Proyecto I – LinkedDB

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Segundo Semestre 2017
Valor 25%

TEC | Tecnológico de Costa Rica

Objetivo General

- ➔ Diseñar e implementar un motor de bases de datos sencillo basado en listas enlazadas

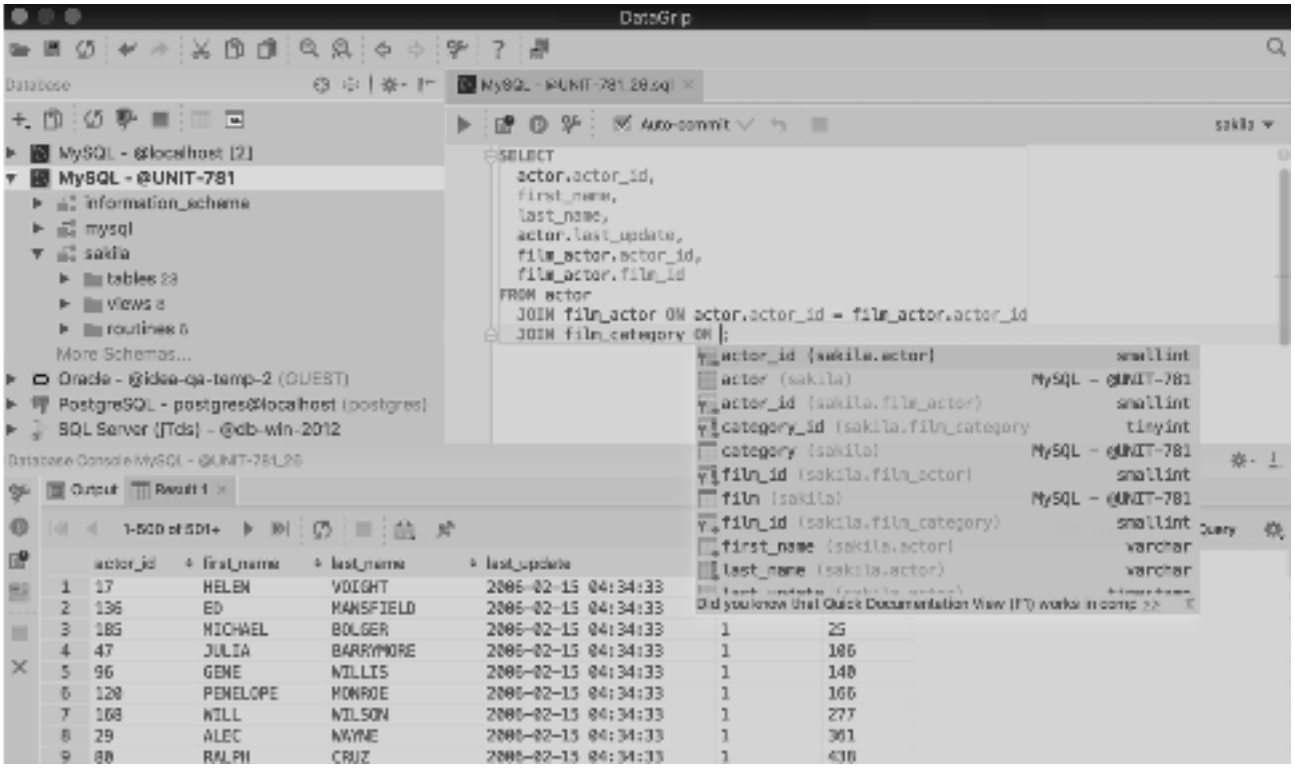
Objetivos Específicos

- ➔ Implementar listas enlazadas y algunas variaciones
- ➔ Aprender sobre algunos conceptos básicos de bases de datos
- ➔ Investigar y desarrollar una aplicación en el lenguaje de programación Java
- ➔ Investigar acerca de programación orientada a objetos en Java.
- ➔ Aplicar patrones de diseño en la solución de un problema.

Descripción del Problema

LinkedDB es un motor de bases de datos noSQL. Permite definir, insertar, eliminar, actualizar y buscar documentos JSON. Un documento JSON contiene objetos JSON que representan una entidad específica, por ejemplo un estudiante específico, una persona, un vehículo, entre otras.

