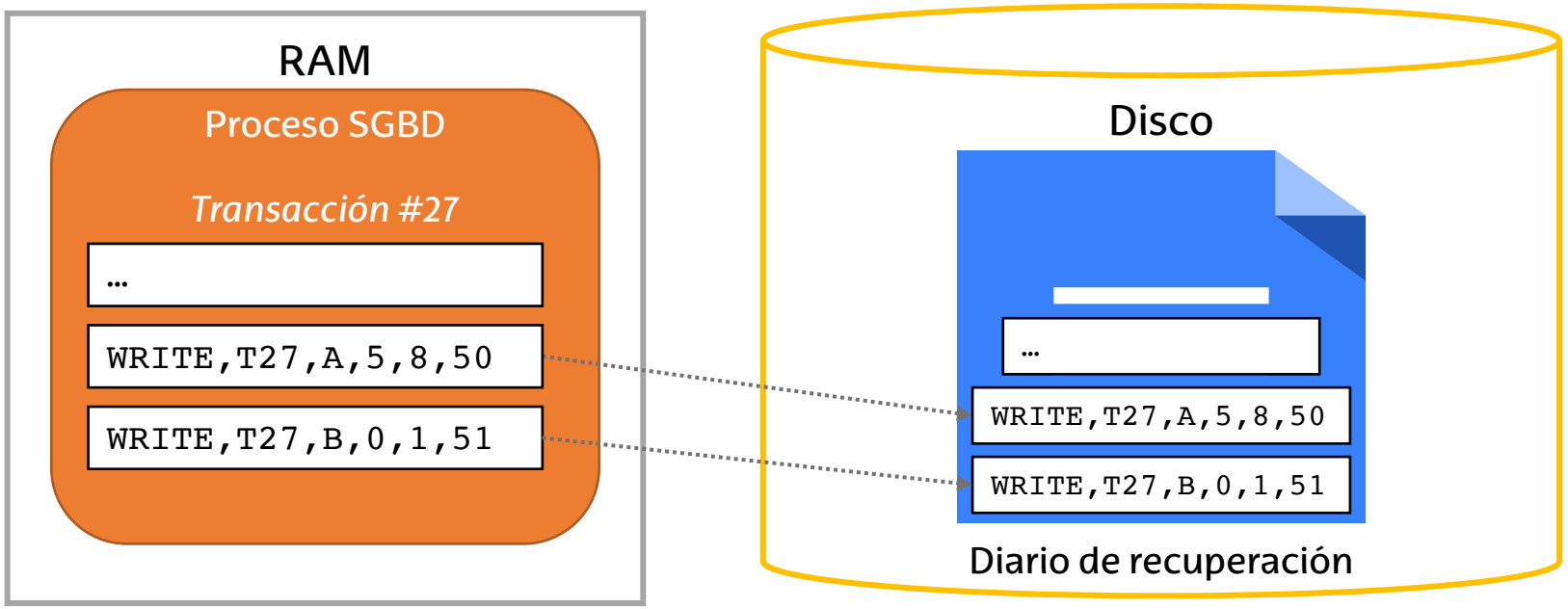
# Proceso de recuperación

Al gestionar el diario, hay 2 parámetros a considerar:

- Frecuencia de actualización del diario:
  - Inmediata
  - Diferida
- Frecuencia de actualización de la BD:
  - Diferida
    - Algoritmo No-Deshacer/Rehacer
  - Inmediata
    - Algoritmo Deshacer/Rehacer

