Instituto Tecnológico de Costa Rica

Escuela de Computadores

CE 1103: Algoritmos y Estructuras de Datos I

Proyecto Programado TinySQLDb

Estudiantes

Guillermo Sánchez Cedeño

Luis Castro Montenegro

Profesor

Leonardo Araya Martínez

Índice

1 INTRODUCCIÓN

En este segundo proyecto se le presenta al estudiantado una problemática que lo incita a indagar en un tema del cual desconoce, así pues, basándose en el conocido motor de administración y creación de bases de datos MySQL nace TinySQLDb.

Tiny busca imitar el funcionamiento de muchas plataformas de bases de datos relacionales, implementando numerosos comandos que funcionan como una conexión entre el usuario final y la complejidad de la administración de una base de datos. Tiny ofrece el guardado permanente de información mediante archivos binarios y el recuperamiento de esta con el uso de su cliente que mostrará el resultado de cada una de las consultas en consola.

A su vez algo que no se ve dentro de los motores de administración se implementa dentro de nuestro proyecto, se habla del sistema de los índices para el acceso inmediato de la información, creando un árbol de búsqueda rápida para ofrecer el menor tiempo de espera entre cada consulta al servidor.

2 BREVE DESCRIPCIÓN DEL PROBLEMA

Se presenta una problemática que se basa en el desarrollo de un sistema cliente servidor relacionado a la administración de bases de datos relacionales, usando como base el lenguaje de programación **C#** junto con **dotNet8**; se plantea una comunicación entre ambas partes utilizando sockets y mensajes con formato JSON que deberán ser decodificados tanto por parte del servidor como por el script del cliente.

2.1 EL SERVIDOR

TinySQLDb cuenta con un servidor asignado de forma local dentro de la red, el cual esperará por cualquier conexión entrante de un cliente, cuando reciba una, empezará a decodificar el mensaje, empezando por analizar la sentencia dentro de este, en busca de cualquier incongruencia de la sintaxis, luego será enviado al procesador de sentencias donde la lógica detrás de éstas será ejecutada, devolviendo una respuesta por el mismo canal abierto por el cliente.

2.2 EL CLIENTE

Nuestro sistema cuenta con un cliente, donde el usuario solo deberá crear un archivo *"script.tinysql"*, donde escribirá todas las sentencias (separadas por punto y coma) que desea que sean ejecutadas por el servidor. Una vez hecho esto el usuario deberá ejecutar el script de PowerShell incluyendo la ruta del script que creó con anterioridad, dirección ip y puerto asignado del servidor. Si ocurre algún error durante la ejecución de las sentencias, el usuario será notificado en consola.

3 DESCRIPCIÓN DE LA SOLUCIÓN

En esta sección para cada uno de los requerimientos planteados para este proyecto se brindará una descripción

3.1 REQUERIMIENTO 000

El cliente se basa en un script escrito en el lenguaje de programación de PowerShell, este debe de ser cargado por el usuario utilizando en la línea de comandos *". .\tiny-sql-client"* después deberá llamar a una de sus funciones llamada **Execute-MyQuery** de parámetros QueryFile, en donde se indica la ruta del script que contiene las sentencias a ser ejecutadas, los argumentos Puerto e IP que indican a donde debe de apuntar los mensajes a la hora de ser enviados.

En términos generales nuestro script abre el archivo que contiene las sentencias e itera en sus líneas, separando cada una de estas por medio de los puntos y comas. Por cada iteración al archivo se extrae la sentencia en ese momento para ser enviada al servidor y esperar una respuesta, (la ejecución de las sentencias no prosigue hasta conseguir una respuesta) habiéndole recibido la información de la consulta es mostrada en consola.

3.2 REQUERIMIENTO 001

El SystemCatalog se encuentra ubicado en la carpeta raíz C: en donde se alojan distintos archivos binarios de los cuales encontramos el SystemDabatases que almacena el nombre de cada una de las bases de datos existentes, el SystemTables que registra el nombre de las tablas para cada una de las bases de datos, el SystemColumns que para cada tabla almacena los tipos de datos de cada columna de las tablas y por último el SystemIndexes que guarda los índices creados para una tabla en específico.

Como ya se había mencionado antes, estos archivos están escritos y son leídos en binario utilizando las clases de C# llamadas BinaryWriter y BinaryReader que nos proporcionan una funcionalidad optimizada para guardar la información de nuestras bases de datos. Estos archivos ayudaran a acceder a los metadatos de la información de guardada en el sistema para ser procesada por el servidor y enviada al cliente.

3.3 REQUERIMIENTO 002

Las bases de datos se modelan como un directorio en la carpeta raíz del sistema, en donde se guardarán allí los archivos tabla, el servidor de TinySQLDb se encarga de crear esta carpeta con el nombre ingresado por el usuario.

