ST0257 – Sistemas operativos
Proyecto 3: Paralelismo con bloqueos en un ambiente controlado

MBA, I.S. José Luis Montoya Pareja
Departamento de Informática y Sistemas
Universidad EAFIT
Medellín, Colombia, Suramérica

RESUMEN
En este proyecto, vamos a construir un manejador de base de datos (DBMS) simple desde cero. Un DBMS es un sistema de software que permite a los usuarios almacenar, recuperar y administrar datos en una forma estructurada. Su DBMS debería soportar accesos concurrentes de múltiples usuarios y asegurar que haya integridad y consistencia en los datos a través de varios mecanismos de concurrencia.

PALABRAS CLAVE
DBMS, acceso concurrente, integridad de los datos, consistencia de los datos, control de concurrencia, KarelJRobot.

CONTEXTO

DBMS

De acuerdo con IBM, un DBMS es esencialmente nada más que un sistema computarizado de mantenimiento de datos. Los usuarios del sistema tienen facilidades para realizar varios tipos de operaciones en dicho sistema, ya sea para la manipulación de los datos en la base de datos o para la gestión de la estructura de la base de datos misma. Los sistemas de gestión de bases de datos (DBMS) se clasifican según sus estructuras o tipos de datos. (IBM, 2010)

LA NECESIDAD

Los robots del proyecto 2 que construyó el profesor, deben dejar un rastro de por donde están pasando en un programa centralizado que vaya capturando el log del toString de cada Robot. La idea es que lo que se muestra en pantalla, se lleve a unas tablas muy simples con la siguiente información:

Robots

1. tipoRobot: Numérico (1, 2 ó 3)
2. idRobot: Numérico (Número mayor que 1)
3. encendido: Boolean (True/False)
4. Todos los atributos del objeto robot.

LogEventos

1. timeStamp: Fecha (en cualquier formato que incluya año, mes, día, hora, minuto y segundo)
2. idRobot: Numérico que debe estar en la tabla Robots
3. avenida: Numérico (donde está el robot)
4. calle: Numérico (donde está el robot)
5. sirenas: Numérico. Cantidad de sirenas que tiene en ese momento.

EstadoPrograma

1. timeStamp: Fecha (en cualquier formato que incluya año, mes, día, hora, minuto y segundo)
2. estado: Numérico que puede tener cualquier de los siguientes valores:
    a. 0: Apagado
    b. 1: Programa activo
    c. 2: Activado el modo hibernación
    d. 3: Recuperación

VariablesEstaticas

1. timeStamp: Fecha (en cualquier formato que incluya año, mes, día, hora, minuto y segundo)
2. Se almacenan todas las variables estáticas del programa cuando se activa el modo hibernación

CONCEPTOS BÁSICOS A MANEJAR

Debe diseñar su funcionalidad básica, incluyendo:

1. Almacenamiento de la información
2. Indexamiento de los datos
3. Consulta
4. Procesamiento de transacciones

Las bases de datos no estructuradas pueden ser una forma de solucionar el problema; usted puede proponer su manera de gestionar la información suficientemente argumentado, claro está.

El control de concurrencia debe ser implementado mediante mecanismos que permitan manejar accesos simultáneos a la base de datos a varios robots. Explique los conceptos como bloqueos, ordenación y control de concurrencia optimista para asegurar que las transacciones se ejecutan correctamente y se mantiene la consistencia de la información.

La gestión de transacciones debe hacerse mediante el concepto ACID (Atomicidad, Concurrencia, Isolation – aislamiento y Durabilidad). Implemente además niveles de aislamiento transaccional, registro de transacciones realizadas y recuperación para asegurarnos que las transacciones son ejecutadas de manera confiable y de una manera serializada.

Para probar la concurrencia, evalúe el performance de su sistema bajo escenarios de concurrencia (1 robot, 1 de cada tipo, 2 de cada tipo y los que

haya determinado que era su máximo en el proyecto 2). Mida métricas como el throughput, latencia y escalabilidad para asegurar la efectividad de su implementación.

Debe considerarse que puede haber un proceso que vaya y consulte la información en la base de datos y que el sistema responda a consultas simples a las tablas.

CONSIDERACIONES GENERALES

1. El robot debe dejar rastro en la base de datos de:
    a. Cuando se crea
    b. Cada que se mueve
    c. Cuando se apaga

2. El desarrollo de la práctica puede ser individual o en equipos de máximo tres personas.

3. La entrega de la práctica se realizará adjuntando el código fuente y el informe por el buzón recepción de trabajos de Eafit Interactiva (cualquier otro medio no será admitido).

4. Se debe informar al profesor a más tardar el 12 de abril a las 6:00 p.m. los integrantes del equipo.

5. El informe final es una presentación que deberá contener una breve descripción de cómo funciona el programa, que dificultades debieron superar para el desarrollo de la práctica y posibles mejoras que consideran, se puede hacer a la misma.