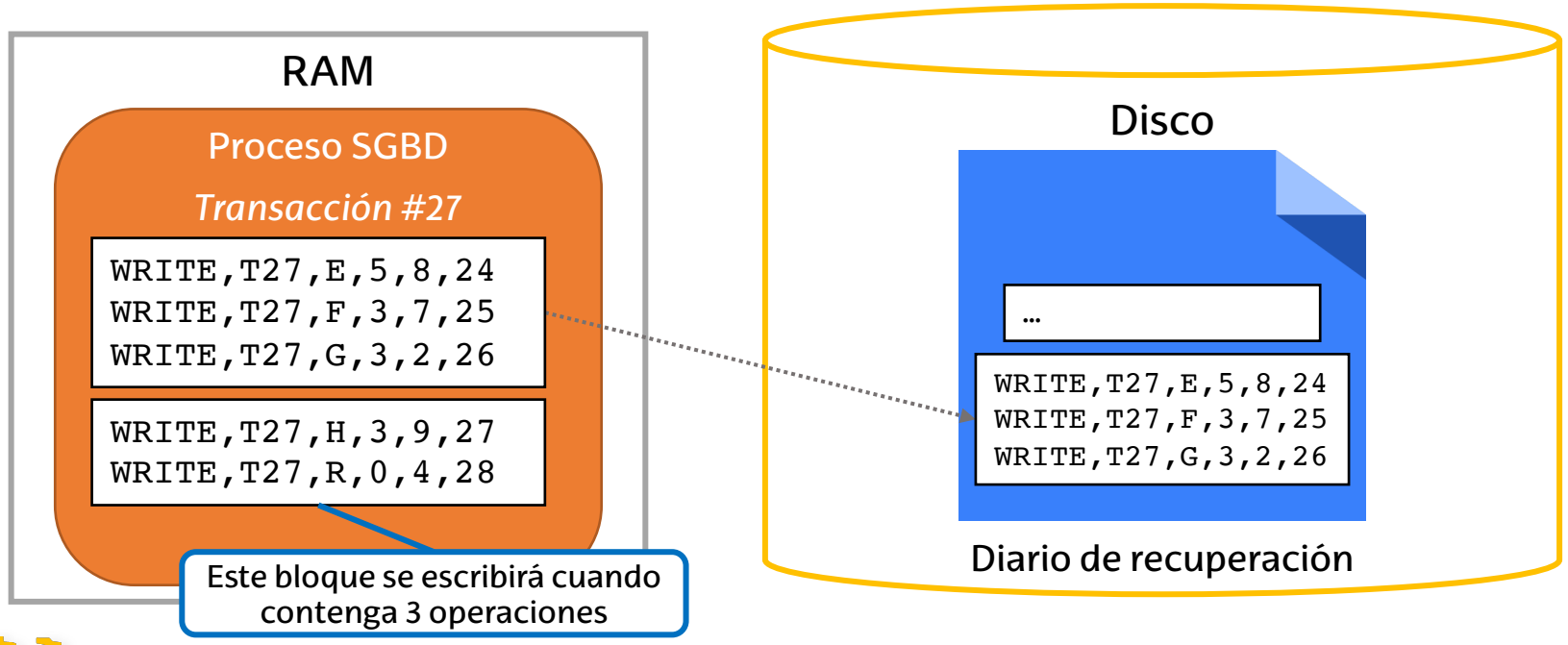
# Actualización del diario: Inmediata

- El SGBD escribe cada operación en el diario inmediatamente después de realizarse.
  - Ejemplo:



# Actualización del diario: Diferida

- En RAM, se crean bloques que contienen un nº de operaciones realizadas por las transacciones.
  - Se escribe el bloque entero desde RAM al diario.
  - Ejemplo con tamaño de bloque = 3 operaciones.



# Actualización del diario

- Inmediata
  - Mayor fiabilidad
  - Peor rendimiento
    - Más escrituras en disco.
- Diferida
  - Mayor rendimiento
    - Realizar menos escrituras (aunque sean de mayor tamaño) hace un mejor uso de la jerarquía de memoria
  - Mayor riesgo.
    - En caso de fallo, las operaciones en un bloque sin completar se pierden.

