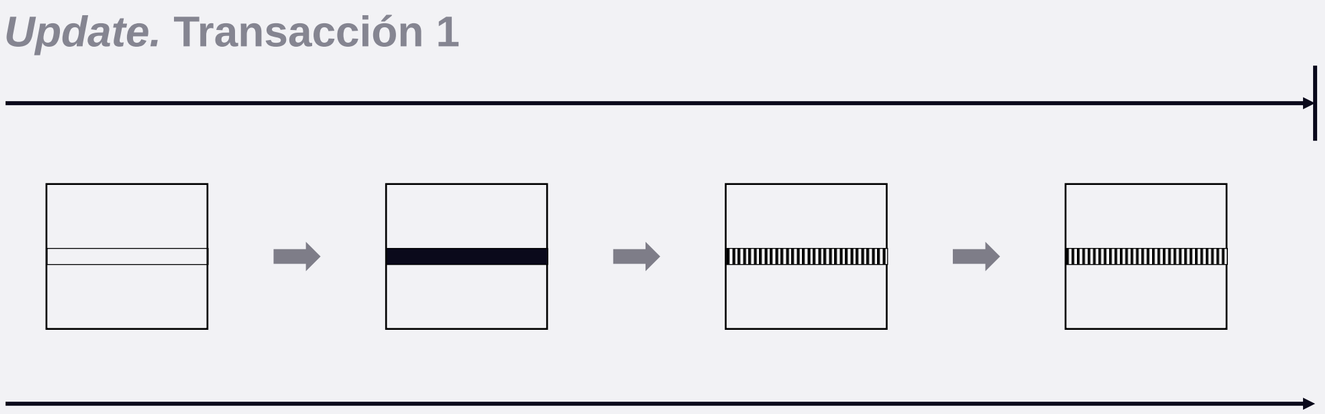
Concurrencia

Concurrencia

- Son procesos que garantizan la consistencia de la BD con el objetivo de mantener un alto número de transacciones activas, buscando un balance con la sobrecarga de controles necesarios para su implementación.
- Los problemas ocasionados por falta del control de concurrencia suceden cuando múltiples transacciones enviadas por varios usuarios interfieren entre sí de una manera que producen resultados incorrectos.
- Si no se hace un adecuado control de concurrencia, se pueden perder actualizaciones de datos provocando que los efectos de algunas transacciones no se reflejen en la base de datos. Por tal motivo, pueden ocurrir recuperaciones de datos inconsistentes.
- Algunas anomalías que pueden suceder se las conoce como:
    - Lost Update
    - Dirty Read (o The Temporary Update)
    - Unrepeatable Read
    - Phantom

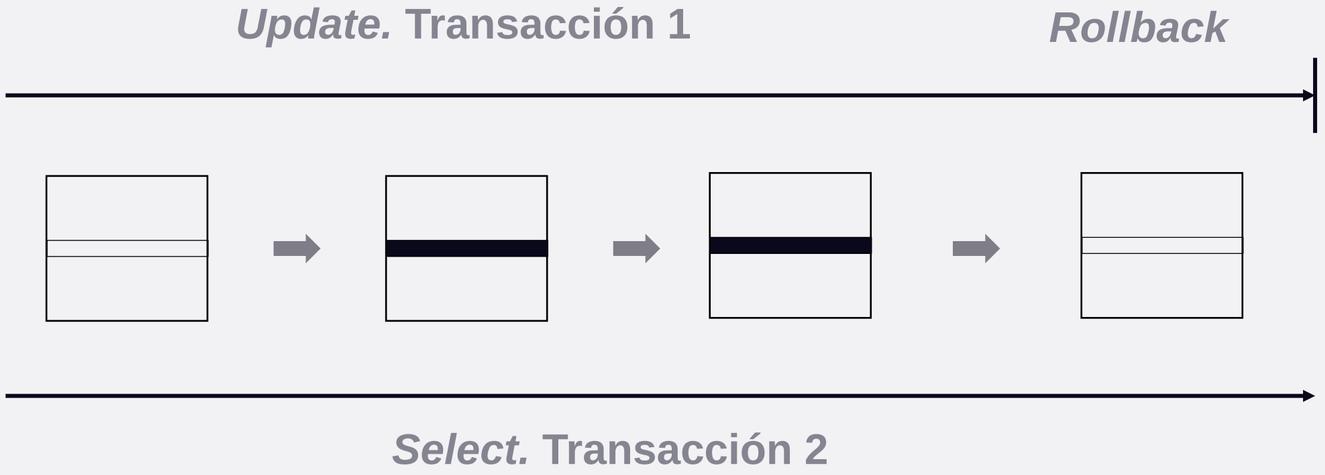
Concurrencia (cont.)

