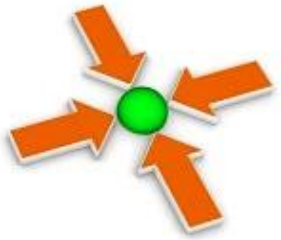

# Sistemas de Bases de Datos 2

**2024**

**Ing. Luis Alberto Arias Solórzano**

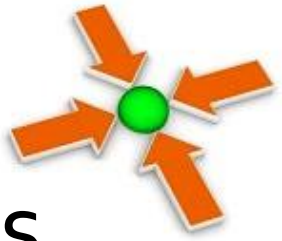
**Unidad 1**



# Concurrencia

- Se refiere al hecho de que los DBMSs (Sistemas de Administración de Bases de Datos) permiten que muchas transacciones accedan a una misma base de datos a la vez.

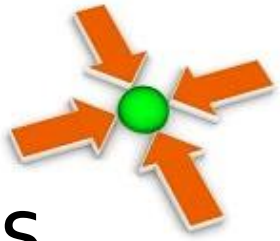
- Es la secuencia de interacciones o comunicaciones entre los procesos y el acceso coordinado de recursos que se comparten por todos los procesos o tareas.



# Concurrencia: Problemas

- Existen esencialmente tres maneras en las que las cosas pueden salir mal; esto es, tres formas en las que una transacción, aunque sea correcta por sí misma, puede producir una respuesta incorrecta si alguna otra transacción interfiere con ella en alguna forma.

  - **El problema de la *actualización perdida*,**
  - **El problema de la dependencia no confirmada y**
  - **El problema del análisis inconsistente.**



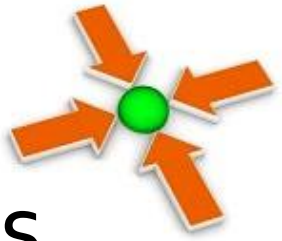
# Concurrencia: Problemas

- **El problema de la *actualización perdida***

Ejemplo:

La transacción A recupera alguna tupla t en el tiempo t1; la transacción B recupera la misma tupia t en el tiempo t2; la transacción A actualiza la tupia en el tiempo t3 (con base en los valores vistos en el tiempo t1), y la transacción B actualiza la misma tupia en el tiempo t4 (con base en los valores vistos en el tiempo t2, que son los mismos vistos en el tiempo t1).

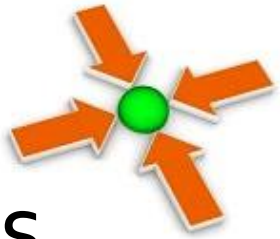
La actualización de la transacción A se pierde en el tiempo t4, ya que la transacción B la sobrescribe sin siquiera mirarla.



# Concurrencia: Problemas

- **El problema de la dependencia no confirmada**

Se presenta al permitir que una transacción recupere, o lo que es peor, actualice una tupia que ha sido actualizada por otra transacción pero que aún no ha sido confirmada por esa misma transacción. Puesto que no ha sido confirmada, sigue existiendo la posibilidad de que nunca lo sea y de que en su lugar se deshaga mediante un ROLLBACK; en cuyo caso, la primera transacción habrá visto datos que ya no existen (y en cierto sentido nunca existieron).



# Concurrencia: Problemas

- **El problema del análisis inconsistente.**

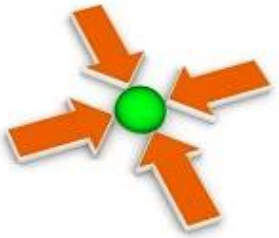
Ejemplo:

Dos transacciones, A y B, operando sobre tuplas de una cuenta (ACC):

La transacción A está sumando saldos de cuenta y la transacción B está transfiriendo una cantidad X de la cuenta 3 a la cuenta 1.

