4.6.- Integridad: Control de concurrencia.

4.6.1.- Introducción

4.6.2.- Técnicas de Bloqueo.

4.6.2.1.- Bloqueo (variable cerrojo)

- Tipos, protocolos
- Problemas. Interbloqueo
- Granularidad

4.6.2.2.- Marcas de Tiempo

- Protocolos
- Marcas de Tiempo Multiversion

4.6.- Integridad: Control de concurrencia.
4.6.1.- Introducción (1).

- En sistema multiusuario es imprescindible, un mecanismo de control de concurrencia para conservar la integridad de la BD.
  - Todos los datos deben ser iguales para todos los usuarios.

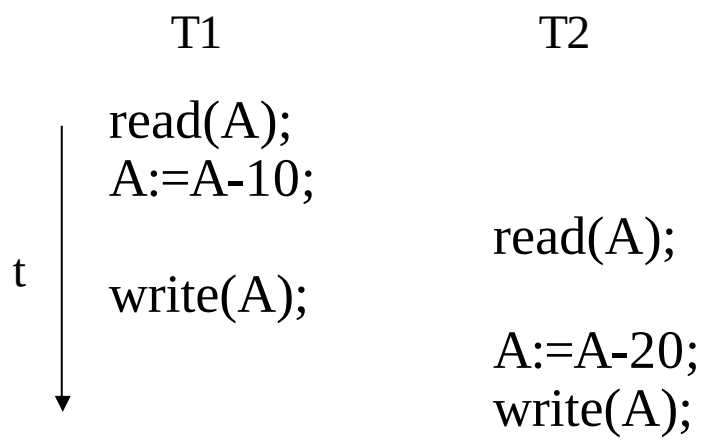
- Cuando se ejecutan varias transacciones simultáneamente pueden producirse estados inconsistentes en la BD:
  - Una transacción bancaria lee un importe y le resta 100 euros y antes de actualizar la BBDD otra transacción lee ese dato.

- **ORACLE** es una BD multiusuario. Se necesita
  - Concurrencia
  - Consistencia

Maximización de concurrencia → maximiza productividad y desarrollo

4.6.- Integridad: Control de concurrencia.
4.6.1.- Introducción (2).

- Ejemplo de “problema” de concurrencia:



Estas dos transacciones simultaneas NO pueden acceder al mismo dato.

OBJETIVO para controlar la concurrencia:

Garantizar que las transacciones sean seriables

así, se garantizará la consistencia de las transacciones.

4.6.- Integridad: Control de concurrencia.
4.6.1.- Introducción (3).

- Problemas clásicos de concurrencia:
  - Modificación perdida
  - Modificación temporal
  - Totalización incorrecta
  - Lectura no repetible
- En **ORACLE** los fenómenos prevenibles son:
  - Lectura sucia
  - Lectura no repetible (borrosa)
  - Lectura fantasma

4.6.- Integridad: Control de concurrencia.
4.6.1.- Introducción (4).

Para prevenir problemas → Técnicas de Control de Concurrencia

–Técnicas Pesimistas (Prevención)
- •Bloqueos, Marcas de Tiempo,..

–Técnicas Optimistas (Corrección)
- •Técnicas de Validación

ORACLE tiene como soluciones:

- Varios tipos de **bloqueo**
- Un **modelo** de consistencia **multiversión**.
- Niveles de **AISLAMIENTO**:
  - Aceptación de lectura (read commited)
  - Serializable
  - Modo de solo lectura (read-only mode)

4.6.- Integridad: Control de concurrencia.
4.6.1.- Introducción (5).

- **ORACLE** para prevenir interacción destructiva de datos entre usuarios:
  - Consistencia: Asegura que los datos que estamos viendo no cambian (por otros usuarios) hasta que acabemos la transacción
  - Integridad: Asegura que los datos y estructuras reflejan los cambios en una secuencia correcta.

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (1)

- Un bloqueo, asocia una variable (cerrojo) a cada elemento de datos que describe el estado de dicho elemento, respecto a las posibles operaciones que sobre él se puede realizar.
  - Identificador del Elemento bloqueado
  - Identificador de la Transacción que lo bloquea

- Una transacción obtiene un bloqueo solicitándolo al Gestor de Bloqueos.
- Un bloqueo es una garantía de ciertos derechos de exclusividad para la Transacción.

