Práctica 2.ª
Programación con semáforos en Ada

Escuela de Ingeniería y Arquitectura
Depto. de Informática e Ingeniería de Sistemas

1. Objetivos

1. Trabajar en Ada con semáforos
2. Implementar una solución al problema de los lectores-escritores aplicado a una base de datos con distintos niveles de bloqueo
3. Trabajar con los mecanismos de genericidad de Ada

2. Introducción

En esta práctica se va a trabajar con el problema los lectores-escritores, accediendo concurrentemente a una base de datos común. Asegurar la integridad de los datos exige que las operaciones de escritura se hagan en exclusión mutua con respecto a cualquier otra operación de lectura o escritura. Esto se suele implementar mediante *cerrojos* (*locks*), de manera que quien accede a un dato o conjunto de datos debe tomar previamente los cerrojos asociados a esos datos, condición necesaria para acceder a los mismos. Una vez ha terminado su operación, debe liberar los cerrojos, de manera que otros procesos puedan acceder a los datos.

Los cerrojos pueden establecerse a distintos niveles, pudiendo implicar desde el bloqueo de la base de datos completa hasta el bloqueo de registros concretos de tablas concretas. Cuanto *más* bloquee un cerrojo, más sencilla es su manipulación, y más entrelazados elimina (el sistema tendrá *menos concurrencia*). En general, es deseable que el número de entrelazados eliminados sea tan pequeño como sea posible. El objetivo de esta práctica es definir distintas formas de bloqueos en la base de datos y medir sus prestaciones. En esta práctica, los accesos en exclusión mutua se van a implementar a través de semáforos.

Vamos a simular un pequeño sistema gestor de base de datos dedicado a un problema en concreto: la gestión de una biblioteca. Se va a trabajar con las entidades *socio*, *libro* y *préstamo*. Los datos de este sistema van a ser gestionados por un TAD denominado `Biblioteca` y van a estar almacenados en memoria a través de listas doblemente enlazadas. El TAD `Biblioteca` va a permitir la gestión de las tres entidades a través de las operaciones típicas: inserción, eliminación, modificación y consulta, que deben ejecutarse de forma atómica.

En el contexto de esta práctica, no nos van a interesar aspectos como la unicidad de claves primarias o la integridad referencial.

3. Material de partida

En la página web `http://webdiis.unizar.es/~ezpeleta/doku.php?id=practicas_0708_pc` se encuentra un fichero comprimido en formato ZIP denominado «`pract2.zip`» que contiene:

- Los TAD `Socio`, `Libro` y `Prestamo`, con sus ficheros de especificación e implementación. En el contexto de las simulaciones que se van a plantear en esta práctica, solo nos interesará conocer que cada tipo de entidad tiene definida una clave primaria: para los socios, el DNI; para los libros, su ISBN; y para los préstamos, el par (DNI, ISBN). Estas claves están accesibles a partir de datos de tipo `tpSocio`, `tpLibro` y `tpPrestamo` a través de las funciones `dni`, `isbn` e `id`, respectivamente. También tienen una operación de construcción (`crear`), otra para convertir sus valores en una cadena de caracteres que pueda escribirse en la pantalla (`cadena`) y otra para leer datos de los distintos tipos desde un fichero de texto (`leer`).
- El TAD genérico `Lista`, que contiene el código básico para permitir las operaciones de inserción, eliminación, modificación y consulta de datos.

  El TAD, debido a necesidades de implementación, no tiene definido ningún operador de construcción, aunque está definido de tal forma que cada declaración de variables del tipo `tpLista` asegura que dicha variable estará inicializada como una lista vacía.
- El TAD `Semaforo`, que contiene una implementación en Ada de semáforos binarios.
- La especificación e implementación de un TAD denominado `Biblioteca`, que permite la operaciones de gestión de la biblioteca sobre las entidades de socios, libros y préstamos. El TAD `Biblioteca` encapsula una lista por cada tipo de entidad, y ofrece operaciones para realizar las operaciones de inserción, eliminación, modificación y consulta.
- Un paquete denominado `ProcesadorGenerico` con un procedimiento genérico denominado `procesar` capaz de leer de un fichero de texto la definición de una operación concreta con la base de datos y ejecutarla.

- Un programa denominado `Simulacion`, que es el programa principal.
- Un conjunto de ocho ficheros de texto distintos que incluyen ejemplos simulados de uso de la base de datos por ocho clientes.

4. Descripción del trabajo a realizar

4.1. Versión inicial sin control de la exclusión mutua

Como trabajo previo a la implementación de los distintos niveles de bloqueo que se plantean en las secciones siguientes, y con el objeto de que te familiarices con el código suministrado, las primeras tareas a realizar no tendrán en cuenta ningún nivel de bloqueo.