- ***Lost Update***
  - Ocurre cuando dos transacciones intentan modificar el mismo dato y una de las actualizaciones se pierde.
  - La T1 y la T2 leen la misma fila y calculan un nuevo valor.
  - Si la T1 actualiza la fila con el nuevo valor y la T2 actualiza la misma fila, se pierden los cambios efectuados por la primera.



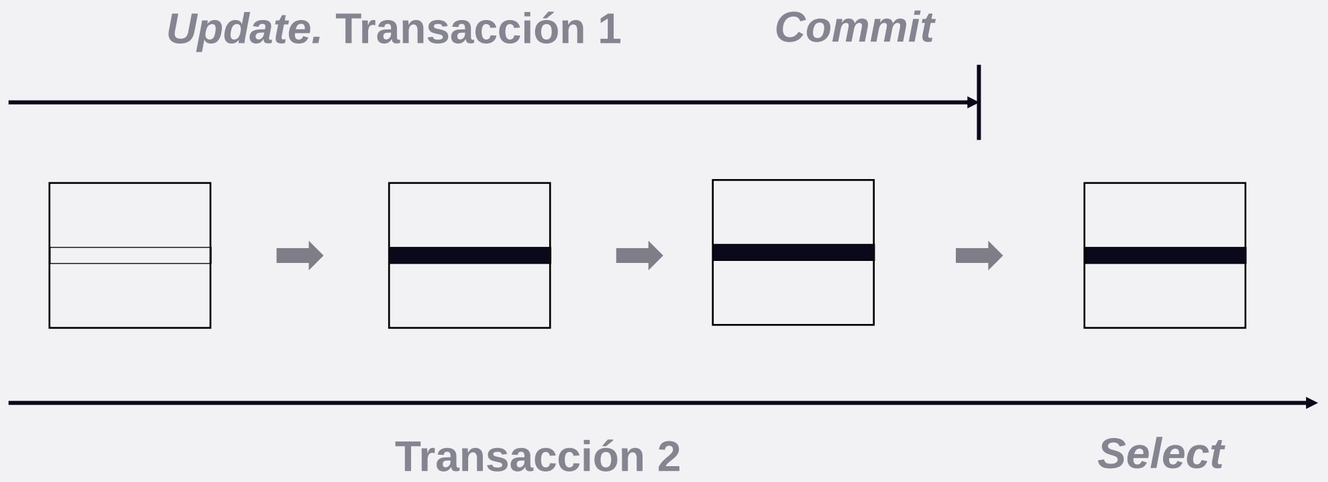
Concurrencia (cont.)

- ***Dirty Read***
  - Ocurre cuando una transacción lee datos no confirmados (no *commit*).
  - La T1 cambia una fila.
  - La T2 lee la fila antes que la T1 sea confirmada.
  - Si la T1 hace un *rollback*, la T2 leyó datos que nunca existieron.