# Actualización del diario

- Cada cierto tiempo, el SGBD realiza una operación llamada **Checkpoint**.
  - En escritura diferida de diario, provoca que el bloque en RAM se envíe al diario aunque no haya alcanzado su tamaño total.
  - Revisa en el diario las modificaciones a realizar por las transacciones finalizadas con COMMIT y las escribe en la BD.
  - Crea una lista con las transacciones activas en ese momento.
    - Las que no hayan finalizado con `COMMIT` o `ROLLBACK`.
- Esta operación se representa en el diario como:
  - `CHECKPOINT, <time>.`
    - Indica que se realiza un Checkpoint en el instante ‹time›.

# Actualización del diario

- Los puntos `CHECKPOINT` permiten:
  - Recorrer el diario desde el último `CHECKPOOINT`.
  - Ignorar las transacciones confirmadas antes del último `CHECKPOINT`.

- En escritura diferida de diario, hay 3 operaciones que fuerzan la escritura de bloques en memoria al fichero del diario en disco:
  - `CHECKPOINT`
  - `ROLLBACK`
  - `COMMIT`

# Actualización de la BD

- Hay 2 formas de gestionar las actualizaciones de la BD
- Actualización diferida:
  - Los cambios que realizaría una transacción se escriben en la BD después de haber finalizado con Commit.
    - No hay cambios en la BD hasta la confirmación de la transacción.
- Actualización inmediata:
  - Los cambios que realiza una transacción pueden escribirse en la BD sin que haya llegado a confirmarse.

# Actualización de la BD: Diferida

- Se consideran 2 situaciones de fallo:


1) Si el fallo sucede antes de realizar Commit, no es necesario deshacer nada.
2) Si el fallo sucede después de realizar Commit, es necesario rehacer sus operaciones

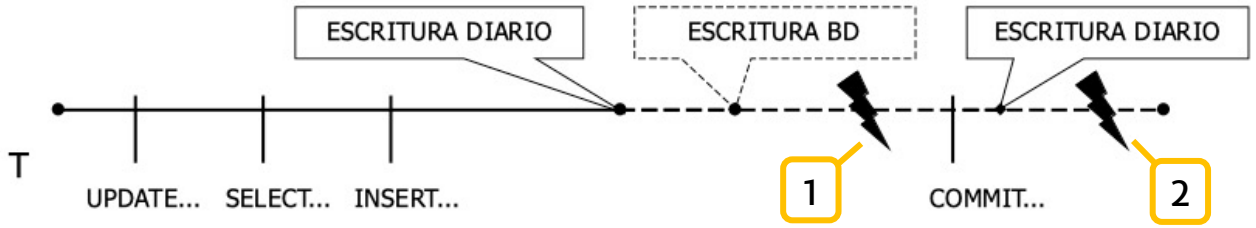
- Adecuado para entornos con transacciones cortas.
- Se aplica el algoritmo No-Deshacer/Rehacer.

# Algoritmo No-Deshacer/Rehacer

1. Crear 2 listas vacías: *Activas* y *Confirmadas*
2. Inicializar *Activas* con la lista de transacciones activas en el último punto *checkpoint* del diario.
3. Examinar el diario a partir del último *checkpoint*:
    1. Añadir a la lista *Activas* las transacciones que se hayan iniciado.
    2. Mover de la lista *Activas* a *Confirmadas* las transacciones que hayan finalizado con **Commit**, y eliminar de la lista *Activas* las transacciones que hayan finalizado con **Rollback**.
4. Al terminar de examinar el diario:
    1. Rehacer las operaciones **Write** de las transacciones en la lista *Confirmadas*, en el mismo orden en el que aparecen en el diario.
    2. Reiniciar las transacciones de la lista *Activas*.

# Actualización de la BD: Inmediata

- Se consideran 2 situaciones de fallo:




1) Si el fallo sucede antes de realizar Commit, es necesario deshacer sus operaciones.
2) Si el fallo sucede después de realizar Commit, es necesario rehacer sus operaciones

- Adecuado para entornos con transacciones largas.
- Se aplica el algoritmo Deshacer/Rehacer.