3.4 REQUERIMIENTO 003

El contexto de todas las sentencias enviadas por el usuario se establece mediante el comando **"SET <DATABASE>"**, el servidor verifica que la base de datos exista mediante la capa de archivos, el System Catalog, que leyendo por todo el archivo binario

SystemDatabases busca el nombre ingresado por el usuario. Si se encuentra se devuelve un mensaje satisfactorio, de lo contrario se retorna un error.

3.5 REQUERIMIENTO 004

Una vez se selecciona una base de datos se pueden crear archivos que funcionan como tablas para almacenar la información. Al crear una tabla se debe actualizar el System Catalog, más específicamente el System Tables y System Columns, el primero almacena el nombre de la base y tabla, el segundo los dos datos anteriores y por cada columna su nombre, tipo y cualquier otra indicación.

El sistema identifica si la consulta tiene el siguiente formato **"CREATE TABLE <Nombre Tabla> AS (<columna> <tipo>, <columna> <tipo>)"**. Para esto se utiliza la expresión regular Regex definiendo un patrón para compararlo con consulta. Si es igual Regex permite acceder a los distintos valores según el formato.

3.6 REQUERIMIENTO 005

A la hora de eliminar una tabla el servidor tiene que validar dos puntos, empezando por verificar que la tabla exista en el sistema mediante la capa de archivos, también se debe validar que la tabla esté vacía, si no lo está, la operación no podrá llevarse a cabo. Una vez verificado, el servidor borrará el archivo tabla relacionado a la base de datos que el usuario indicó.

3.7 REQUERIMIENTO 006

Para la creación de los índices el sistema revisa la consulta y sigue el formato **"CREATE INDEX <nombre> ON <tabla>(<columna>) OF TYPE <tipo>"**, utilizando Regex el cual permite extraer los valores para los que se quiere crear el índice. Lo primero es actualizar el System Indexex para agregar el índice usando el nombre de la base, el nombre de la tabla, nombre del índice, columna y tipo. Tras guardar los datos en el archivo se procede a crear un árbol según el tipo, BTREE o BST. Este nuevo árbol se almacena en una lista estática que permite al sistema acceder a los índices creados. Al momento de seleccionar una base de datos el sistema revisa los índices creados para esa base y los crea.

3.8 REQUERIMIENTO 007

Para seleccionar datos específicos de una tabla, los archivos dentro de esta se modelan como una lista de listas, ósea una matriz, con esta se harán las comparaciones necesarias dentro de la cláusula WHERE, que funciona indicando una de las columnas de la tabla y comparándola con cualquier valor como por ejemplo **"WHERE ID == 1"**, esto devolverá la fila donde la columna ID sea igual a 1.

La cláusula ORDER BY nos ayudará a ordenar la información obtenida por la consulta siguiendo el tipo de dato de la columna ingresada por el usuario, por ejemplo, si se escoge una columna tipo VARCHAR, la información será mostrada en orden alfabético. También se

debe de escoger una dirección para este ordenamiento; todo esto se realiza utilizando el modelo de algoritmo llamado QuickSort, que resulta versátil para todo tipo de comparaciones y ordenamiento.

La información resultante de estas dos cláusulas (Ambas opcionales) es ingresada en una lista de diccionarios, donde las llaves de cada uno de estos serán el nombre de la columna y el valor, la información almacenada en la columna y fila correspondiente. Esta lista es codificada a un formato JSON y enviado como respuesta hacia el cliente que se encargará de formatear la tabla.

3.9 REQUERIMIENTO 008

La información almacena en una tabla puede ser actualizada mediante el comando **“UPDATE <tabla> SET <columna> = <nuevo valor> [WHERE condiciones]”**, utilizando Regex se valida si la consulta sigue este formato, el where es opcional. Una vez que se tienen los datos el sistema debe validar si para la base seleccionada existe la tabla y dentro de esta la columna. Se valida que el nuevo valor sea válido para el tipo de dato asociado a la columna, finalmente se valida si el where es válido.

Con las validaciones realizadas se usa el nombre de la base y la tabla para ir al folder donde se encuentra el archivo que contiene los datos, se lee todo el archivo para obtener sus datos y poder ejecutar la validación del where para así solo actualizar las líneas que cumplen las condiciones. Si la consulta no cuenta con un where se actualiza la columna entera. Una vez que se termina de escribir en el archivo se valida si hay índices asociados a la tabla y columna para actualizarlo.

3.10 REQUERIMIENTO 009

La información almacena en una tabla puede ser eliminada mediante el comando **“DELETE FROM <tabla> [WHERE condiciones]”**, utilizando Regex se valida si la consulta sigue este formato, el where es opcional. Una vez que se tienen los datos el sistema debe validar si para la base seleccionada existe la tabla y dentro de esta la columna. Se valida que el nuevo valor sea válido para el tipo de dato asociado a la columna, finalmente se valida si el where es válido.

