ATOMICIDAD Y DURABILIDAD/ RECUPERACIÓN ANTE FALLAS

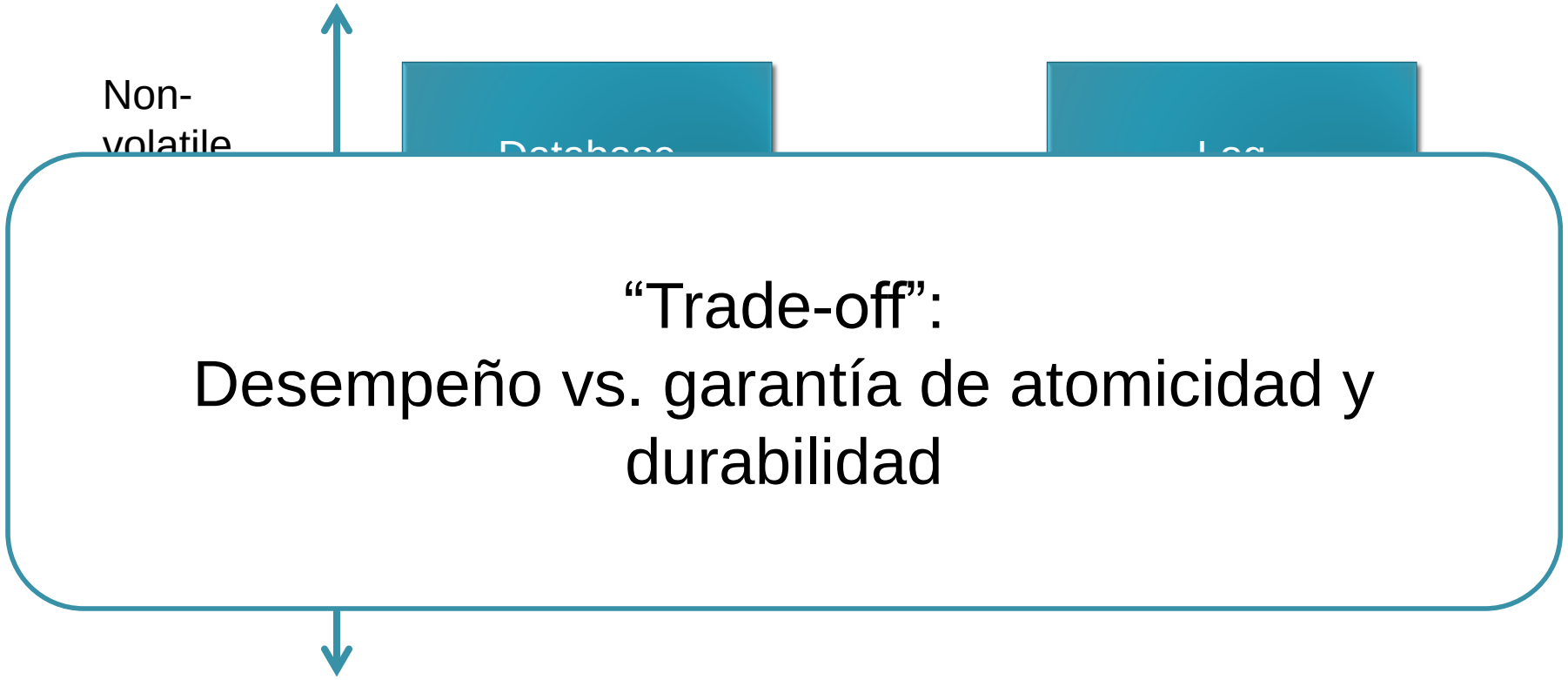
Diana Benavides
José Abásolo

Basada en: María del Pilar Villamil

Transacción

- Conjunto de operaciones que deben cumplir las propiedades de: **atomicidad**, ***consistencia, aislamiento y* durabilidad *o permanencia***.

Organización de la memoria en sistemas de actualización inmediata



Basado en KIFER, et al. Database systems: an application-oriented approach. Pág. 921.