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (2).

- **Tipos**:
  - Bloqueos exclusivos: Un único bloqueo por recurso
  - Bloqueos Compartidos: Muchos bloqueos por recurso

- **ORACLE** tiene los dos tipos de bloqueo así como control de consistencia multiversión para asegurar acceso concurrente a los datos.

- Un **Protocolo de Bloqueo** indica cuando una transacción puede bloquear y desbloquear elementos. Si los bloqueos se realizaran de manera arbitraria se podrían producir inconsistencias.

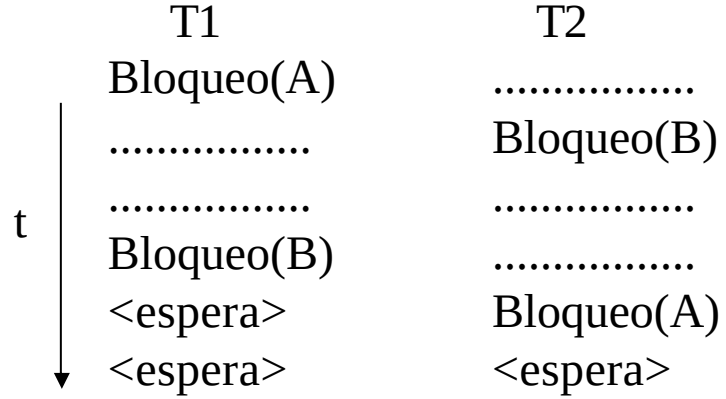
- Protocolo de Bloqueo en dos Fases (Two-Phase-Locking)
  - Fase de Crecimiento: Bloquea pero no Libera
  - Fase de Encogimiento: Libera pero no Bloquea

Asegura la seriabilidad

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (3).

- Problema de Interbloqueo. DEADLOCK (1)

✓ Dos o más transacciones se quedan a la espera de que se liberen elementos que tiene bloqueados otra.



✓ Puede producirse incluso con protocolos de bloqueo que garantizan la seriabilidad

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (4).

- Problema de Interbloqueo. DEADLOCK (2)
- Soluciones:
  - **Prevención del interbloqueo**, obligando a que las transacciones bloqueen todos los elementos que necesitan por adelantado
  - **Detectar el interbloqueo**, controlando de forma periódica si se ha producido, para lo que suele construir un grafo de espera:
    - Cada nodo representa una transacción
    - Existe un arco entre Ti y Tj si Ti está esperando un recurso que tiene bloqueado Tj
    - Existe interbloqueo si el Grafo tiene un ciclo.

Cada SGBD tiene su propia política para escoger las victimas, aunque suelen ser las transacciones mas recientes.

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (5).

- Problema de Interbloqueo. DEADLOCK (3)
  - **Prevención el interbloqueo**:
    - Bloquear todo lo necesario por adelantado
    - Comprobar por adelantado la posibilidad de Interbloqueo, la transacción esperará en el caso en que de atenderse su petición se produciría Interbloqueo.
    - Nuevos protocolos de bloqueo que además de garantizar la seriabilidad, eviten el interbloqueo. (ej. Protocolo del árbol).
    - Utilización de Marcas de Tiempo para cada transacción MT(Ti):
      - Usar expropiación y retroceso de Transacciones
        - esperar-morir (Wait – Die)
        - herir-esperar (Wound – Wait)

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (6).

- Problema de Interbloqueo. DEADLOCK (4)

  - **Detección del Interbloqueo y Recuperación**
    - Periódicamente se comprueba la existencia de Interbloqueo:
      - Si se detecta Interbloqueo, se elige una transacción como víctima y se la mata.
      - Se puede producir inanición si se elige siempre a la misma víctima.

¿Cuándo usar prevención y cuando detección?

4.6.2.- Técnicas de Bloqueo.
4.6.2.1.- Bloqueo. Variable cerrojo (7).

