Instituto Tecnológico de Costa Rica

Ingeniería en Computadores

Algoritmos y Estructuras de Datos I (CE 1103)

# TinySQLDb

Alejandro Arias Alfaro

Josue Venegas Arias

II Semestre 2024

# Introducción

El presente proyecto tiene como objetivo el desarrollo de TinySQLD, un motor de base de datos relacional sencillo, diseñado para simular el comportamiento básico de los motores de bases de datos comerciales, utilizando el lenguaje de programación C#. Este motor permite a los usuarios realizar operaciones fundamentales sobre bases de datos, tales como la creación de bases de datos y tablas, la inserción, actualización y eliminación de registros, así como la implementación de índices para optimizar consultas.

La implementación de TinySQLD se ha llevado a cabo utilizando el paradigma de orientación a objetos, lo que garantiza la modularidad y escalabilidad del sistema. Además, se ha integrado un cliente en PowerShell que permite a los usuarios interactuar con el motor de base de datos mediante un subconjunto del lenguaje SQL, lo que facilita su uso a través de la línea de comandos.

En este proyecto se ha implementado un motor de base de datos relacional básico llamado TinySQLDb. El objetivo de este sistema es familiarizarse con los componentes clave de un motor de base de datos, desde el manejo de la comunicación con el cliente hasta la gestión de datos y la ejecución de consultas. TinySQLDb permite a los usuarios ejecutar un subconjunto de sentencias SQL a través de una interfaz de comandos en PowerShell. Este motor incluye funcionalidades para crear bases de datos, manipular tablas, insertar datos, ejecutar consultas, y utilizar índices para optimizar la búsqueda de información.

# Breve descripción del problema

El proyecto tiene como propósito diseñar e implementar un sistema administrador de bases de datos relacional básico, TinySQLDb, que permita ejecutar sentencias SQL sencillas a través de un cliente en PowerShell. El desafío principal es crear un motor de bases de datos funcional y eficiente que ofrezca las operaciones básicas como la creación de bases de datos y tablas, la inserción de datos, la actualización de registros y la utilización de índices para mejorar el rendimiento de las consultas.

En la práctica, se busca simular cómo funcionan motores de bases de datos comerciales, pero de manera simplificada, para que los estudiantes se familiaricen con los conceptos fundamentales detrás de los sistemas de gestión de bases de datos (DBMS).

# ### Requerimiento 000: Cliente PowerShell

**Implementación**

Se ha desarrollado un módulo de PowerShell que permite a los usuarios ingresar comandos SQL mediante la línea de comandos. El cliente envía estas consultas al servidor utilizando sockets, y los resultados son mostrados en formato de tabla en la consola. Para formatear los resultados en tabla, PowerShell proporciona funciones nativas que se han utilizado para este propósito.

**Limitaciones**

El cliente solo soporta un subconjunto limitado del lenguaje SQL. Además, no es compatible con transacciones ni consultas SQL complejas. En este caso al ser un TinySQL no es un problema.

# ### Requerimiento 001: Creación de base de datos

**Implementación**

Una base de datos es representada como una carpeta en el sistema de archivos. Dentro de esta carpeta se almacenan los archivos binarios correspondientes a cada tabla, así como los índices asociados a estas. Se ha implementado un catálogo de sistema que almacena metadatos sobre las bases de datos existentes.

**Limitaciones**

No se permite realizar operaciones sobre bases de datos que no se encuentren en el sistema de archivos.

# ### Requerimiento 002: Sentencia `CREATE DATABASE`

**Implementación**

Se ha implementado la funcionalidad para crear una base de datos a través de la sentencia `CREATE DATABASE <database-name>`. El servidor verifica si el nombre de la base de datos ya existe y, de no ser así, crea una carpeta correspondiente.