Logs – bitácoras

- Información que manejan:
  - Id. Transacción, tipo de operación, id. Registro
  - Apuntador al anterior registro asociado a la transacción.
  - Registro de actualización: Estado anterior (before-image), posterior (after-image)
  - Registros de *Commit, Rollback*
  - Datos de los cursores abiertos
  - Registro de *checkpoint*
  - Registro de *savepoint*

Tipos de fallas en una BD

- “Crash”: Falla intempestiva del sistema
  - ¿Cómo garantizar atomicidad y durabilidad? → estrategia de recuperación:

1. **Rollback (UNDO)**
2. Rollforward (REDO)

- Falla del(os) medio(s) de almacenamiento
  - Mecanismos de redundancia

Atomicidad y durabilidad en sistemas de actualización inmediata: Rollback (UNDO)

1. Se inserta un **“update record”** al log, cuando una transacción ejecuta un cambio sobre una base de datos → **Id transacción + “undo record”**.

2. En el momento en que la transacción termina (o aborta) se escribe un registro de “commit” o “rollback” en el log (incluye el id de la transacción).

3. **En el momento de una falla (“crash”), se revisa el log (hacia atrás) para determinar las transacciones que realizaron cambios y que no fueron terminadas (no tienen registro de commit ó rollback en el log)**.

Implicaciones:

1. Se debe poder determinar el inicio de la transacción (begin record).
2. Para incrementar eficiencia, pueden utilizarse “savepoints”, puntos en los cuales la transacción puede ser desecha.
3. La transacción hace commit (ó abort) si y solo si se pudo escribir el registro commit (o abort) correspondiente en el log, y suelta los candados.

Atomicidad y durabilidad en sistemas de actualización inmediata: Checkpoints

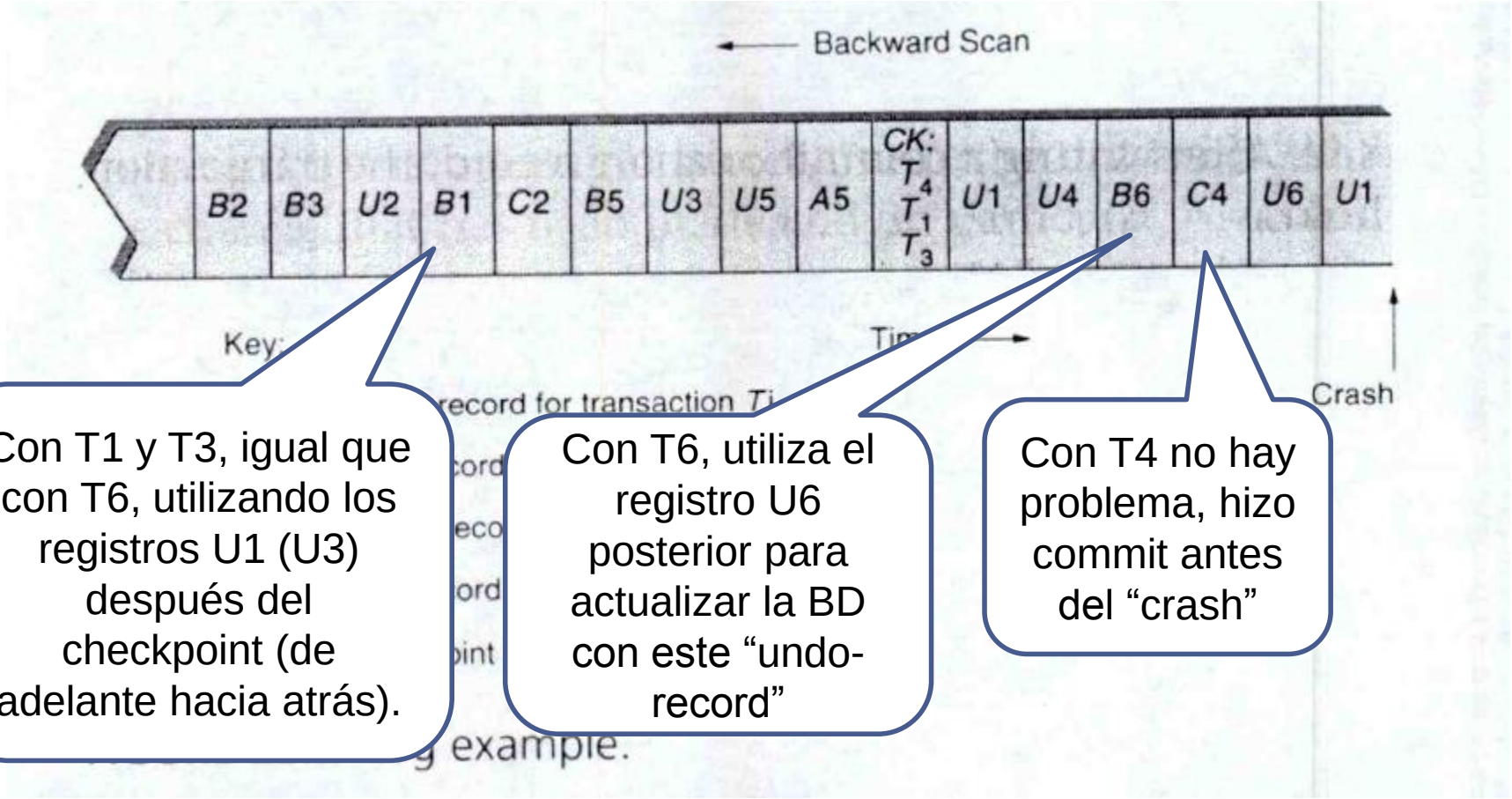
¿Cómo garantizar que la revisión del log, en el momento de una falla, se hace de forma eficiente?

