Cristopher Jose Rodolfo Barrios Solis - 18207

Ejercicio 1

Basado en Ullman 2ª, ejercicio 6.6.1 al 6.6.3.

Desarrolle el siguiente esquema relacional en una base de datos PostgreSQL, utilizando los tipos de datos que considere más convenientes:

Producto (fabricante, modelo, tipo)
PC (modelo, velocidad, ram, disco, precio)

Recuerde del Laboratorio #8 sobre Álgebra relacional que tipo puede ser uno de tres valores: “PC”, “laptop” o “impresora”, pero en este laboratorio nos interesa solo trabajar con PCs.

Desarrolle un programa en Python capaz de interactuar con su base de datos y que contenga las funciones que se describen a continuación. Investigue y agregue a sus funciones las instrucciones BEGIN TRANSACTION, COMMIT y ROLLBACK en los momentos que considere pertinentes considerando que múltiples usuarios pueden invocar sus funciones concurrentemente, y recuerde especificar a su programa las transacciones que sean de solo lectura.

Función 1: Dada una velocidad y un tamaño de RAM como argumentos de la función, localice las PCs con esa velocidad y RAM, e imprima el número de modelo y precio de cada una.

Función 2: Dado un número de modelo, elimine la tupla para ese modelo tanto de la relación PC como de la relación Producto. Si el número de modelo no existe su programa debe hacerlo saber.

Función 3: Dado un número de modelo, decrecer el precio de ese modelo en $100.00. Si el número de modelo no existe su programa debe hacerlo saber.

Función 4: Dado un fabricante, número de modelo, velocidad de procesador, tamaño de RAM tamaño de disco duro y precio verifique si hay o no una PC que cumpla con esas características.

(i) Si sí encuentra un modelo muestre un mensaje de error al usuario
(ii) Si no se encuentra un modelo insértelo en las tablas Producto y PC.

Su programa debe tener una interfaz de línea de comando (CLI) o una interfaz gráfica (GUI) que permita interactuar correctamente con estas funciones especificando los parámetros solicitados.

Respuesta en el Zip

Cristopher Jose Rodolfo Barrios Solis - 18207

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 (find_pc):

- Problemas de atomicidad: Si el sistema se interrumpe durante la ejecución de la función, pueden ocurrir problemas de atomicidad si se interrumpe entre la ejecución de las consultas "SELECT" y "COMMIT". Si la interrupción ocurre antes del COMMIT, la transacción no se completará y los datos consultados no serán comprometidos en la base de datos.
- Problemas de concurrencia: Si dos o más invocaciones de la función ocurren al mismo tiempo, podrían ocurrir problemas de concurrencia si se modifican los mismos datos al mismo tiempo, lo que podría llevar a resultados inconsistentes.

Función 2 (delete_pc):

- Problemas de atomicidad: Si el sistema se interrumpe durante la ejecución de la función, pueden ocurrir problemas de atomicidad si la interrupción ocurre después del DELETE de la tabla "PC" pero antes del DELETE de la tabla "Producto" o del COMMIT. Si la interrupción ocurre después del DELETE de la tabla "PC" pero antes del DELETE de la tabla "Producto", la transacción se perderá y la tabla "Producto" no se actualizará correctamente. Si la interrupción ocurre después del DELETE de la tabla "PC" y antes del COMMIT, la transacción no se completará y los datos eliminados de la tabla "PC" no serán comprometidos en la base de datos.
- Problemas de concurrencia: Si dos o más invocaciones de la función ocurren al mismo tiempo para eliminar la misma PC, puede ocurrir un problema de concurrencia si ambas transacciones intentan eliminar los mismos datos al mismo tiempo. Esto podría llevar a resultados inconsistentes.

Función 3 (decrease_price):

- Problemas de atomicidad: Si el sistema se interrumpe durante la ejecución de la función, pueden ocurrir problemas de atomicidad si la interrupción ocurre después del UPDATE de la tabla "PC" pero antes del COMMIT. Si la interrupción ocurre después del UPDATE de la tabla "PC" y antes del COMMIT, la transacción no se completará y el precio de la PC no se actualizará correctamente.
- Problemas de concurrencia: Si dos o más invocaciones de la función ocurren al mismo tiempo para actualizar el precio de la misma PC, puede ocurrir un problema de concurrencia si ambas transacciones intentan actualizar el mismo dato al mismo tiempo. Esto podría llevar a resultados inconsistentes.

Función 4 (verify_pc):

- Problemas de atomicidad: Si el sistema se interrumpe durante la ejecución de la función, pueden ocurrir problemas de atomicidad si la interrupción ocurre después del INSERT en la tabla "Producto" pero antes del INSERT en la tabla "PC" o del COMMIT. Si la interrupción ocurre después del INSERT en la tabla "Producto" pero antes del INSERT en la tabla "PC", la transacción se perderá y la tabla "PC" no se actualizará correctamente. Si la interrupción ocurre después del INSERT en la tabla "Producto" y antes del COMMIT, la transacción no se completará y los datos no serán comprometidos en la base de datos.
- Problemas de concurrencia: Si dos o más invocaciones de la función ocurren al mismo tiempo para agregar la misma PC, puede ocurrir un problema de concurrencia si ambas transacciones intentan insertar los mismos datos al mismo tiempo. Esto podría llevar a resultados inconsistentes.

Cristopher Jose Rodolfo Barrios Solis - 18207

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.

En general, el nivel de aislamiento READ UNCOMMITTED permite que una transacción pueda leer datos que han sido modificados pero no confirmados por otra transacción, lo que podría generar inconsistencias en los datos. Por otro lado, el nivel de aislamiento SERIALIZABLE asegura que las transacciones se ejecuten de forma secuencial, evitando que dos o más transacciones accedan a los mismos datos simultáneamente y generando así una consistencia en los datos.

- **Función 1 (find_pc):** Si una transacción T ejecuta esta función con READ UNCOMMITTED, podría leer datos que aún no han sido confirmados por otra transacción y, por lo tanto, obtener resultados incorrectos o incompletos. Con SERIALIZABLE, se asegura que la transacción T lea datos consistentes y completos, evitando así posibles inconsistencias.

- **Función 2 (delete_pc)**: Si una transacción T ejecuta esta función con READ UNCOMMITTED, podría borrar una PC que otra transacción aún no ha confirmado como borrada, lo que generaría inconsistencias en los datos. Con SERIALIZABLE, se evita que dos o más transacciones intenten borrar la misma PC al mismo tiempo.

- **Función 3 (decrease_price):** Si una transacción T ejecuta esta función con READ UNCOMMITTED, podría decrementar el precio de una PC que otra transacción aún no ha confirmado como actualizada, lo que generaría inconsistencias en los datos. Con SERIALIZABLE, se evita que dos o más transacciones intenten actualizar el mismo campo de la misma PC al mismo tiempo.

- **Función 4 (verify_pc):** Si una transacción T ejecuta esta función con READ UNCOMMITTED, podría leer datos que otra transacción aún no ha confirmado como agregados, lo que podría generar inconsistencias en los datos. Con SERIALIZABLE, se evita que dos o más transacciones intenten agregar la misma PC al mismo tiempo.