
**Instituto Tecnológico de Costa Rica**

**Escuela de Ingeniería en Computadores**

**Proyecto 2**

# TheDeGeO TinySQL

**Autores:**

**David Obando Blanco**

**Andy López Mora**

**Octubre 2024**

# Índice

# 1. Introducción

Este proyecto consiste en la implementación de un sistema administrador de bases de datos relacional básico denominado TinySQLDb. Su propósito es ayudar a los estudiantes a entender el funcionamiento de un motor de bases de datos al implementar componentes clave en C#, utilizando el paradigma de programación orientada a objetos.

El sistema permite ejecutar sentencias SQL desde un cliente de Powershell, el cual se comunica con un servidor mediante sockets, procesando consultas y administrando datos en el sistema de archivos. Este documento detalla la implementación del sistema, los retos encontrados y cómo se resolvieron los diferentes requerimientos.

# 2. Descripción del problema

## 2.1. Requerimiento 000 - Cliente PowerShell

### 2.1.1. Implementación:

Aunque el código del cliente PowerShell no se proporcionó, el servidor está diseñado para manejar conexiones entrantes y procesar consultas SQL. La clase Server en Server.cs implementa un servidor TCP que escucha en un puerto específico (por defecto, el puerto 11000 en localhost). El servidor está preparado para recibir mensajes JSON, lo que sugiere que está diseñado para interactuar con un cliente PowerShell que enviaría consultas en este formato.

### 2.1.2. Limitaciones:

El servidor actualmente solo soporta una conexión paralela, lo que podría ser un cuello de botella si múltiples clientes intentan conectarse simultáneamente.

### 2.1.3. Problemas potenciales:

La falta de un mecanismo de autenticación en el servidor podría plantear problemas de seguridad, permitiendo potencialmente a cualquier cliente conectarse y ejecutar consultas sin restricciones.

## 2.2. Requerimientos 001-003 - Modelado de Base de Datos, System Catalog y Operaciones Básicas

### 2.2.1. Implementación:

El SQLQueryProcessor implementa métodos para manejar la creación de bases de datos (CREATE DATABASE) y el establecimiento del contexto de la base de datos (SET DATABASE). Estos métodos extraen el nombre de la base de datos de la sentencia SQL y delegan la ejecución a clases específicas (CreateDatabase y SetDatabase). Sin embargo, la implementación real de estas clases no se proporciona en el código dado.

### 2.2.2. Limitaciones:

No se observa una implementación explícita del System Catalog o del sistema de archivos para almacenar las bases de datos y tablas. Esto sugiere que estas funcionalidades cruciales aún necesitan ser desarrolladas.

### 2.2.3. Aspectos a mejorar:

Sería beneficioso implementar una clase dedicada para manejar el System Catalog y las operaciones de archivos, asegurando que la metadata de las bases de datos y tablas se almacene y gestione adecuadamente.

## 2.3. Requerimientos 004-006 - Creación de Tablas, Eliminación de Tablas y Creación de Índices

### 2.3.1. Implementación:

El SQLQueryProcessor incluye métodos para manejar la creación y eliminación de tablas, así como la creación de índices. Para la creación de tablas, el procesador extrae el nombre de la tabla y las definiciones de las columnas de la sentencia SQL. La creación de índices se maneja de manera similar, extrayendo la información relevante como el nombre del índice, la tabla, la columna y el tipo de índice.

### 2.3.2. Limitaciones:

Aunque se define una clase CIndex para representar los índices, no se observa la implementación real de las estructuras de árbol B o BST mencionadas en los requerimientos. Además, no hay

evidencia de la verificación de valores repetidos al crear índices en tablas existentes.

### 2.3.3. Aspectos a mejorar:

Es necesario implementar la lógica para crear y mantener las estructuras de árbol en memoria, así como la verificación de unicidad al crear índices en tablas con datos existentes.

## 2.4. 007-010 - Operaciones SELECT, UPDATE, DELETE, INSERT

### 2.4.1. Implementación:

El SQLQueryProcessor incluye métodos para manejar estas operaciones CRUD básicas. Cada método extrae la información relevante de la sentencia SQL (como nombres de tablas, columnas, cláusulas WHERE, etc.) y delega la ejecución a clases específicas para cada operación. Este enfoque permite una buena separación de responsabilidades y facilita la extensibilidad del sistema.