# Algoritmo Deshacer/Rehacer

1. Crear 2 listas vacías: *Activas* y *Confirmadas*
2. Inicializar *Activas* con la lista de transacciones activas en el último punto *checkpoint* del diario.
3. Examinar el diario a partir del último *checkpoint*:
    1. Añadir a la lista *Activas* las transacciones que se hayan iniciado.
    2. Mover de la lista *Activas* a *Confirmadas* las transacciones que hayan finalizado con **Commit**, y eliminar de la lista *Activas* las transacciones que hayan finalizado con **Rollback**.
4. Al terminar de examinar el diario:
    1. Deshacer las operaciones **Write** de las transacciones la lista *Activas* en orden inverso a su aparición en el diario.
    2. Rehacer las operaciones **Write** de las transacciones en la lista *Confirmadas*, en el mismo orden en el que aparecen en el diario.
    3. Reiniciar las transacciones de la lista *Activas*.

# Ejercicio 3

- Resolver las siguientes situaciones de la hoja de ejercicios:

- Diario nº 2:
  - Instante de fallo: 405
  - Actualización de diario: Diferida – Tamaño de bloque: 3
  - Actualización de BD: Inmediata
- Diario nº 2:
  - Instante de fallo: 205
  - Actualización de diario: Diferida – Tamaño de bloque: 9
  - Actualización de BD: Diferida
- Diario nº 3:
  - Instante de fallo: 355
  - Actualización de diario: Diferida – Tamaño de bloque: 7
  - Actualización de BD: Inmediata

# Estructura de InnoDB

- InnoDB es el motor de almacenamiento por defecto de MySQL 8.0.

- Provee un buen balance entre alta fiabilidad y alto rendimiento.
  - Soporta las propiedades ACID.
  - Control de concurrencia multi-usuario.