Checkpoint:

Periódicamente, el sistema escribe en el log checkpoints, puntos en los cuales se listan los identificadores de las transacciones activas en ese momento.

En el momento de falla, se chequea el log solo hasta encontrar los inicios de las transacciones activas hasta el checkpoint más reciente.

Atomicidad y durabilidad en sistemas de actualización inmediata: Checkpoints



Con T1 y T3, igual que con T6, utilizando los registros U1 (U3) después del checkpoint (de adelante hacia atrás).

Con T6, utiliza el registro U6 posterior para actualizar la BD con este "undo-record"

Con T4 no hay problema, hizo commit antes del "crash"

Atomicidad y durabilidad en sistemas de actualización inmediata: Write-ahead logging

¿Qué operación se realiza primero? ¿Actualizar el ítem en la base de datos o escribir el registro de actualización en el log?

Write-ahead logging:

Escribir primero el registro de actualización en el log, antes de actualizar el ítem en la base de datos.

Si existe una falla entre el momento en que se ha escrito en el log, y la actualización del ítem en la BD, no hay problema, porque el ítem se restaurará al estado original (que es el mismo actual).

Atomicidad y durabilidad en sistemas de actualización inmediata: Write-ahead logging y desempeño

¿Qué implicaciones tiene el uso de Write-ahead logging sobre el desempeño del sistema?

Log-buffer:

Almacenamiento temporal, en memoria volátil, de lo que se almacenará en el log.

Se preserva la propiedad Write-ahead: **Mientras no se escriba el registro al log, no se actualizará el ítem correspondiente en la base de datos**.

Atomicidad y durabilidad en sistemas de actualización inmediata: Write-ahead logging y desempeño

¿Cómo aprovechar el log-buffer para tratar de mantener el desempeño del sistema?

Unforced operation:

El log-buffer escribe al log no-volátil periódicamente (flush).

Forced policy:

Forzar a que el registro (y los registros anteriores) del log-buffer sea(n) escrito(s) en el log no-volátil, cada vez que se requiera actualizar el ítem en la BD.