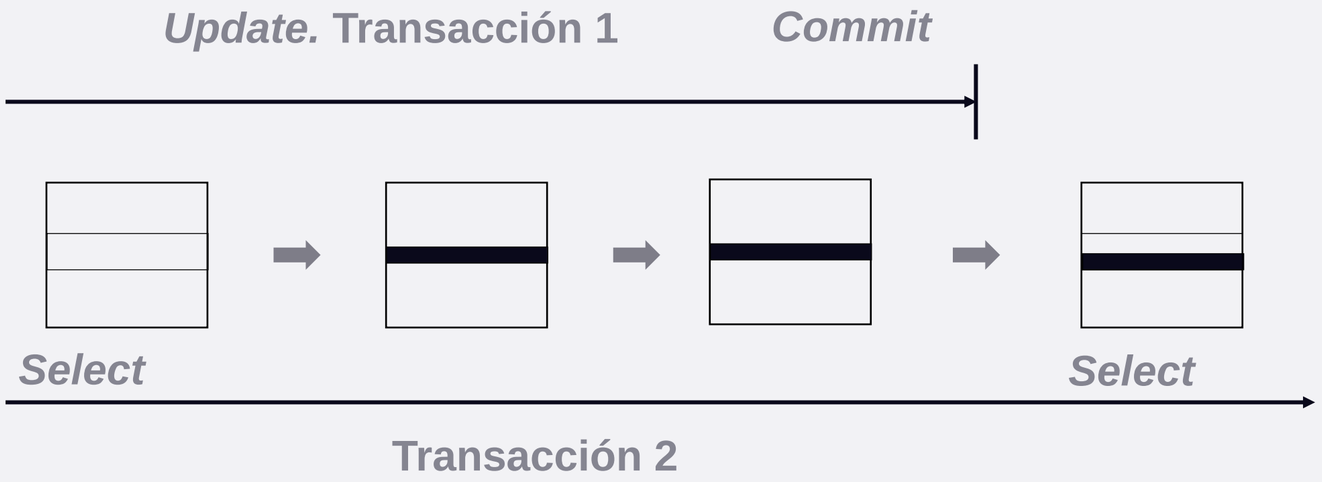
Concurrencia (cont.)

- ***Unrepeatable Read o Nonrepeatable Read***
  - Ocurre cuando una transacción lee la misma fila dos veces con valores distintos en ambas lecturas.
  - La T1 lee una fila.
  - La T2 cambia o borra la fila y confirma los cambios.
  - Cuando T1 relee la fila se encuentra con datos distintos o directamente no los encuentra.



Concurrencia (cont.)

- ***Phantom***
  - Ocurre cuando una fila que cumple un cierto criterio de búsqueda no es hallada en una primera lectura, pero sí en operaciones siguientes.
  - La T1 lee un conjunto de filas.
  - La T2 inserta una nueva fila que cumple con el criterio.
  - Al re-ejecutarse la consulta se obtiene un conjunto diferente de filas.



Concurrencia (cont.)

- **Serealización de transacciones:** Define el orden de ejecución de todas las operaciones en su dominio.
- **Conflicto de operaciones y serialización**
  - La función primaria de un controlador de concurrencia es generar un orden serializado para la ejecución de todas las transacciones.
  - El problema radica en desarrollar algoritmos que garanticen que únicamente se genere un orden serializado
  - Dos operaciones que acceden al mismo dato de la BD se dice que están en CONFLICTO si al menos una de ellas es de escritura.
  - Las operaciones de lectura no tienen conflicto consigo mismas
- Existen dos tipos de conflictos:
  - *Read-write (o write-read)*
  - *Write-write*

Concurrencia (cont.)

- **Mecanismos de control de concurrencia**
  - Algoritmos basados en LOCK con accesos mutuamente exclusivos a datos compartidos.
  - Algoritmos basados en protocolos, que ordenan la ejecución de las transacciones de acuerdo a un conjunto de reglas.
- Estos métodos se aplican según dos puntos de vista:
  - Pesimista: Sincronizan la ejecución concurrente de las transacciones en la etapa inicial del ciclo de ejecución.
  - Optimista: Retrasan la sincronización de las transacciones hasta su finalización

Mecanismos de control de concurrencia

Técnicas pesimistas:

- Algoritmos basados en candados

Existen candados de lectura (rl), de escritura (wl) y existe un administrador de candados (LM) que recibe del administrador de transacciones el elemento de dato y la operación a realizar.