El resultado de A termina incorrecto; y si A continuara y escribiera ese resultado en la base de datos, dejaría en efecto a la base de datos en un estado inconsistente. Decimos que A ha visto un estado inconsistente de la base de datos y por lo tanto, ha realizado un análisis inconsistente.

Observe la diferencia entre este ejemplo y el anterior: aquí no hay posibilidad de que A sea dependiente de un cambio no confirmado, ya que B confirma todas sus actualizaciones antes de que A vea a ACC3.



# Concurrencia: Control

- Cuando se ejecutan varias transacciones concurrentemente en la base de datos, puede que deje de conservarse la propiedad de aislamiento. Es necesario que el sistema controle la interacción entre las transacciones concurrentes; dicho control se lleva a cabo a través de uno de los muchos mecanismos existentes llamado esquemas de *control de concurrencia*.

# Bloqueos

- Una forma de asegurar la secuencialidad es exigir que el acceso a los elementos de datos se haga en exclusión mutua; es decir, mientras una transacción accede a un elemento de datos, ninguna otra transacción puede modificar dicho elemento. El método más habitual que se usa para implementar este requisito es permitir que una transacción acceda a un elemento de datos sólo si posee actualmente un bloqueo sobre dicho elemento.

- La idea básica es simple: Cuando una transacción deba asegurarse de que algún objeto en el que está interesada no cambiará de ninguna forma mientras lo esté usando, **adquiere un bloqueo** sobre ese objeto. El efecto del bloqueo es "inhibir todas las demás transacciones" en ese objeto y por lo tanto, impedir que lo cambien. Por lo tanto, la primera transacción es capaz de realizar todo su procesamiento con el conocimiento certero de que el objeto en cuestión permanecerá en un estado estable durante todo el tiempo que ésta lo desee.

# Bloqueos

- Primero suponemos que el sistema soporta dos tipos de bloqueos, bloqueos **compartidos** (S) y bloqueos **exclusivos** (X). En ocasiones, a los bloqueos S y X se les llama bloqueos **de lectura y de escritura ,** respectiva mente.

- **1. Compartido.** Si una transacción *Ti* obtiene un **bloqueo en modo compartido** (denotado por S) sobre el elemento *Q*, entonces *Ti* puede leer *Q* pero no lo puede escribir.

- **2. Exclusivo.** Si una transacción *Ti* obtiene un **bloqueo en modo exclusivo** (denotado por X) sobre el elemento *Q*, entonces *Ti* puede tanto leer como escribir *Q*.

| | X | S | - |
|---|---|---|---|
| X | No | No | Sí |
| S | No | Sí | Sí |
| - | Sí | Sí | Sí |

# Bloqueos

- Es necesario que toda transacción **solicite** un bloqueo del modo apropiado sobre el elemento de datos *Q* dependiendo de los tipos de operaciones que se vayan a realizar sobre *Q*. La petición se hace al gestor de control de concurrencia. La transacción puede realizar la operación sólo después de que el gestor de control de concurrencia **conceda** el bloqueo a la transacción.

| | X | S | - |
|---|---|---|---|
| X | No | No | Sí |
| S | No | Sí | Sí |
| - | Sí | Sí | Sí |

# DEADLOCK

El bloqueo puede introducir problemas por sí solo, principalmente el problema del deadlock:

- El deadlock es una situación en la que dos o más transacciones se encuentran en estados simultáneos de espera, cada una de ellas esperando que alguna de las demás libere un bloqueo para poder continuar.
- Un bloqueo mortal que involucra a dos transacciones, pero también son posibles, al menos en principio, bloqueos mortales que involucren a tres, cuatro, o más transacciones.
- Si ocurre un bloqueo mortal es preferible que el sistema lo detecte y lo rompa.

# DEADLOCK

La detección del deadlock implica la detección de un ciclo en el grafo de espera; es decir, el grafo de "quien está esperando a quien". La ruptura del bloqueo mortal implica seleccionar una de las transacciones bloqueadas mortalmente es decir, una de las transacciones dentro del ciclo del grafo como victima y entonces deshacerla liberando por lo tanto sus bloqueos y permitiendo que continúen las demás transacciones.