LinkedDB se empaqueta como un único JAR. El usuario ejecuta dicho JAR y se presenta una interfaz gráfica desarrollada con Java FX. Dicha interfaz es similar a una herramienta de bases de datos, como DBVisualizer, DataGrip, MySQL Workbench, entre otros:



A la izquierda se muestra un árbol de JSON Stores. El usuario puede crear un JSON store con el nombre que desee. Físicamente, un JSON store es una carpeta en una ruta definida por LinkedDB. Cada vez que se crea un store, se crea una nueva carpeta y se almacena en una lista enlazada.

El usuario puede crear documentos JSON en un store particular. A la hora de crear un documento, se le solicitan los siguientes datos al usuario:

- Nombre del documento
- Lista de atributos del documento, donde cada atributo se define como
    - Nombre del atributo
    - Tipo del atributo: entero, flotante, cadena, fecha-hora
    - Tipo especial: llave primaria o llave foránea. Una llave foránea indica el atributo del otro documento que hace referencia. Adelante, se explica más esto.
    - Requerido o no requerido
    - Valor por defecto en caso de no ser requerido

Cuando se crea un documento, se crea un archivo JSON con el nombre del documento dentro de la carpeta correspondiente al store seleccionado, se muestra en el árbol de la izquierda en la interfaz gráfica y se agrega como una nueva lista dentro de la lista del Store. Al presionar click derecho sobre el documento, el usuario puede:

- Mostrar un listado con todos los objetos en memoria. En este caso se deben considerar las llaves foráneas también.
- Agregar un nuevo objeto JSON al documento: cuando se agrega un nuevo objeto, se debe validar que el objeto que se trata de insertar cumpla con la estructura del documento definido previamente. El objeto se agrega en memoria.
- Eliminar todos los objetos JSON del documento. Elimina los documentos en memoria.
- Eliminar un objeto al buscarlo por llave. Elimina un objeto en memoria.
- Buscar objetos por cualquier atributo en memoria.
- Actualizar uno o más objetos. Se especifica los atributos por actualizar. Se especifica también una condición de búsqueda que indique cuáles objetos actualizar. La actualización se realiza en los objetos en memoria.

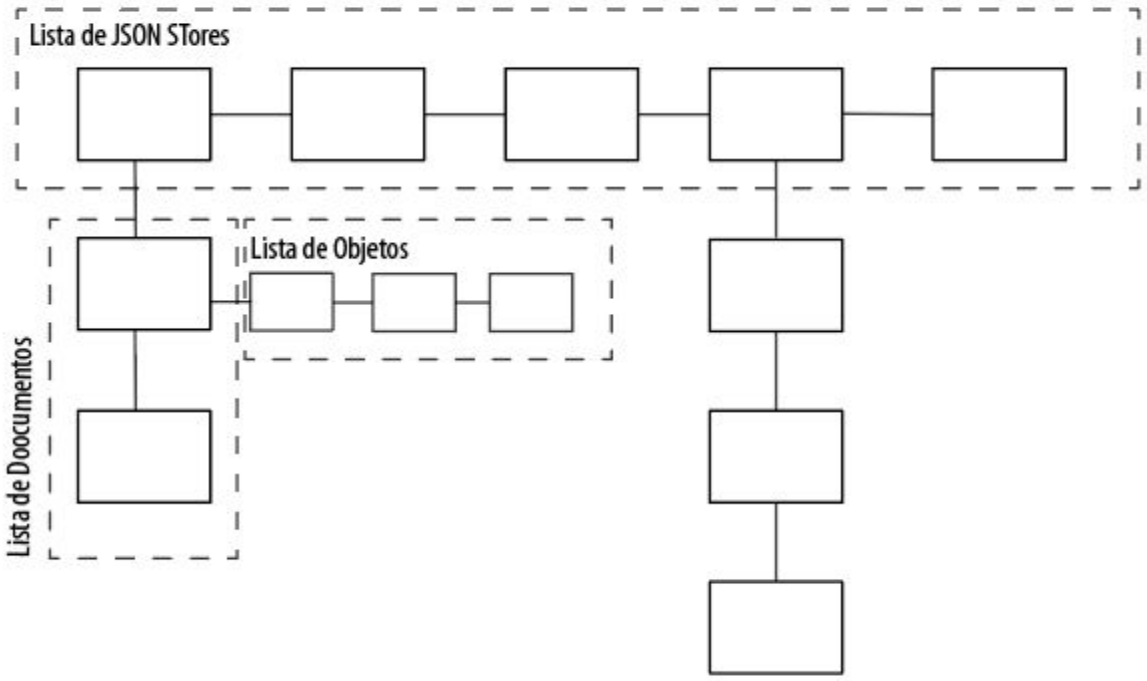
La interfaz gráfica, tiene un botón "Commit" que se activa luego de realizar una inserción, eliminación o actualización. La operación Commit, escribe en disco los cambios realizados.

