Distributed Graph Database

Andre Carlos Alvarez Cardenas
Universidad Católica San Pablo
Arequipa, Perú
Email: andre.alvarez@ucsp.edu.pe

Aarón Misash Apaza Coaquira
Universidad Católica San Pablo
Arequipa, Perú
Email: aaron.apaza@ucsp.edu.pe

Jorge Luis Huanca Mamani
Universidad Católica San Pablo
Arequipa, Perú
Email: jorge.huanca@ucsp.edu.pe

I. Introducción

Lo que se busca en este proyecto es hacer una Base de Datos Distribuida usando grafos. La forma que se desarrollará es mediante la técnica Client-Server, donde el servidor tendrá las tablas originales que serán actualizadas mediante la actividad del cliente. Por otro lado, el cliente tendrá copias de la base de datos que serán modificadas y tras pasar por algunos comandos y protocolos se actualiza en la base de datos del servidor. Las tecnologías utilizadas para este proyecto serán SQLite, lenguaje C++, TCP/IP, Visual Studio, Linux y Github.

II. Marco Teórico

Para entender a más detalle la funcionalidad de este proyecto, necesitamos conocimiento previo de los siguientes conceptos:

A. Base de Datos Distribuida

Una Base de Datos Distribuida es una base de datos que consiste de 2 o mas archivos ubicados en diferentes sitios, ya sea en la misma red o en redes completamente diferentes. Las partes de la base de datos se almacena en varias ubicaciones físicas. El procesamiento es distribuido entre varios nodos de la base de datos. Algunas ventajas de usar Base de Datos Distribuidas, son que son capaces de un desarrollo modular, lo que significa que los sistemas se pueden expandir agregando nuevas computadoras y datos locales al nuevo sitio y conectándolos al sistema distribuido sin interrupción.

B. SQLite

SQLite es un sistema para la gestión de bases de datos relacionales, contenida en una pequeña biblioteca escrita en C, que se enlaza con el programa principal, pasando a ser un proceso integrado del mismo.
SQLite tiene las siguientes características notables:

- **Multisistema** la biblioteca SQLite se incluye en los sistemas operativos más utilizados, como Windows, Linux, Android y los sistemas de Apple (iOS y macOS).
- **Open Source** no requiere del pago de una licencia
- **Multilenguaje**, dispone de diferentes API que permite trabajar con lenguajes de programación como C++, Python o PHP, entre otros.
- **Soporta múltiples tablas**, índices y vistas.
- **No necesita configuración**
- **Sencillez**, SQLite dispone de una API que es muy simple, por lo que su uso es muy fácil y no requiere de grandes conocimientos técnicos.
- **Autonomía**, no tiene dependencias externas.

C. Modelo Client-Server

Como dice en el nombre, este modelo consiste en 2 partes. El Cliente y el Servidor. Un servidor es una aplicación que ofrece un servicio a usuarios de Internet; un cliente es el que pide ese servicio. Una aplicación consta de una parte de servidor y una de cliente, que se pueden ejecutar en el mismo o en diferentes sistemas. El servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de una respuesta. Generalmente un servidor puede tratar múltiples peticiones(múltiples clientes) al mismo tiempo.

D. TCP/IP

Transmission Control Protocol/Internet Protocol(TCP/IP) es un conjunto de reglas estandarizadas que permiten a los equipos comunicarse en una red como Internet. TCP/IP incide mucho en la precisión y dispone de varios pasos para garantizar la correcta transmisión de los datos entre ambos equipos. Lo que hace TCP/IP es descomponer cada mensaje en paquetes que se vuelven a ensamblar en el otro extremo. Este protocolo tiene 5 capas que se encargan de crear un sistema estandarizado, sin que los distintos fabricantes de hardware y software tengan que gestionar la comunicación por su cuenta.

- Aplicación
- Transporte
- Red
- Enlace
- Física

III. Protocolo

Nuestro protocolo define los formatos de intercambio de paquetes de información para lograr la comunicación entre dos o mas clientes a través de una red distribuida.

1. **Create**: Al trabajar con el protocolo Create, se trendra 3 versiones. La primera es para crear un nodo nuevo.

La segunda es para crear un enlace entre 2 nodos ya existentes. Y la tercera sera para crear un atributo dentro de un nodo.