# Estructura de InnoDB

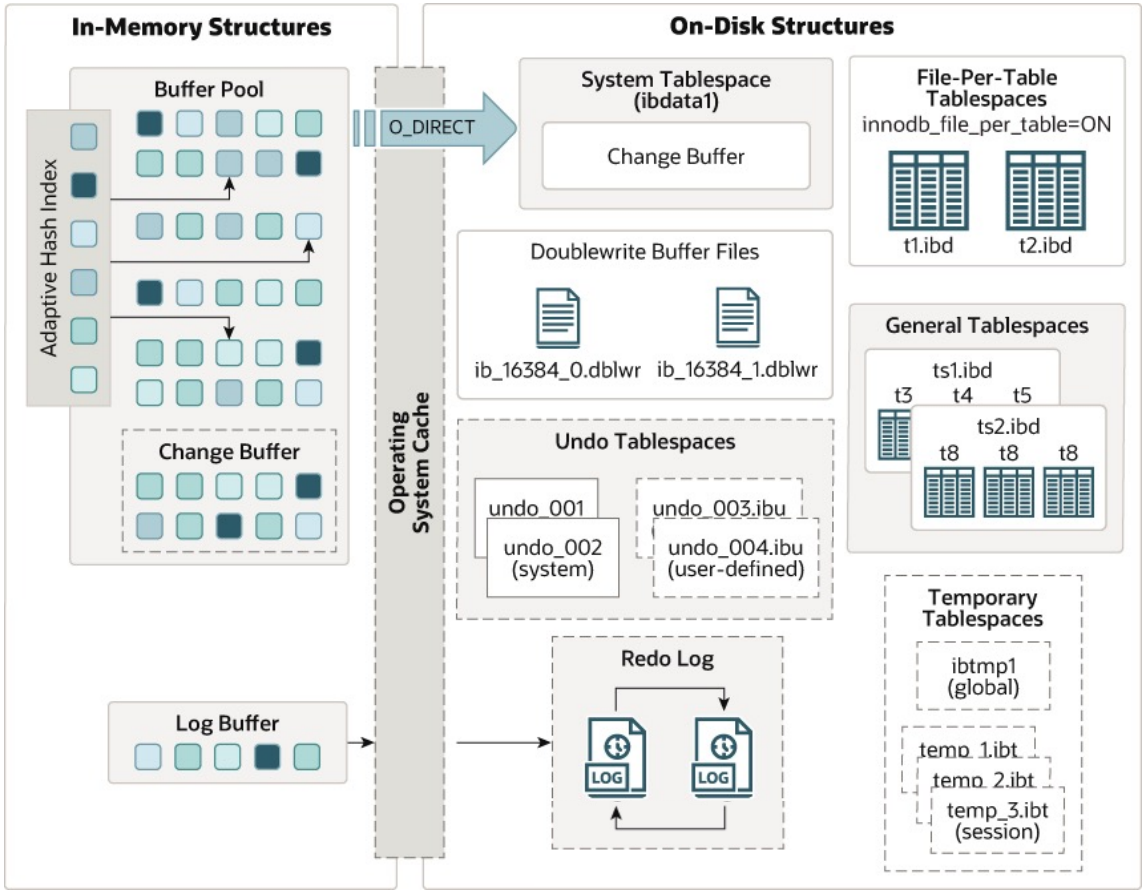
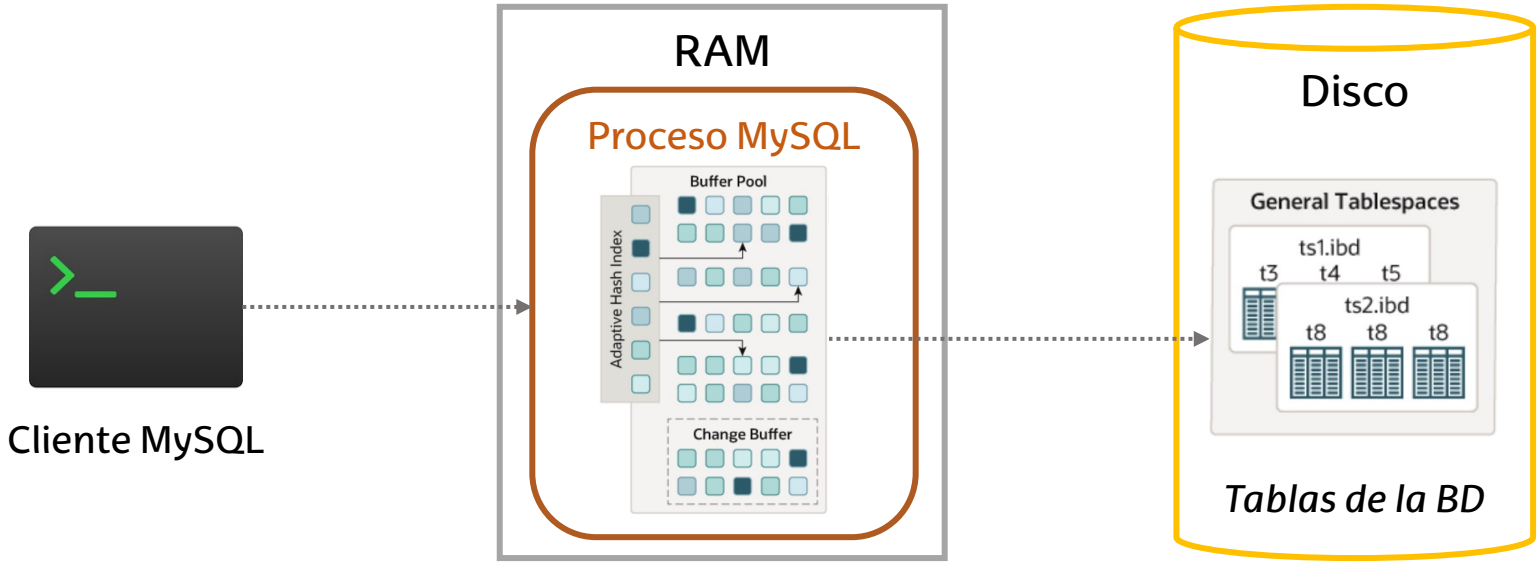