El LM verifica la compatibilidad de los candados y si no hay conflicto fija un nuevo candado, de lo contrario retrasa la transacción.

Una vez terminada la transacción se eliminan todos los candados.

Las técnicas de bloqueo o candado pueden producir DEADLOCK, donde dos o más transacciones están esperando cada una de ellas que la otra libere algún objeto antes de seguir.

- Algoritmos basados en *slice* de tiempo

Seleccionan un orden de serialización a priori y ejecutan las transacciones de acuerdo a ellas.

Para establecer el orden, el administrador de transacciones le asigna a cada transacción Ti un *slice* de tiempo único TS (Ti).

Un *slice* de tiempo es un identificador simple que sirve para identificar cada transacción de manera única.

Mecanismos de control de concurrencia

Transacción pesimista:

Concurrencia (cont.)

Técnicas optimistas

- Las transacciones acceden libremente a los elementos y antes de finalizar se determina si ha habido interferencias.
- Las operaciones de lectura, computo y escritura de cada transacción se procesan libremente sin actualizar la BD de corriente.
- Cada transacción inicialmente hace sus cambios en copias locales de los datos.
- La fase de validación consiste en verificar si esas actualizaciones conservan la consistencia de la BD, si la respuesta es positiva, los cambios se hacen globales (escritos en la BD corriente). De otra manera, la transacción es abortada y tiene que reiniciarse.

Concurrencia (cont.)

Técnicas optimistas

- Este tipo de técnicas considera que las transacciones tienen 3 fases:
  - **Lectura**: Las transacciones realizan operaciones sobre copias privadas de los objetos (accesibles solo para la transacción).
  - **Validación**: Se comprueba si el conjunto de objetos modificados por una transacción se solapa con el conjunto de objetos modificados por alguna otra transacción que haya hecho la validación durante la fase de lectura de dicha transacción.
  - **Grabación**: en el caso de no detectar interferencias se graban las modificaciones, convirtiendo las versiones privadas de los objetos en versiones actuales
- En todo sistema de concurrencia se tiene que tener cuidado con los *deadlocks.*

Concurrencia (cont.)

Transacción optimista:

Integridad

Integridad

- Tiene como función proteger la BD contra operaciones que introduzcan inconsistencia en los datos, en el sentido de corrección, validez o precisión de los datos.
- El subsistema de **integridad** de un SGBD **debe,** por tanto, detectar y corregir, en la medida de lo posible, las operaciones incorrectas.
- En la práctica casi toda la verificación de integridad se realiza mediante código de procedimientos escritos por los usuarios.

Integridad (cont.)

- **Operaciones semánticamente inconsistentes:**
  - Transgreden las restricciones que ha definido el administrador al diseñar la BD:
  - Restricciones estáticas:
    - Restricciones sobre los dominios, por ejemplo: el dominio edad debe estar comprendido entre 18 y 65 años.
    - Restricciones sobre atributos, por ejemplo: la edad de los empleados ingenieros debe ser mayor a 21 años.
  - Restricciones dinámicas:
    - El sueldo de un empleado no puede disminuir.

Integridad (cont.)

Otra forma de clasificar las restricciones

- **Simples**: Si se aplican a una ocurrencia de un atributo con independencia de los demás, por ejemplo: el sueldo de un empleado tiene que ser mayor que 6000.
- **Compuestas**: Si implican más de una ocurrencia, como es el caso de las restricciones de comparación, por ejemplo: el sueldo de un empleado debe ser menor que el de su jefe.
- **De globalidad**, por ejemplo el sueldo medio de los empleados de un determinado departamento debe ser menor de 25000.

Las reglas de **integridad** deben almacenarse en el diccionario de datos, como parte integrante de los datos (control centralizado de la semántica), de modo que no han de incluirse en los Programas.