Es importante notar, que cuando la interfaz gráfica carga, se deben cargar los archivos a memoria. Si la aplicación se cierra sin hacer commit, los cambios se descartan.

Una llave foránea es una referencia a otro objeto JSON del mismo u otro Store. Por ejemplo, el usuario podría crear el documento Persona cuya llave primaria es la cédula. Posteriormente puede crear el documento Vehículo, que tiene un atributo llave foránea propietario, que hace referencia al atributo cédula del documento persona. Esto implica que:

- No se puede poner un valor en el atributo propietario que no tenga su correspondiente Persona.
- No se puede eliminar una Persona que aparezca como propietario en el documento Vehículo.
- Se tendría que eliminar el vehículo primero para eliminar la persona.

Para aclarar la estructura de datos en memoria, se tendría lo siguiente:



La lista de JSON Stores es doble. La Lista de Documentos es Doble Circular y la lista de Objetos es sencilla.

Documentación requerida

1. Se deberá documentar el código fuente utilizando JavaDoc.
2. Se deberá entregar un documento que contenga:
    a. Todas las partes estándar: Portada, índice, introducción, conclusión, bibliografía y anexos.
    b. El cuerpo del documento debe contener:
        - ➔ Breve descripción del problema
        - ➔ **Planificación y administración del proyecto**
            - ◆ Lista de features e historias de usuario identificados de la especificación
            - ◆ Distribución de historias de usuario por criticalidad y secuencia de uso
            - ◆ Minimal system span
            - ◆ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
            - ◆ Asignación de user stories por miembro del equipo
            - ◆ Descomposición de cada user story en tareas.
            - ◆ Bitácora de trabajo que muestre tiempo invertido para cada actividad y la fecha en que se realizó
        - ➔ **Diseño**
            - ◆ Diagrama de arquitectura de la solución
            - ◆ Diagrama de componentes
            - ◆ Diagrama de secuencia (Seleccionar al menos 4 Historias de Usuario)
            - ◆ Diagrama de clases

- ➔ **Implementación**
  - ◆ Descripción de las bibliotecas utilizadas
  - ◆ Descripción de las estructuras de datos desarrolladas.
  - ◆ Descripción detallada de los algoritmos desarrollados.
  - ◆ Problemas encontrados: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo. Incluye descripción detallada, intentos de solución sin éxito, solución encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 25% de la nota del curso
3. El trabajo es individual.
4. Es obligatorio utilizar un manejador de versiones del código. Puede ser git o svn. Se revisará que cada commit lleve comentarios relevantes relacionados con alguna tarea identificada en la sección de planificación
5. Los proyectos que no cumplan con los siguientes requisitos **no serán revisados**:
   a. Toda la solución debe estar integrada
6. El código tendrá un valor total de 75%, la documentación 15% y la defensa 10%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se debe enviar el código (preliminar) y la documentación a más tardar a las 23:59 del día de la fecha de entrega al email del profesor. **Se debe seguir el formato del Subject descrito en el Programa del Curso.**
9. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
10. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial, momento en el cual deben enviar un email con el código fuente, si no lo hacen se asumirá que la versión oficial del código fue la enviada el día de la fecha de entrega junto con la documentación. **Se debe enviar en el mismo correo del punto 7**
11. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en Java, en caso contrario se obtendrá una nota de 0.
    f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
    g. La nota de la documentación debe ser acorde a la completitud del proyecto.
12. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
13. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.

14. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
15. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
16. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.