Imagen: https://dev.mysql.com/doc/refman/8.0/en/innodb-architecture.html

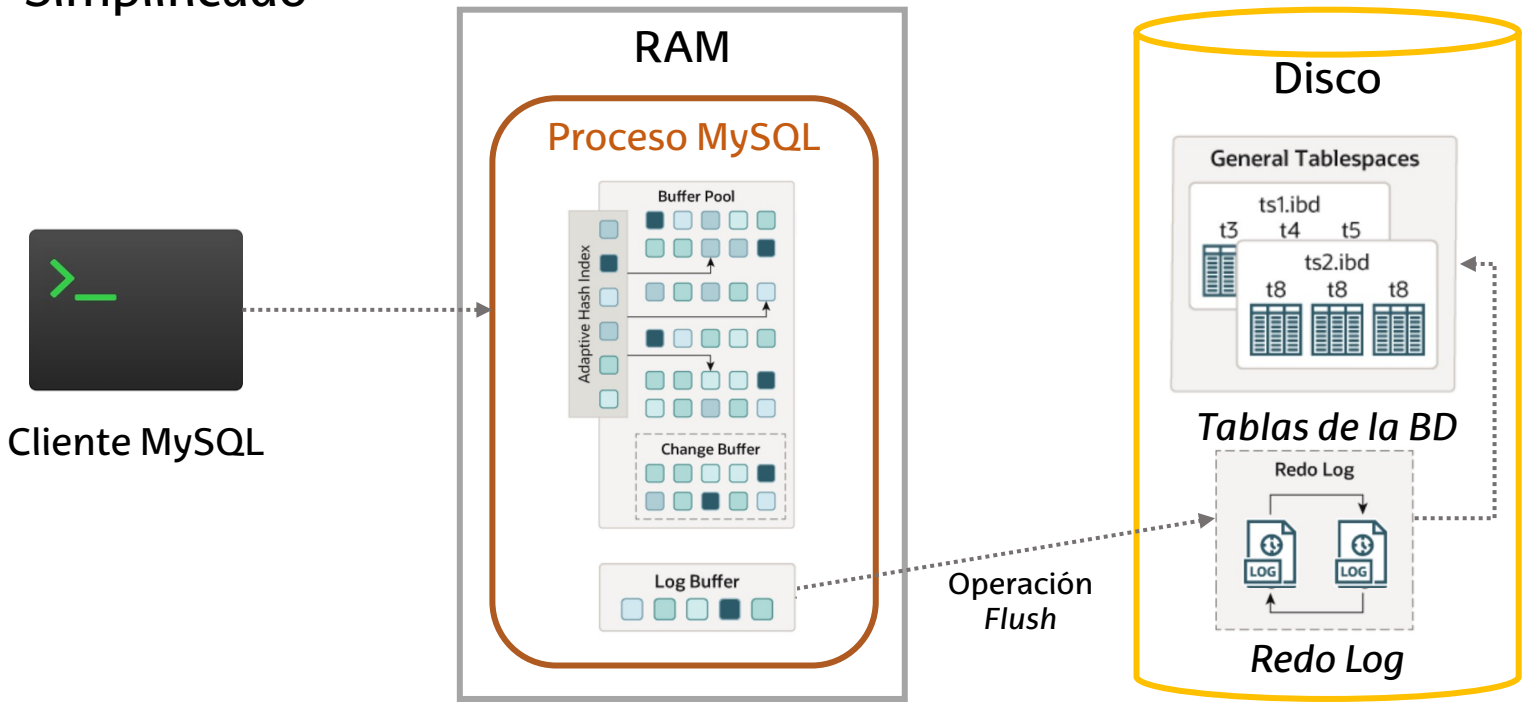
# Estructura de InnoDB

- Flujo de una operación de lectura:



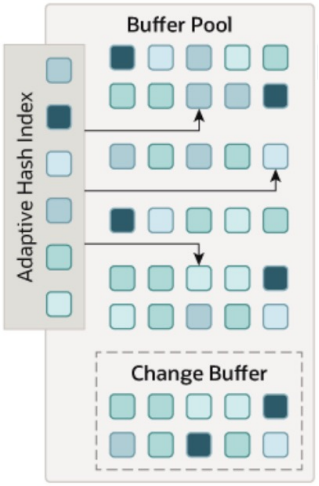
# Estructura de InnoDB

- Flujo de una operación de escritura:
  - Simplificado



# Estructura de InnoDB

- Estructuras en memoria:

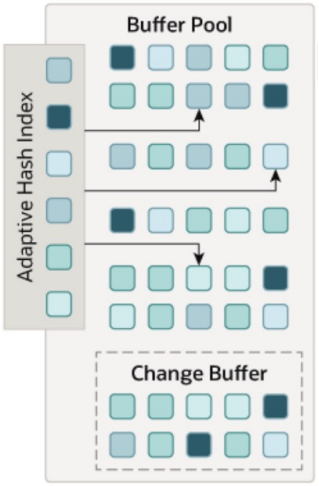



- *Buffer Pool:*
  - Caché de datos
  - Permite que los datos más usados se lean sin acceder a disco.
    - Mayor velocidad
  - Organiza los datos con una política LRU
    - *Least Recently Used*
  - Por defecto, ocupa 128 MB.
    - Se puede verificar con *(resultado en Bytes)*:

```
mysql> show global variables like 'innodb_buffer_pool_size';
```

# Estructura de InnoDB

- Estructuras en memoria:




- *Buffer Pool:*
  - Se puede obtener una métrica de su eficiencia: % de operaciones de lectura que no necesitaron acceder a disco.
  - Se calcula con 2 variables de estado:
    - **Innodb_buffer_pool_read_requests**: nº de operaciones de lectura.
    - **Innodb_buffer_pool_reads**: nº de operaciones de lectura que no se pudieron servir desde el Buffer Pool y fueron leídas desde disco.
    - *Obtener su valor con:*

```
mysql> show global status like '<VARIABLE>';
```

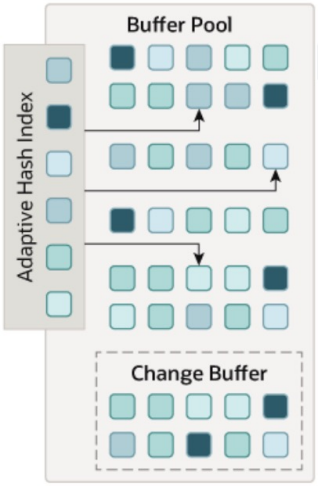
  - Calcular la eficiencia del Buffer Pool:

```
100 - (100 * Innodb_buffer_pool_reads / Innodb_buffer_pool_read_requests)
```

    - Cuanto más cercano a 100, mejor.

