Vázquez Zavala Oliver Alexis
Tarea 23

Concurrencia en SO y BD

- Concurrencia Sistemas Operativos

La concurrencia comprende un gran número de cuestiones de diseño, incluyendo la comunicación entre procesos, comparación y competencia por los recursos, sincronización de la ejecución de varios procesos y asignación del tiempo de procesador a los procesos y es fundamental para que existan diseños como Multiprogramación, Multiproceso y Proceso distribuido.

Los procesos son concurrentes si existen simultáneamente, los procesos concurrentes pueden funcionar en forma totalmente independiente unos de otros, o pueden ser asíncronos, lo cual significa que en ocasiones requieren cierta sincronización o cooperación.

La concurrencia puede presentarse en tres contextos diferentes:

- Varias aplicaciones: La multiprogramación se creó para permitir que el tiempo de procesador de la máquina fuese compartido dinámicamente entre varios trabajos o aplicaciones activas.

- Aplicaciones estructuradas: Como ampliación de los principios del diseño modular y la programación estructurada, algunas aplicaciones pueden implementarse eficazmente como un conjunto de procesos concurrentes.

- Estructura del sistema operativo: Las mismas ventajas de estructuración son aplicables a los programadores de sistemas y se ha comprobado que algunos sistemas operativos están implementados como un conjunto de procesos.

Los procesos que ejecutan de forma concurrente en un sistema se pueden clasificar como:

- Proceso independiente: Es aquel que ejecuta sin requerir la ayuda o cooperación de otros procesos. Un claro ejemplo de procesos independientes son los diferentes shells que se ejecutan de forma simultánea en un sistema.
- Procesos son cooperantes: Son aquellos que están diseñados para trabajar conjuntamente en alguna actividad, para lo que deben ser capaces de comunicarse e interactuar entre ellos.

En ambos tipos de procesos (independientes y cooperantes), puede producirse una serie de interacciones entre ellos y pueden ser de dos tipos:

1.- Interacciones motivadas porque los procesos comparten o compiten por el acceso a recursos físicos o lógicos. Por ejemplo, dos procesos independientes compiten por el acceso a disco o para modificar una base de datos.

2.- Interacción motivada porque los procesos se comunican y sincronizan entre sí para alcanzar un objetivo común, Por ejemplo, un compilador que tiene varios procesos que trabajan conjuntamente para obtener un solo archivo de salida.

Beneficios del uso de la concurrencia:

- Trata de evitar los tiempos muertos de la UCP
- Comparte y optimiza el uso de recursos
- Permite el modularidad en las diferentes etapas del proceso
- Acelera los cálculos
- Da mayor comodidad

Desventajas de la concurrencia:

- Inanición e interrupción de procesos
- Ocurrencia de bloqueos
- Que dos o más procesos requieran el mismo recurso (no apropiativo).

Concurrencia en Bases de Datos

La concurrencia de bases de datos es la capacidad de una base de datos para permitir que varios usuarios afecten a varias transacciones, esta es una de las principales propiedades que separa una base de datos de otras formas de almacenamiento de datos, como las hojas de cálculo. La capacidad de ofrecer simultaneidad es exclusiva de las bases de datos, las hojas de cálculo u otros medios de almacenamiento de archivos planos a menudo se comparan con las bases de datos, pero difieren en este aspecto importante ya que no pueden ofrecer a varios usuarios la capacidad de ver y trabajar con los diferentes datos en el mismo archivo, porque una vez que el primer usuario abre el archivo, está bloqueado para otros usuarios, y otros usuarios pueden leer el archivo, pero no pueden editar los datos.

Los problemas causados por la concurrencia de la base de datos son incluso más importantes que la capacidad de admitir transacciones concurrentes, por ejemplo, cuando un usuario está cambiando datos, pero aún no ha guardado (confirmado) esos datos, entonces la base de datos no debe permitir que otros usuarios que consultan los mismos datos vean los datos cambiados y no guardados, en cambio, el usuario solo debe ver los datos originales.

Casi todas las bases de datos tratan la concurrencia de la misma manera, el principio general es que los datos modificados, pero no guardados se guardan en algún tipo de registro o archivo temporal y una vez que se guarda, se escribe en el almacenamiento físico de la base de datos en lugar de los datos originales, siempre que el usuario que realiza el cambio no haya guardado los datos, solo él debería poder ver los datos que está cambiando y todos los demás usuarios que soliciten los mismos datos deben ver los datos que existían antes del

cambio, una vez que el usuario guarda los datos, las nuevas consultas deberían revelar el nuevo valor de los datos.

El control de la concurrencia es la gestión de la contención de los recursos de datos, un esquema de control se considera pesimista cuando bloquea un recurso determinado al principio de la transacción de acceso a datos y no lo libera hasta que se cierra la transacción, un esquema de control de simultaneidad se considera optimista cuando se adquieren y liberan bloqueos durante un breve periodo de tiempo al final de una transacción.

El objetivo de la simultaneidad optimista es minimizar el tiempo que un recurso determinado no está disponible para que lo utilicen otras transacciones, esto es especialmente importante con las transacciones de larga duración, que bajo un esquema pesimista bloquearían un recurso para períodos de tiempo inaceptablemente largos.

Por otro lado, bajo un esquema optimista, los bloqueos se obtienen inmediatamente antes de una operación de lectura y se liberan inmediatamente después, los bloqueos de actualización se obtienen inmediatamente antes de una operación de actualización y se mantienen hasta el final de la transacción, parta tener simultaneidad optimista, este producto utiliza un esquema de actualización sobrecalificado para probar si otra transacción ha actualizado el origen de datos subyacente desde el principio de la transacción actual, con este esquema, las columnas marcadas para la actualización y sus valores originales se añaden explícitamente a través de una cláusula WHERE en la sentencia UPDATE para que la sentencia falle si se han cambiado los valores de columna subyacentes, como resultado, este esquema puede proporcionar control de simultaneidad a nivel de columna; los esquemas pesimistas sólo pueden controlar la simultaneidad a nivel de fila.

Los esquemas optimistas normalmente realizan este tipo de prueba sólo al final de una transacción, si las columnas subyacentes no se han actualizado desde el principio de la transacción, se confirman las actualizaciones pendientes de los campos de persistencia gestionada por contenedor y se liberan los bloqueos, si no se pueden adquirir bloqueos o si alguna otra transacción ha actualizado las columnas desde el principio de la transacción actual, la transacción se retrotrae: se pierde todo el trabajo realizado dentro de la transacción y es por esto que los esquemas de concurrencia pesimista y optimista requieren diferentes niveles de aislamiento en las transacciones respectivas.

Referencias:

1.- Socrates, “Concurrencia de la base de datos,” Techinfo.wiki, 02-Feb-2021. [Online]. Available: https://techinfo.wiki/concurrencia-de-la-base-de-datos/. [Accessed: 05-May-2022].

2.- G. Blokdyk, IBM docs: Complete self-assessment guide. North Charleston, SC: Createspace Independent Publishing Platform, 2018.

3.- Edu.ar. [Online]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/73561/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y. [Accessed: 06-May-2022].