III-A. Crear Nodo

Para esto, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{C}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node}}_{variable}$

III-B. Crear Edge

Para crear un enlace entre 2 nodos, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id, 3 bytes para el tamaño de relación, 3 bytes para el tamaño del nombre del nodo final.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{CE}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node1}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{relation}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node2}}_{variable}$

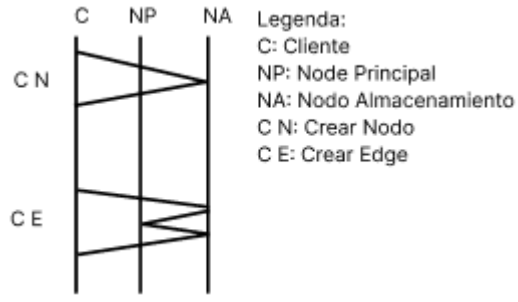
III-C. Crear Atributo

Para crear un Atributo en un nodo, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id, 3 bytes para el tamaño de atributo.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{CA}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node1}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{attribute}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{value}}_{variable}$

III-D. Diagrama de Create



2. **Read**: El protocolo Read, se creo para obtener información de un nodo o registro, se tendrá 3 versiones. La primera es para leer los atributos que están en un nodo. El segundo es para leer información ubicada en un nodo. Y el tercero es para leer la relaciones a una profundidad dada.

III-E. Read Attribute

Para leer los atributo que se tienen en un nodo, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño el tamaño, 1 byte para indicar que se información del nodo.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{RA}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node}}_{variable}$	$\underbrace{A}_{1byte}$

Nodos de Almacenamiento → Nodo Principal → Cliente

$\underbrace{A}_{1byte}$	$\underbrace{totalsize}_{3bytes}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{attribute}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{value}}_{variable}$

III-F. Read Information

Al querer obtener información de un nodo se aplica este protocolo. se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id, 1 byte para indicar que se busca la información del nodo.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{RI}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node}}_{variable}$	$\underbrace{i}_{1byte}$

Nodos de Almacenamiento → Nodo Principal

$\underbrace{i}_{1byte}$	$\underbrace{sizeAttributes}_{3bytes}$	$\underbrace{sizeRelations}_{3bytes}$

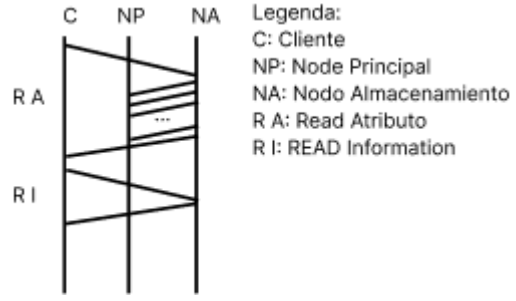
Nodo Principal → Cliente

$\underbrace{i}_{1byte}$	$\underbrace{sizeAttributes}_{3bytes}$	$\underbrace{sizeRelations}_{3bytes}$	$\underbrace{idNode}_{3bytes}$

III-G. Read Relation

???

III-H. Diagrama de Read



3. **Update**: La actualización de un nodo se creo para poder actualizar 3 cosas dentro del nodo. La primera es el mismo nombre del nodo. La segunda es para poder actualizar los enlaces que se tienen y la tercera es para poder actualizar los atributos que se tienen dentro del nodo.

III-I. Update Node

Este protocolo actualiza el nombre del nodo, para esto se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id y 3 byte para el tamaño del nuevo id.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{U}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{nodeName}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{newNodeName}}_{variable}$

III-J. Update Edge

Para actualizar el enlace entre 2 nodos, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id y 3 byte para el tamaño de la relación, 3 bytes para el tamaño del id del nodo objetivo, 3 bytes para el tamaño del id original, 3 byte para el tamaño de la relación, 3 bytes para el tamaño del id del nodo objetivo.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{UE}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node1}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{relation}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node2}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{newRelation}}_{variable}$

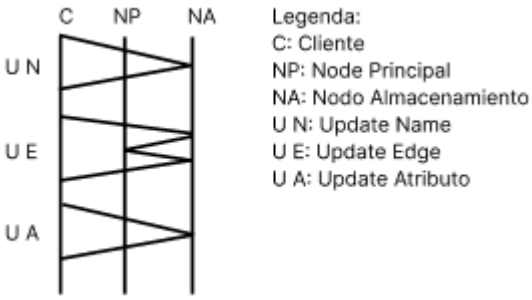
III-K. Update Attribute

Para actualizar los atributo que tiene un nodo, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id, 4 bytes para el nuevo atributo.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{UA}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{attribute}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{newValue}}_{variable}$

III-L. Diagrama de Update



4. **Delete**: Para eliminar un nodo se requiere solo el nombre del nodo, sin embargo si se quiere eliminar un atributo de este se debe ingresar el nombre de este igualmente para la eliminación de un enlace.

III-M. Delete Node

Al querer eliminar un nodo se utiliza este protocolo, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{D}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node}}_{variable}$

III-N. Delete Edge

Caundo se quiere eliminar un enlace se utilizara este protocolo, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id, 3 byte para el tamaño de la relación, 3 bytes para el tamaño del id del nodo objetivo.