Forced policy con LSN (Log Sequence Number):

1) Asignar a cada registro del log un LSN.
2) Asignar a la página de la BD que está en la caché el LSN del ítem más reciente.
3) Una vez la página debe ser transferida a disco, si el LSN de la página es igual que el del log-buffer, se fuerza la escritura al log no-volátil.

Atomicidad y durabilidad en sistemas de actualización inmediata: Write-ahead logging y forced policy con LSN

Forced policy con LSN:

Cuando la transacción está lista para hacer commit:

1) Si el último registro de actualización aun está en el log-buffer, forzarlo al log no-volátil → todos los registros de actualización serán durables.

2) Si existe alguna página que no ha sido escrita a memoria no volátil (disco), forzarla → todas las actualizaciones sobre los datos serán durables.

3) Añadir el registro de commit al log-buffer → **cuando sea escrito al log no-volátil, la transacción será durable**.

Atomicidad y durabilidad en sistemas de actualización inmediata: Write-ahead logging y forced policy con LSN

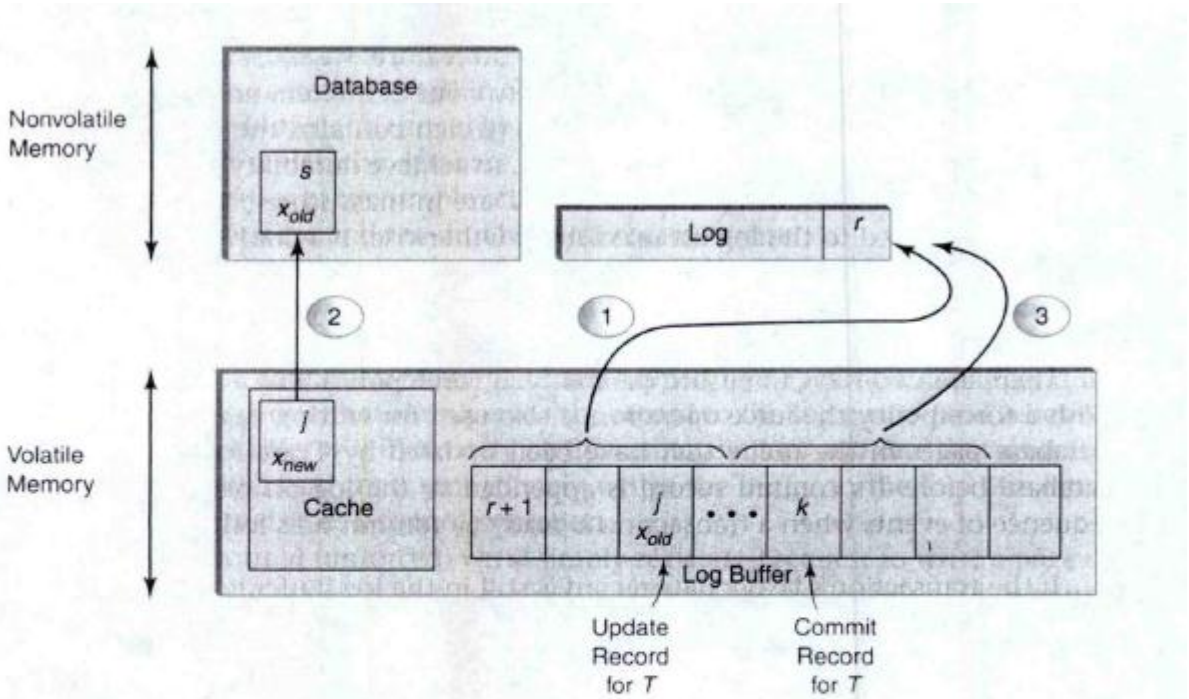


FIGURE 22.4 Implementing durability using a force policy. It might be necessary to force the pages that a transaction has updated out of the cache before the transaction's commit record is written to the log. Before the dirty page updated by T, with LSN value j, can overwrite the earlier version of the page, with LSN s, $s < j$, in the database, it might be necessary to force the log buffer (1) so that the update record with LSN equal to j is on mass storage. After the dirty page has been written, (2) the commit record for T can be appended to the log buffer (3). The log buffer can be written to the log at a later time.