### 2.4.2. Limitaciones:

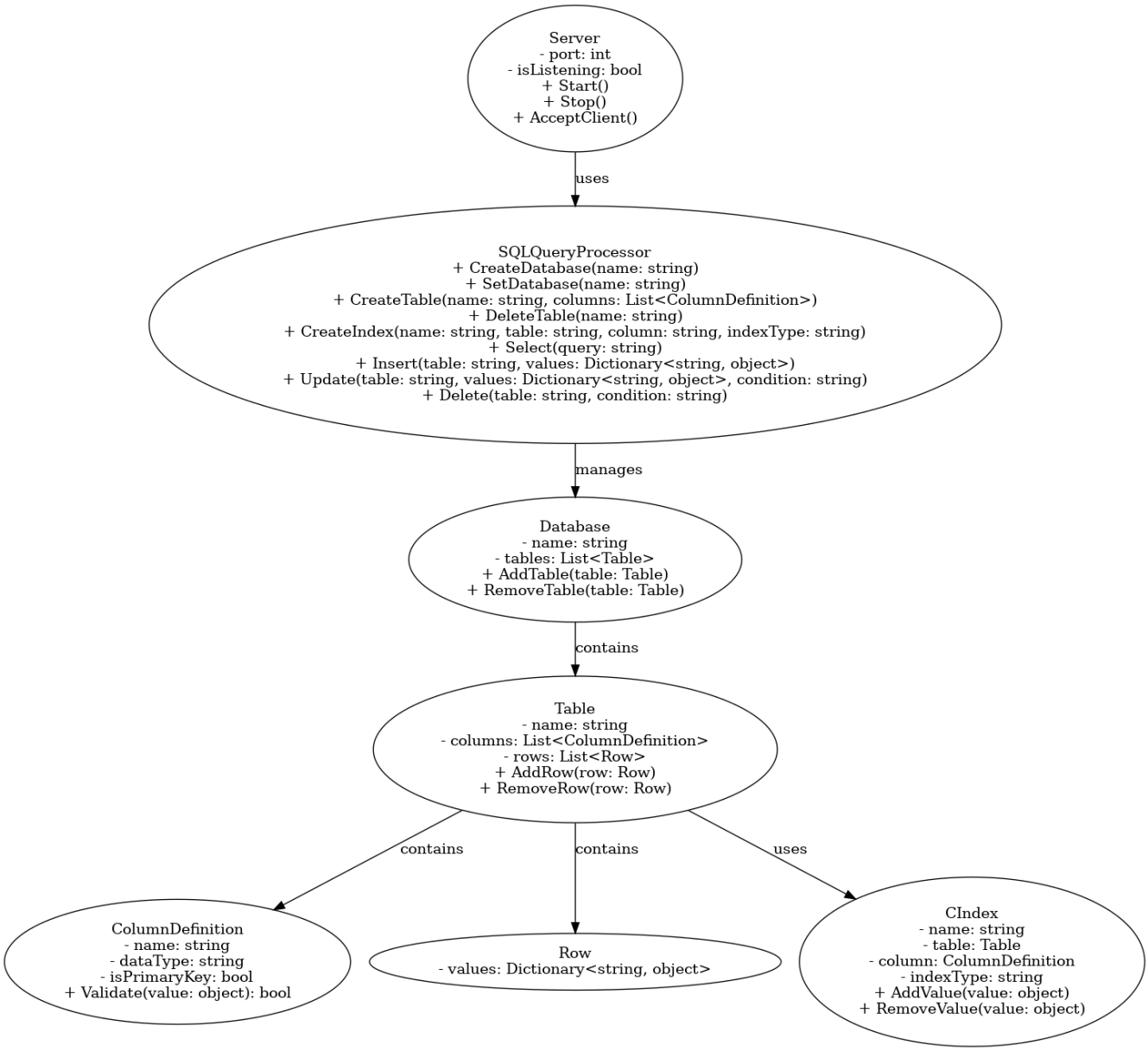
No se observa la implementación de la lógica de ordenamiento (Quicksort) mencionada en los requerimientos para las consultas SELECT. Además, complicaciones para implementar el LIKE y NOT.

### 2.4.3. Aspectos a mejorar:

Es necesario implementar la lógica de utilización de índices para optimizar las búsquedas, especialmente en las operaciones SELECT, UPDATE y DELETE. También se debe agregar la funcionalidad de ordenamiento para las consultas SELECT utilizando el algoritmo Quicksort como se especifica en los requerimientos.

# 3. Diseño General

## 3.1. Diagrama de Clases UML



### 3.1.1. Descripción de Clases para el Diagrama UML

- **Server:** Esta clase representa el servidor que maneja las conexiones TCP. Tiene atributos para el puerto y el estado de escucha. Sus métodos incluyen iniciar y detener el servidor, así como aceptar conexiones de clientes.
- **SQLQueryProcessor:** Esta clase se encarga de procesar las consultas SQL. Incluye métodos para crear bases de datos, establecer el contexto de la base de datos, y manejar operaciones CRUD (crear, leer, actualizar y eliminar) para tablas.
- **Database:** Representa una base de datos específica, que tiene un nombre y contiene una lista de tablas. Permite añadir y eliminar tablas de su colección.
- **Table:** Esta clase representa una tabla dentro de una base de datos. Tiene atributos para el nombre, definiciones de columnas y filas. También incluye métodos para añadir y eliminar filas.
- **ColumnDefinition:** Define la estructura de una columna en una tabla, incluyendo su nombre, tipo de datos y si es clave primaria. También incluye un método para validar los valores.
- **Row:** Representa una fila en una tabla, con un diccionario que almacena los valores de las columnas.
- **CIndex:** Esta clase representa un índice en una tabla, incluyendo el nombre del índice, la tabla a la que pertenece, la columna asociada y el tipo de índice. Tiene métodos para añadir y eliminar valores del índice.

## 3.2. Relaciones entre Clases

- Server se comunica con SQLQueryProcessor.
- SQLQueryProcessor utiliza Database, Table, y CIndex para gestionar las operaciones de base de datos.
- Database contiene múltiples Table.
- Table contiene múltiples ColumnDefinition y Row.
- CIndex está asociado con una Table y una ColumnDefinition