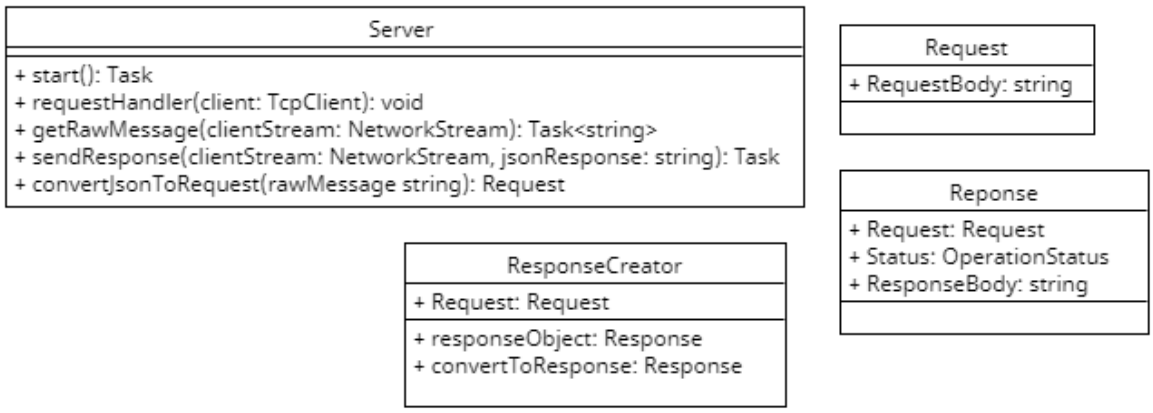
Con las validaciones realizadas se usa el nombre de la base y la tabla para ir al folder donde se encuentra el archivo que contiene los datos, se lee todo el archivo para obtener sus datos y poder ejecutar la validación del where para así solo actualizar las líneas que cumplen las condiciones. Si la consulta no cuenta con un where se elimina la tabla entera. Una vez que se termina de escribir en el archivo se valida si hay índices asociados a la tabla y columna para actualizarlo.

3.11 REQUERIMIENTO 010

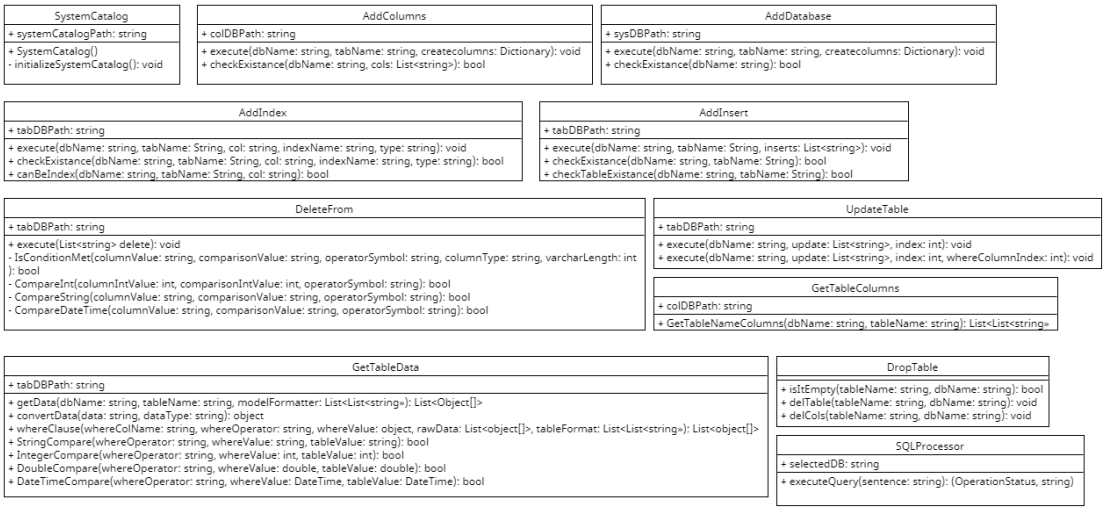
Con las tablas creadas se puede agregar datos a esta con el comando **“INSERT INTO <tabla> VALUES(<value>, <value>)”**. Primero se verifica que la tabla pertenece a la base de datos, luego que tenga la misma cantidad de valores a insertar que numero de columnas. Cada valor se compara contra su columna para validar el tipo, si es correcto entonces procede a guardar. Se una un patrón para obtener la tabla y los valores, luego cada uno de estos es separado.

DISEÑO GENERAL

3.12 TinySQLDb



3.13 StoreSytem



3.14 QueryProcessing