Atomicidad y durabilidad en sistemas de actualización inmediata: Write-ahead logging y forced policy con LSN

¿Implicaciones del uso de este mecanismo?

La escritura del registro de commit en el log debe esperar a que las actualizaciones de los ítems se escriban en memoria no-volátil.

Debe escribirse la página modificada a memoria no-volátil cada vez que hay una transacción que hace commit sobre ella → impacta páginas que son modificadas frecuentemente por distintas transacciones.

Atomicidad y durabilidad en sistemas de actualización inmediata: Roll-forward (REDO) con no-forced policy

Permite escribir los nuevos valores de los ítems de la base de datos (after-images) en el log (además de los valores viejos existentes).

Ahora el registro del commit no tendrá que esperar a que la página sea escriba a memoria no-volátil (disco) → las páginas de datos en caché pueden ser escritas posteriormente ,y el commit puede ser escrito previamente, proveyendo desempeño al sistema.

En el momento de falla:

1) Si no se ha escrito el registro commit en el log, las operaciones de la transacción deben ser deshechas (**rollback**)

2) Si se ha escrito el registro del commit en el log, las operaciones de la transacción deben ser rehechas (**rollforward**)

Atomicidad y durabilidad en sistemas de actualización inmediata: Roll-forward (REDO) con no-forced policy