# Estructura de InnoDB

- Estructuras en memoria:

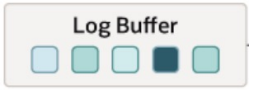



- *Buffer Pool:*
  - Después de un reinicio de MySQL, si el Buffer Pool estuviese vacío, perjudicaría al rendimiento.
    - Al comienzo, se pedirían todos los datos a disco.
  - Para evitar problemas de rendimiento:
    - Al apagar MySQL, un porcentaje del Buffer Pool se escribe a disco.
    - Al inicio de MySQL, el Buffer Pool guardado se carga desde disco.
    - El porcentaje guardado a disco se puede consultar:

```
mysql> show global variables like
'innodb_buffer_pool_dump_pct';
```

# Estructura de InnoDB

- Estructuras en memoria:




  - *Log Buffer*:
    - Contiene los datos de las transacciones que se deben escribir en el Redo Log.
    - Por defecto, ocupa 16 MB de RAM.
      - Su tamaño se puede consultar con:

```
show global variables like 'innodb_log_buffer_size';
```

    - La frecuencia con la que los datos se copian del *Log Buffer* al *Redo Log* se controla con la variable **innodb_flush_log_at_trx_commit**.
      - Más info: https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html#sysvar_innodb_flush_log_at_trx_commit

# Ajuste de InnoDB

- La configuración de los parámetros de InnoDB afecta directamente al rendimiento de MySQL.
  - P.e. un *Log Buffer* pequeño puede implicar un mayor nº de escrituras en disco.

- Hay cambios que pueden tener efectos indeseados.
  - P.e. un mayor % de volcado de *Buffer Pool* a disco implica:
    - Mayor rendimiento de las consultas desde el comienzo. 👍
    - Mayor tiempo de carga de MySQL. 👎

# Ajuste de InnoDB

- En MySQL 8.0 se introdujo la variable de sistema. *innodb_dedicated_server*.
- Si se configura como ON, MySQL ajusta varios parámetros automáticamente.
  - Se configura en el fichero *mysqld.cnf*.
- Se debe activar sólo en entornos donde MySQL sea la única aplicación.
  - P.e. en una máquina virtual sin más aplicaciones.

# Ajuste de InnoDB

- Parámetros alterados por *innodb_dedicated_server*:
  - *innodb_buffer_pool_size*: Tamaño de Buffer Pool.
  - *innodb_redo_log_capacity*: Tamaño de Redo Log.
  - *innodb_flush_method*: Método de escritura a disco.

  - Valores: https://dev.mysql.com/doc/refman/8.0/en/innodb-dedicated-server.html

- Es posible no ver un incremento inmediato en el rendimiento tras su activación.
  - Los cambios afectan en situaciones concretas.

# Ejercicio 4

- Revisar vuestros valores actuales para las variables que afecta *innodb_dedicated_server*.
- Activar el parámetro Innodb_dedicated_server.
  - Necesario reiniciar MySQL.
- Revisar los valores de las variables afectadas, ¿han cambiado?
  - Revisar el manual de *innodb_dedicated_server* y verificar que los nuevos valores se corresponden con los heurísticos definidos ahí.

# Bibliografía


- General:
  - R. Elmasri, S. B. Navathe. “Fundamentals of Database Systems”, 7th edition, Pearson, 2017.
  - A. Silberschatz, H. F. Korth, S. Sudarshan. “Database Systems Concepts”, 7th edition, McGraw-Hill, 2019.

- Online:
  - Manual oficial de MySQL 8.0
    - https://dev.mysql.com/doc/refman/8.0/en/
  - H.P.P. “MySQL Server Configuration for High Performance”
    - https://youtu.be/d37_2tFPstU
  - DistributedDBA. “Basic InnoDB Tuning in MySQL 8.0”
    - https://youtu.be/grWwLEIPKrg


# Bibliografía

- Sección “Jerarquía de memoria”:
  - General:
    - https://scoutapm.com/blog/understanding-disk-i-o-when-should-you-be-worried
  - Comparativa de latencias:
    - https://gist.github.com/jboner/2841832?permalink_comment_id=3272283#gistcomment-3272283
  - Comparativa de anchos de banda:
    - Caché: https://en.wikipedia.org/wiki/Memory_hierarchy
    - RAM: https://www.transcend-info.com/Support/FAQ-292
    - SSD: https://en.wikipedia.org/wiki/Solid-state_drive