Campus Querétaro

Lab 25: Manipulación de datos usando Transacciones

Marco Iván Flores Villanueva
A01276586

Materia: Construcción de software y toma de decisiones
Grupo: 501

Deadlock

En el ámbito de las bases de datos, un *deadlock* es una situación en la que dos o más transacciones están esperando a que la otra libere sus *locks*. Si consideramos la transacción A que ha puesto un *lock* en algunas filas de la tabla X, y necesita actualizar algunas filas de la tabla Y para terminar, y también consideramos la transacción B que ha puesto un *lock* en esas mismas filas de la tabla Y, y necesita actualizar las mismas filas de la tabla X que la transacción A ha bloqueado, entonces nos encontramos con un *deadlock*, pues ninguna de las dos transacciones puede terminar al haberse bloqueado mutuamente. Por consiguiente, toda actividad se detiene para siempre a menos que el DBMS detecte este *deadlock* y aborte alguna de las transacciones.

Los *deadlock* pueden ocurrir en los ambientes multiusuario cuando dos o más transacciones se ejecutan concurrentemente e intentan acceder a los mismos recursos en un orden diferente. Cuando esto ocurre, una transacción puede bloquear un recurso que otra transacción necesita, mientras que la segunda puede bloquear un recurso que la primera necesita. Ambas transacciones se bloquean y ningún proceso se finaliza. Sin embargo, las DBMSs a menudo utilizan técnicas para automáticamente detectar y evitar *deadlocks*.

Entre estas técnicas encontramos mecanismos de *timeout*, los cuales fuerzan a que alguna transacción libere su *lock* después de un cierto tiempo, y algoritmos de detección de *deadlocks*, los cuales periódicamente escanean el registro de transacciones buscando ciclos *deadlock*, para después elegir una transacción a abortar y resolver el bloqueo.

También es posible prevenir estos *deadlocks* diseñando de buena manera las transacciones en primera instancia. Una práctica positiva de diseño de transacciones es hacer que siempre adquieran *locks* en el mismo orden o que liberen los mismos tan pronto sea posible. Un buen diseño del esquema de la base de datos y la aplicación también ayudarán a minimizar el riesgo de que un bloqueo de este tipo ocurra.

Referencias

GeeksforGeeks. (2024, 28 diciembre). *Deadlock in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/deadlock-in-dbms/

Oracle. (s. f.). *Deadlocks*. Deadlocks. https://docs.oracle.com/javadb/10.4.1.3/devguide/cdevconcepts28436.html