- GRANULARIDAD del Bloqueo: para mejorar rendimiento
  - Son posibles diversos niveles de bloqueo
    - un campo de un registro - un registro
    - un fichero - la base de datos
  - Granularidad gruesa →menor gestión de bloqueos y menor nivel de concurrencia.
  - Granularidad fina → mayor gestión de bloqueos y mayor nivel de concurrencia.
  - **ORACLE** bloquea automáticamente:
    - **Objetos de usuario** (tablas, filas, etc. (estructuras y datos))
    - **Objetos del sistema** → invisibles a los usuarios (estructuras de datos compartidas en memoria, filas de diccionario de datos, etc.)

4.6.2.- Técnicas de Bloqueo.
4.6.2.2.- Marcas de Tiempo (1).

- En lugar de determinar el orden entre las transacciones en conflicto en función del momento del acceso a los elementos, determinan por adelantado una ordenación de las transacciones.
- El interbloqueo es imposible.
- Una marca de tiempo es un identificador único asociado a cada transacción
- Las actualizaciones físicas se retrasan hasta la confirmación de las transacciones. No se puede actualizar ningún elemento actualizado por otra transacción más reciente.

4.6.2.- Técnicas de Bloqueo.
4.6.2.2.- Marcas de Tiempo (2).

- **Protocolos:**
  - **Wait-die** que fuerza a una transacción a esperar en caso de que entre en conflicto con otra transacción cuya marca de tiempo sea mas reciente, o a morir (abortar y reiniciar) si la transacción que se esta ejecutando es más antigua.
  - **Wound-wait**, que permite a una transacción matar a otra que posea una marca de tiempo más reciente, o que fuerza a la transacción peticionaria a esperar.

4.6.2.- Técnicas de Bloqueo.
4.6.2.2.- Marcas de Tiempo (3).

- **Marcas de tiempo multiversión (1)**
  - Varias transacciones leen y escriben diferentes versiones del mismo dato siempre que cada transacción sea un conjunto consistente de versiones de todos los datos a los que accede.
  - **ORACLE** utiliza un sistema multiversión
  - **Protocolo**:
    - Esta basado en marcas de tiempo
    - El control de concurrencia es de varias versiones a la vez de un item de datos.
    - Cuando una transacción requiere acceder a un item, la marca de tiempo de la transacción es comparada con las marcas de tiempo de las diferentes versiones del item.
    - Se elige una versión de los items para mantener la serializabilidad del plan de items que se este ejecutando.

4.6.2.- Técnicas de Bloqueo.
4.6.2.2.- Marcas de Tiempo (4).

- **Modelo Multiversión de ORACLE (1):**
  - Consistencia en lectura:
    - Imaginar cada usuario operando con una copia privada de la BD (modelo consistente multiversión)
    - Características
      - garantiza que los datos no cambian durante la ejecución
      - Los lectores no esperan a los que escriben o a otros lectores
      - Los que escriben no esperan a los lectores, pero sí a otros escritores de los mismos datos
    - Niveles de concurrencia
      - Sentencia
      - Transacción

4.6.2.- Técnicas de Bloqueo.
4.6.2.2.- Marcas de Tiempo (5).

- **Modelo Multiversión de ORACLE (2):**
  - Cuando se lee y modifica al mismo tiempo, Oracle crea:
    - Conjunto de datos (vistas) consistente en lectura
    - Cuando se modifica (antes del COMMIT)
      - se almacenan los valores antiguos de los datos en *Segmentos de Rollback*
      - Crea la Vista Consistente a partir de:
        - Información actual en el *Área Global del Sistema*
        - Información antigua en los *Segmentos de Rollback*
    - Al hacer el COMMIT:
      - Se hacen los cambios permanentes para todas las vistas posteriores

Bibliografía

- Fundamentos de Sistemas de Bases de Datos", 3ª Edición, Elmasri y Navathe, Addison Wesley, 2002. Capitulo 20.
- Database systems : a practical approach to design, implementation and management. Thomas M. Connolly. Addison-Wesley. 2005.
- Sistemas de bases de datos : un enfoque práctico para diseño, implementación y gestión. Thomas M. Connolly. Addison-Wesley. 2005.
- Documentación de Oracle. http://otn.oracle.com