En primer lugar, es necesario modificar el código fuente del programa `Simulacion`. La versión suministrada corresponde con un programa secuencial que lee el contenido del fichero «`traza-1.txt`» y ejecuta las operaciones en él planteadas contra la base de datos. La modificación que se pide consiste en definir un conjunto de ocho tareas que ejecuten, de forma concurrente, las instrucciones especificadas en cada uno de los ocho ficheros de traza suministrados.

Fíjate en el bloque de captura de excepciones que hay en el código del programa `Simulacion` suministrado y ten especial cuidado en colocarlo en el lugar adecuado cuando escribas las tareas. Es en esta versión del programa (la única que no va a controlar el acceso en exclusión mutua) en la que pueden aparecer más excepciones, ya que es posible que la representación de la base de datos se corrompa.

Como resultado de esta parte, deberás entregar los ficheros con el código fuente y las trazas en un directorio denominado «`1-concurrente`».

4.2. Simulación con bloqueo de toda la base de datos

La primera de las simulaciones asegurará la integridad de las operaciones bloqueando la base de datos completa. Es decir, cuando una tarea realice una operación de escritura en la base de datos, ésta deberá bloquearse por completo y no permitir accesos, ni de lectura ni de escritura por parte de otras tareas. Para ello, es preciso modificar el código del procedimiento `procesar`, identificando qué partes del código son lectoras y cuáles son escritoras e implementando en dichos lugares una solución al problema de los lectores y escritores basada en semáforos.

Una vez realizados los cambios, ejecuta el simulador varias veces y anota el tiempo que le cuesta llevar a cabo cada ejecución. Compara estos tiempos con los obtenidos en la parte anterior y explica las causas de esta variación.

Como resultado de esta parte, deberás entregar los ficheros con el código fuente y las trazas en un directorio denominado «`2-bloqueoBD`».

4.3. Simulación con bloqueo de tabla

La segunda de las simulaciones asegurará la integridad de las lecturas y escrituras bloqueando únicamente las tablas necesarias en lugar de bloquear la base de datos completa. Es decir, cuando una tarea realice una operación de escritura relativa a un registro de una tabla de la base de datos, la tabla deberá bloquearse y no se permitirá el acceso a cualquier dato de esa tabla por parte de otras tareas.

Existen varias alternativas relativas a los ficheros que se pueden modificar para lograr este comportamiento. Identifícalas y justifica la razón de escoger la que finalmente implementes.

Una vez hechas las modificaciones, ejecuta de nuevo el código, comparando los resultados con los obtenidos en las dos partes anteriores. Explica las causas de los nuevos resultados obtenidos

Como resultado de esta parte, deberás entregar los ficheros de código fuente y las trazas en un directorio denominado «`3-bloqueoTabla`».

4.4. Simulación con bloqueo de registro

El tercero de los escenarios de simulación debería ser aquel en el que se asegura la integridad de los accesos bloqueando únicamente los registros necesarios: cuando una tarea realice una operación de escritura relativa a un registro de una tabla de la base de datos, debería ser solo ese registro el que estará bloqueado.

Teniendo en cuenta que la representación para cada tabla son listas doblemente enlazadas, piensa en si es posible realizar operaciones de consulta, inserción, modificación y eliminación bloqueando únicamente un registro cada vez. Analiza qué sería lo mínimo que habría que bloquear para cada una de las cuatro operaciones contempladas y cómo podría implementarse dicho bloqueo con semáforos.

El resultado de esta parte es una breve discusión sobre estos aspectos en el fichero «`segunda.txt`». Si te animas a implementar la solución (habla antes con el profesor de prácticas para que te indique si vas bien encaminado), puedes entregar los ficheros con el código fuente y las trazas en un directorio denominado «`4-bloqueoRegistro`».

4.5. Análisis de resultados

Hay que escribir un pequeño informe con la explicación de las distintas estrategias de bloqueos aplicadas, de los resultados esperados, de los resultados obtenidos, de las ventajas e inconvenientes de cada una de las soluciones (desde el punto de vista de un programa concurrente –grado de concurrencia permitida–), de los posibles problemas que puedan surgir (¿tiene el programa un comportamiento equitativo?, etc.) y con las respuestas a las preguntas que se hayan intercaladas en el resto del enunciado.

5. Entrega de material

- Como resultado de esta práctica hay que enviar por correo electrónico al profesor de prácticas (`latre@unizar.es`) un fichero comprimido en formato ZIP denominado «`pract2.zip`» que contenga los tres directorios solicitados (cuatro si se implementa la estrategia de bloqueo de registro) y el informe, denominado «`segunda.txt`».
- Fecha límite de entrega: 18 de abril de 2012.