Cliente → Nodo Principal → Nodos de Almacenamiento

$\underbrace{DE}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node1}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{relation}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{node2}}_{variable}$

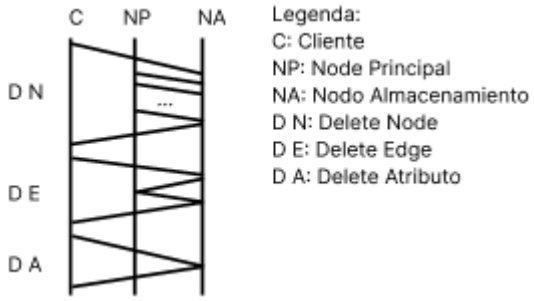
III-Ñ. Delete Attribute

Para eliminar un atributo, se utilizara un byte para indicar el tipo de comando, 3 bytes para el tamaño del id, 3 byte para el tamaño del atributo.

Cliente → Nodo Principal → Nodos de Almacenamiento

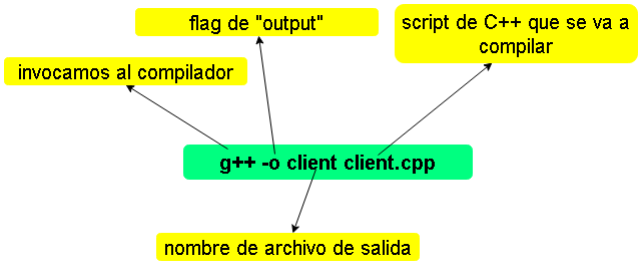
$\underbrace{DA}_{1byte}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{nodeName}}_{variable}$	$\underbrace{size}_{3bytes}$	$\underbrace{\text{attribute}}_{variable}$

III-O. Diagrama de Delete



IV. Comandos generales

Se utiliza el compilador de GCC del paquete build-essential disponible en los repositorios oficiales de las principales distribuciones linux (Arch, Ubuntu, OpenSuse, RedHat)
Para compilar el código generado y luego crear un ejecutable se debe correr el siguiente comando



Una vez compilado el archivo del cliente y el servidor se procede a ejecutar los binarios generados

IV-A. Comando de la clase

1. GDBU U educador RC Julio educador RC
2. GDBU D RC
3. GDBU D Julio educador RC
4. GDBU D Julio nacionalidad
5. GDBU U Julio Julius
6. GDBU R Julio R 5
7. GDBU R Julio A
8. GDBU R Julio I
9. GDBU C Julio
10. GDBU C Julio educador RC
11. GDBU C Julio CA email ucsp.pe

V. SQLite

Dado que SQLite se desarrolló pensando en simplificar el uso y los requisitos para gestionar una base de datos en una aplicación, poniendo el foco en la simplificación y el rendimiento.

Teniendo en cuenta lo siguiente:

- SQLite es muy sencillo de utilizar, ya que no utiliza una comunicación cliente-servidor para las consultas, ya que se comunica con un archivo que es la base de datos y que puede ser auto generado por la propia aplicación.
- El almacenamiento de una base de datos SQLite se realiza en un solo archivo y tiene una huella de código pequeña (ocupa poco espacio).

Para crear una nueva base de datos se debe correr el siguiente comando,

$$Sqlite3DatabaseName.db$$

previamente debemos haber instalado el paquete de $sqlite3$ disponible en los repositorios oficiales.

Como SQLite almacena los datos en un archivo, se enviara una copia de dicho archivo a los clientes cumpliendo con el propósito de cumplir la funcionalidad de la base de datos distribuida, brindando la posibilidad de correr los comandos nativos propios de SQLite para cumplir con la funcionalidad del CRUD.

VI. Tipo de Comunicación

En este proyecto, la comunicación se basa en 3 componentes. El cliente, el nodo principal y los *n* nodos de almacenamiento. Debido a esto, se explicara cada conexión en las siguientes secciones:

VI-A. Cliente

El Cliente tendrá conexión directa con el Nodo Principal al momento de hacer una consulta. Esta conexión durara mientras se obtiene la información medida o se hacen las modificación hechas por el cliente. Al momento de terminar, se corta la conexión pero esta listo para recibir una nueva conexión con un nuevo Request". En otras palabras la conexión entre el Cliente y el Nodo Principal es temporal.

VI-B. Nodo Principal

El Nodo Principal tiene conexión con el Cliente y con los n Nodos de Almacenamiento. Como mencionado en la sección anterior, la conexión entre el Nodo Principal y el Cliente es temporal. Y es la misma idea para la conexión entre el NP y NA. La conexión se da una vez que se ubico el NA correspondiente gracias a la función "HASH". Esta función se encargara de ubicar que nodo se tiene que acceder. Una vez ubicado, el NP se conecta en el NA que se ubico.

VI-C. Nodo Almacenamiento

Ultimo tenemos al Nodo de Almacenamiento. Este tiene una conexión temporal con el Nodo Principal que es basada en el resultado de la función "HASH". La conexión entre los NA dependen también del NP. Ya que si se quiere acceder o modificar información de un nodo vecino, se tiene que pasar por el NP para poder obtener la ubicación, identificar cuales nodos están vinculados y se actualiza los cambios en todos los nodos que tengan alguna vinculación.