6. La práctica se debe realizar en cualquier lenguaje y deben proveer la conectividad para que la aplicación del profesor de Java se conecte a su DBMS para almacenar la información.

7. Cada semana los jueves, se sacará un espacio de 10 a 15 minutos al inicio de la clase para hablar de la práctica y resolver dudas. Durante ese tiempo, al azar se elegirá a uno o varios equipos para que expongan el avance. Esto se hará mediante preguntas que el profesor realizará de la siguiente manera:

    a. Qué entendieron de la práctica y cómo van a solucionar el problema
    b. Diseño de la solución
    c. Cómo van con la implementación
    d. Problemas que tengan este momento para cumplir con el programa

8. Criterios de evaluación (ver Anexo 1)

FECHA DE ENTREGA
Viernes 17 de mayo en clase a través de Eafit Interactiva.

BONUS TRACK
En el diseño de la base de datos (pagina 2 tabla EstadoPrograma) se habló de un esquema de Hibernación. Para ganar esta bonificación (reemplaza la nota que tengan en el 10%), el programa de Java debe modificarse para que:

1. El hilo principal que crea los robots, reciba el comando nuevo número 2 (Hibernar)
2. Cuando se active la hibernación, se deben bloquear todos los robots, guardar en la base de datos la posición en la que están y el estado en el que se encuentran. Adicionalmente, se debe almacenar la información de cuantas minas hay en cada punto en ese momento y terminar el programa.
3. Cuando el programa arranque, si se da cuenta que en la base de datos está activa la hibernación, debe leer:
    a. Posición donde están los robots, recreándolos y guardando los valores de cada atributo del robot (incluyendo si lleva sirenas en ese momento).
    b. Todo el contenido de las variables estáticas.
    c. Asignar a cada posición del mundo donde haya sirenas, la cantidad que tenga en ese momento.
    d. Y los hilos deben seguir ejecución en el punto donde estaban.
4. No se tomará en consideración el caso de recuperación de fallas ante la cancelación del programa (deja de correr de manera intempestiva). Si ese es el caso, se reiniciará el programa desde cero sin tomar en cuenta la información de la base de datos.
5. Para la calificación del bonus track, aplican los mismos criterios de calificación del anexo 1.

SUSTENTACIÓN
Martes 28 y Jueves 30 de mayo en clase. El mecanismo de sustentación es el siguiente:

1. Cada equipo muestra su desarrollo ejecutando en vivo.
2. Debe mostrar cómo solucionó cada uno de los retos y cómo lo relacionaron con conceptos vistos en clase.
3. Se realizarán preguntas por parte del docente para validar el entendimiento individual de los conceptos aplicados. Esto significa que, aunque el trabajo es en equipo, la nota del trabajo puede ser diferente para cada uno de acuerdo con la calidad de las respuestas.

Nombre de la asignatura: Sistemas Operativos

Competencia a la que aporta la asignatura: Conocer el sistema operativo del computador para un mejor desarrollo, diseño y ejecución de las aplicaciones y aplicar nuevas soluciones

Resultado de asignatura evaluado: Solución de problemas de persistencia de la información aplicando conceptos de concurrencia.

Evento evaluativo: Proyecto 3 Porcentaje del evento evaluativo: 30%

Criterios (que tributen al RA de asignatura)	**Cumple con altos estándares (4.5-5)**	**Cumple a satisfacción (4-4.4)**	**Cumple parcialmente (3.5-3.9)**	**Incumple parcialmente (2.5-3.4)**	**Incumple totalmente (0-2.4)**	**Peso asignado al criterio sobre la calificación.**
Análisis de fundamentación para la solución: Persistencia, concurrencia Claridad en el concepto	Entiende completamente la necesidad y plantea varias opciones de solución. Utiliza conceptos adecuadamente para la solución.	Entiende completamente la necesidad y plantea una opción de solución.	Omitió un elemento clave para el entendimiento de la necesidad	Omitió varios elementos para el entendimiento de la necesidad.	Demuestra poco o nulo entendimiento del problema.	40%
Diseño de la solución Solución óptima	El diseño tiene en cuenta los conceptos vistos en clase y argumenta la elección de su solución. La solución elegida es la óptima para el problema.	El diseño tiene en cuenta los conceptos vistos en clase y argumenta la elección de su solución.	Aunque se tuvieron en cuenta los conceptos vistos en clase, no hubo argumentación correcta en la elección de la solución.	No se tuvieron en cuenta los conceptos vistos en clase.	Demuestra poco o nulo entendimiento de la persistencia y de la concurrencia al momento de explicar la solución.	40%
Funcionalidad Calidad de la solución frente a necesidad planteada	El programa funciona correctamente.	La solución elegida no es la óptima pero el programa funciona correctamente	El programa con los conceptos vistos en clase no funciona correctamente.	El programa no tiene en cuenta ningún concepto visto en clase y la solución funciona correcta o parcialmente.	Se entrega la solución parcialmente o no se entrega ninguna solución.	20%