**Limitaciones**

No se permite el uso de caracteres especiales en los nombres de las bases de datos

# ### Requerimiento 003: Sentencia `SET DATABASE`

**Implementación**

Esta sentencia establece el contexto para las siguientes operaciones SQL del cliente. El servidor valida que la base de datos seleccionada exista antes de permitir que se realicen consultas sobre ella.

**Limitaciones**

El servidor no ofrece soporte para manejar múltiples contextos de base de datos simultáneamente.

# ### Requerimiento 004: Sentencia `CREATE TABLE`

**Implementación**

Esta sentencia permite la creación de tablas dentro de una base de datos. Se especifica el nombre de la tabla y las definiciones de las columnas. Los tipos de datos soportados incluyen `INTEGER`, `DOUBLE`, `VARCHAR` y `DATETIME`.

**Limitaciones**

Solo se soportan tipos de datos básicos, y el tamaño de los campos `VARCHAR` debe ser especificado durante la creación de la tabla.

# ### Requerimiento 005: Sentencia `DROP TABLE`

**Implementación**

Se implementó la funcionalidad para eliminar tablas vacías dentro de una base de datos utilizando la sentencia `DROP TABLE`.

**Limitaciones**

No se pueden eliminar tablas que contengan registros.

# ### Requerimiento 006: Sentencia `CREATE INDEX`

**Implementación**

Se implementaron índices utilizando estructuras de datos como árboles B y árboles binarios de búsqueda. Los índices permiten una búsqueda más rápida en columnas específicas.

**Limitaciones**

Solo se permite un índice por columna, y no se pueden tener valores repetidos en columnas indizadas.

# ### Requerimiento 007: Sentencia `SELECT`

**Implementación**

La sentencia `SELECT` permite recuperar información de las tablas. Se utiliza QuickSort para ordenar los resultados si se especifica la cláusula `ORDER BY`. Si la columna en la cláusula `WHERE` está indizada, se utiliza el índice para realizar la búsqueda más eficiente.

**Limitaciones**

No se soportan consultas con subconsultas o combinaciones de tablas (`JOIN`).

## ### Requerimiento 008: Sentencia `UPDATE`

**Implementación**

La sentencia `UPDATE` permite modificar los valores de las columnas de una tabla. Si la columna especificada en `WHERE` está indizada, se utiliza el índice para acelerar la búsqueda.

**Limitaciones**

El `UPDATE` solo es efectivo sobre columnas de una tabla, sin soporte para operaciones avanzadas.

## ### Requerimiento 009: Sentencia `DELETE`

**Implementación**

Esta sentencia elimina filas de una tabla. Los índices asociados se actualizan automáticamente tras la eliminación de los registros.

**Limitaciones**

No se permite eliminar múltiples tablas en una sola sentencia.

## ### Requerimiento 010: Sentencia INSERT INTO

**Implementación**

La sentencia `INSERT INTO` permite agregar nuevos registros a una tabla. Los valores son validados contra las definiciones de columna antes de su inserción.

**Limitaciones**

Solo se permite la inserción de un registro a la vez.

# Problemas encontrados

### Manejo de Concurrencia y Sockets

La comunicación entre el cliente (PowerShell) y el servidor se realiza mediante sockets, al implementar una base de datos se necesitó investigar para poder implementar el proyecto de manera correcta.

### Validación de Consultas SQL

Asegurar que las consultas SQL recibidas desde el cliente son válidas tanto sintáctica como semánticamente fue un reto especialmente si decides implementar incluso un subconjunto pequeño de SQL.

### Gestión de Archivos Binarios

El manejo de archivos binarios para almacenar las bases de datos, tablas e índices es complejo, especialmente al gestionar actualizaciones, eliminaciones y la integridad de los datos.

Al momento de dar formato a las tablas fue complicado y en algunos casos no se logro esto.

Manejo de control de versiones en equipo.

Manejar correctamente el uso de Git y GitHub, especialmente en equipo, puede ser un reto si no se tiene experiencia previa en control de versiones en equipo.

# Diagrama de clases UML