Observe que la victima ha "fallado" y ha sido deshecha sin ser culpable. Algunos sistemas volverán a iniciar esta transacción desde el principio, bajo la suposición de que las condiciones que causaron el bloqueo mortal probablemente ya no existirán. Otros sistemas simplemente regresan un código de excepción y es asunto del programa manejar la situación de forma adecuada.

# INTERBLOQUEO

Un sistema está en estado de interbloqueo si existe un conjunto de transacciones tal que toda transacción del conjunto está esperando a otra transacción del conjunto.

Con mayor precisión, existe un conjunto de transacciones en espera {T0, T1, …, Tn} tal que T0 está esperando a un elemento de datos que posee T1, y T2 está esperando a un elemento de datos que posee T2, y …, y Tn–1 está esperando a un elemento de datos que posee Tn' y Tn está esperando a un elemento que posee T0. En tal situación ninguna de las transacciones puede progresar.

El único remedio a esta situación no deseada es que el sistema invoque alguna acción drástica, como retroceder alguna de las transacciones involucradas en el interbloqueo. El retroceso de una transacción puede ser parcial: esto es, se puede retroceder una transacción hasta el punto donde obtuvo un bloqueo cuya liberación resuelve el interbloqueo.

# INTERBLOQUEO

Existen dos métodos principales para tratar el problema de los interbloqueos.:

1. Se puede utilizar un protocolo de prevención de interbloqueos para asegurar que el sistema nunca llega a un estado de interbloqueo.

2. De forma alternativa se puede permitir que el sistema llegue a un estado de interbloqueo, y tratar de recuperarse después a través de un esquema de detección y recuperación de interbloqueos.

Ambos pueden provocar retroceso de las transacciones. La prevención se usa normalmente cuando la probabilidad de que el sistema llegue a un estado de interbloqueo es relativamente alta; en otro caso es más eficiente usar la detección y recuperación

# SERIABILIDAD

La seriabilidad es el criterio de corrección aceptado comúnmente para la ejecución de un conjunto dado de transacciones. Para ser más precisos, se considera que la ejecución de un conjunto dado de transacciones es correcta cuando es seriable; es decir, cuando produce el mismo resultado que una ejecución serial de las mismas transacciones, ejecutando una a la vez. Esta es la justificación de esta aseveración:

Las transacciones individuales son tomadas como correctas; es decir, se da por hecho que transforman un estado correcto de la base de datos en otro estado correcto. Por lo tanto también es correcta la ejecución de una transacción a la vez en cualquier orden serial, y se dice "cualquier" orden serial debido a que las transacciones individuales son consideradas independientes entre sí.Por lo tanto, una ejecución intercalada es correcta cuando equivale a alguna ejecución serial; es decir, cuando es seriable.

# Niveles de Aislamiento

Utilizamos el termino **nivel de aislamiento** para referirnos a lo que podría ser descrito vagamente como el *grado de interferencia* que una transacción dada es capaz de tolerar por parte de las transacciones concurrentes.

Entonces, si queremos garantizar la seriabilidad, no podemos aceptar ninguna cantidad de interferencia; en otras palabras, el nivel de aislamiento debe ser el máximo posible. Sin embargo, los sistemas reales comúnmente permiten que las transacciones operen a niveles de aislamiento que son menores a este máximo, por una diversidad de razones pragmáticas

Podemos definir al menos, cinco niveles de aislamiento. En términos generales, entre mayor sea el nivel de aislamiento, menor será la interferencia (y menor la concurrencia); y entre más bajo sea el nivel de aislamiento, mayor será la interferencia (y mayor la concurrencia).

# TAREA

Tarea: Investigar los protocolos de bloqueo

Gracias