Lab 11 Bases de datos
Andrés Coutiño

EJERCICIO 1

Adjunto a esta entrega.

Datos.sql , menú lab 11.py

Link al GIT:
https://github.com/coutino568/Lab11BD

EJERCICIO 2

Para cada una de las funciones del ejercicio anterior discuta qué posibles problemas de atomicidad (si los hubiera) que podrían ocurrir si el sistema fuera interrumpido durante cualquier punto de la ejecución de la función. Discuta además qué posibles problemas de concurrencia podían darse si una invocación de la función ocurre al mismo tiempo que otra.

Su respuesta debe incluir el análisis para cada una de las 4 funciones.

Función 1:

Se espera que si a un registro le están modificando la velocidad o ram , los cambios sean persistentes, si falla una transacción de update, los registros que se muestran en los reportes podrían contener pcs que no cumplen con los parámetros.
No habrían efectos colaterales de invocar esta función varias veces al mismo tiempo

Función 2:

Si falla la operación, debe ser informado el cliente porque el registro permanecerá en la BD hasta que una operación de eliminación se complete.
En este caso en particular, el problema no es tan grande porque se trata de una eliminación individual, pero con un grupo de registros, si podría causar problema porque sería más complicado determinar cuáles fueron los que si se eliminaron con éxito.

Invocar esta función mas de una vez en el mismo momento provocaría que la función intentara eliminar un registro que ya no existe.

Función 3:

Si falla el update del precio, no es un gran problema porque se esta haciendo el cambio en un registro individual, pero si se tratara de una operación con múltiples registros es imperativo que se realicen todos o ninguno porque después seria imposible determinar sobre cuales registros se deben hacer los cambios en el precio y sobre cuales ya se realizó.

Invocar esta función mas de una vez en el mismo momento sobre los mismos registros provocaría que el precio final no fuera el esperado, pues las funciones iniciarían con un precio n e intentarían fijar el precio a n-100 pero lo esperado seria que el precio final sea n- 100*a (a siendo la cantidad de invocaciones)

Función 4:

En esta función es muy importante que se complete la operación de inserción en su totalidad o los registros podrían estar incompletos.
Si múltiples invocaciones de la función ocurren en el mismo momento, se podrían generar registros duplicados.

Ejercicio 3

Suponga que se ejecuta como una transacción T una de las cuatro funciones del ejercicio 1, mientras otras transacciones de esa misma función o de cualquier otra ocurren casi al mismo tiempo. ¿Qué diferencias podría observar para el resultado de T si todas las transacciones se ejecutan con un nivel de aislamiento READ UNCOMMITED en lugar de SERIALIZABLE?

Considere por separado el caso en que T es cada una de las 4 funciones definidas, es decir que su respuesta debe incluir el análisis para cada una de las 4 funciones

Función 1:

En caso de READ UNCOMMITED podría presentar ciertos modelos de computadora a los cuales les estaban modificando la velocidad o la ram en ese instante y que en realidad ya no cumplen con los requisitos, de tal modo que el reporte podría tener modelos que no cumplen con los requisitos, incluso podrían ya no existir esos modelos (si alguien más las estaba eliminando en otra transacción al mismo tiempo.)
Con SERIALIZABLE esto se prevendría porque una transacción se completaría antes que la otra, evitando registros fantasma o desactualizados.

Función 2:

Al usar SERIALIZABLE las transacciones se ejecutarían en orden y se prevendría que una transacción intentara eliminar un registro que ya no existe porque fue eliminado por otra transacción.

Función 3:

Con READ UNCOMMITED podrían darse resultados inesperados porque las transacciones al iniciarse tendrían el precio original n y tras ejecutarse, el resultado sería : (n-100), mientras que con SERIALIZABLE, ya que una transacción tendría que esperar a que el resto se complete, el resultado final sería : (n-200a) , a siendo la cantidad de transacciones del mismo tipo.

Función 4:

Con READ UNCOMMITED se producirían registros duplicados porque al iniciar ambas transacciones, ambas determinarían que ese registro no existe y ambas lo crearían. Mientras que con SERIALIZABLE, tras finalizar la primera transacción, la segunda ya no tendría la necesidad de crear el registro y se obtendría el resultado esperado: que solo exista un registro con los parámetros enviados.