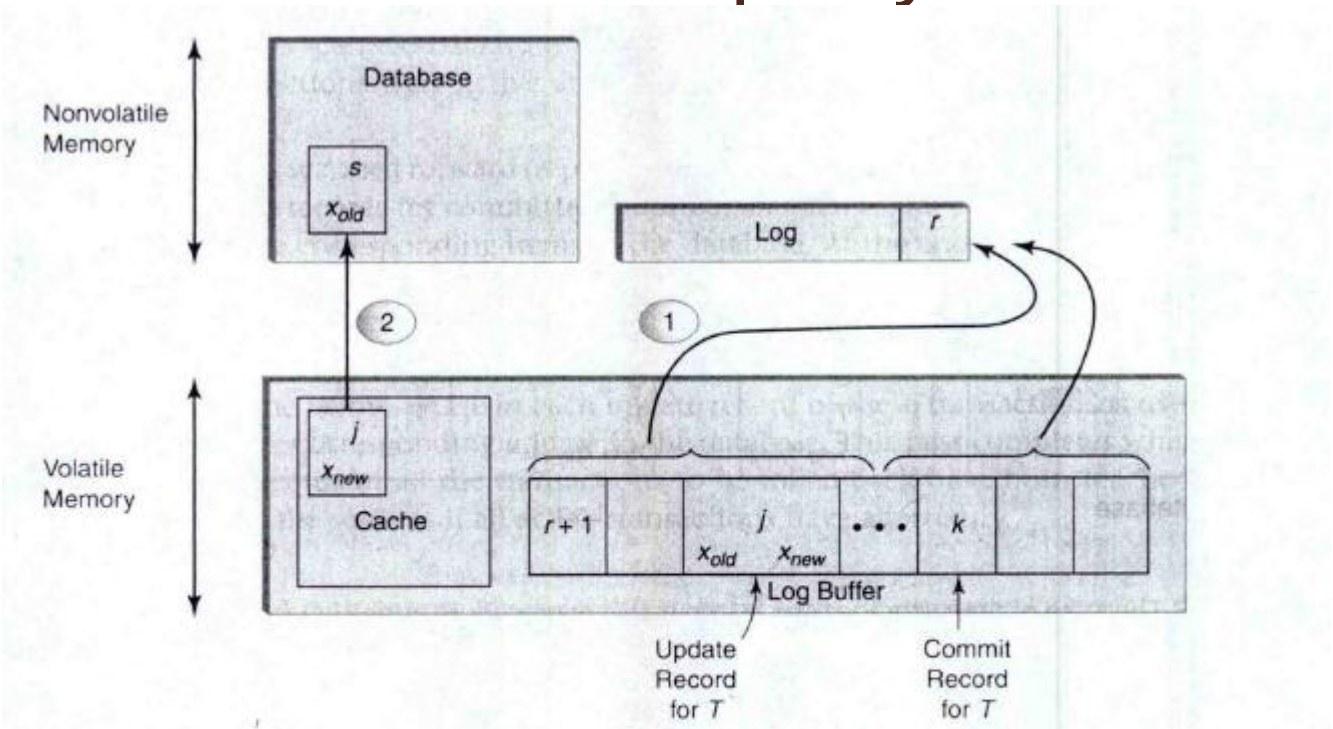


FIGURE 22.5 Implementing durability using an after-image with a no-force policy. The write-ahead feature requires that before a dirty page with LSN j updated by T can overwrite the earlier version of the page, with LSN s, $s < j$, in the database (2), the corresponding update record must be written to the log on mass storage (1). Although the commit record cannot be written before the update record, the relationship between the time the commit record is written and the time the dirty page is written is not constrained.

Atomicidad y durabilidad en sistemas de actualización inmediata: DO-UNDO-REDO y sharp checkpoints

¿Cómo combinar rollback y rollforward?

En el momento de falla:

1) El log se revisa **hacia atrás** hasta el checkpoint más reciente, y se obtienen las transacciones activas en el momento de falla.

2) El log se **revisa hacia delante**, y todas las actualizaciones sobre los ítems de la BD, después del checkpoint, son aplicadas (usando los registros REDO).

3) El log se revisa nuevamente **hacia atrás**, y se deshacen las actualizaciones de las transacciones que no hicieron commit antes de la falla (hasta encontrar el begin de las mismas).

Atomicidad y durabilidad en sistemas de actualización inmediata: DO-UNDO-REDO y sharp checkpoints

¿Qué pasa con las transacciones que abortaron actualizaciones antes de la falla?

El log contendrá dos registros de actualización:

1) El registro de actualización del ítem de la base de datos, con su estado previo (before-image) y posterior (after-image)

2) Un **registro de compensación**, asociado con la reversión de la actualización, con su estado previo (before-image) y posterior (after-image)

Atomicidad y durabilidad en sistemas de actualización inmediata: DO-UNDO-REDO y sharp checkpoints

¿Qué otros problemas de desempeño pueden presentarse?

Aun deben realizarse las actualizaciones de los ítems de la BD (páginas) sobre memoria no-volátil antes de escribir el checkpoint.

Usar **fuzzy checkpoints**:

1) Se obtiene el identificador de las páginas que han sido actualizadas y que aun no van a memoria no-volátil.

2) Se obliga a escribir estas páginas a memoria no-volátil únicamente hasta cuando hay un nuevo checkpoint.

Atomicidad y durabilidad en sistemas de actualización inmediata: archivo del log

¿Cómo suplir el problema de que el log crezca indiscriminadamente?

Una porción del log puede ser movida a un archivo histórico.

Atomicidad y durabilidad en sistemas de actualización diferida

No se requieren registros UNDO, porque los ítems de la BD no serán actualizados hasta después que la transacción haga commit.

Se mantiene una lista de intenciones de transacciones (operaciones de actualizaciones sobre ítems de la BD).

En el momento del commit:

1) Se añade un registro de commit al log.
2) Debido a 1), se fuerza a escribir las actualizaciones que preceden el commit, en memoria no-volátil (disco).
3) Se escribe un **completion record** al log.

Fallas de los medios de almacenamiento (disco)

Varias estrategias:

Mantener una copia (mirror) de la base de datos completa.

Restaurar la BD desde el log → puede ser impráctico debido al tamaño del log.

Hacer una copia periódica de la BD (**dump**):

1) Poner la BD “offline”.
2) Hacer una copia de la BD.

Trabajo en grupo