Практика 2 (NoSQL DBMS Network)

Тема: Разработка сетевого интерфейса для документно-ориентированной СУБД

Цель: Получить практические навыки в разработке сетевых приложений, реализации клиент-серверной архитектуры, сериализации данных для сетевой передачи и организации многопользовательского доступа к системе.

---

1. Задание

Необходимо модифицировать существующую СУБД из предыдущей лабораторной работы, добавив возможность работы по сети. Система должна состоять из двух компонентов: **сервера БД** и **клиента**. Сервер обеспечивает многопользовательский доступ к данным, а клиент предоставляет интерфейс для взаимодействия с сервером.

---

2. Функциональные требования

2.1. Архитектура системы

Серверная часть:

- Запускается как демон и ожидает подключений
- Обрабатывает множественные клиентские подключения
- Управляет конкурентным доступом к данным
- Сохраняет целостность данных при параллельных операциях

Клиентская часть:

- Подключается к серверу по указанному адресу и порту
- Отправляет команды и получает результаты
- Предоставляет пользовательский интерфейс для работы с БД

2.2. Сетевой протокол

Формат запроса:

```
{
  "database": "my_database",  // Название базы данных
  "operation": "insert|find|delete",  // Операция
  "data": [ ... ],  // Данные, нужны для операции вставки, список документов
  "query": { ... }  // Запрос
}
```

Формат ответа:

```
{
  "status": "success|error",  // Статус ответа
  "message": "Fetched 10 docs from my_db",  // Доп. информация + ошибки
  "data": [ ... ],  // Список отобранных документов
  "count": 0  // Количество отобранных документов
}
```

2.3. Команды клиента

Подключение к серверу:

```
./db_client --host localhost --port 8080 --database my_database
```

Выполнение операций:

```
> INSERT users {'name': 'Alice', 'age': 25}
> FIND users {'age': {'$gt': 20}}
> DELETE users {'name': 'Alice'}
```

2.4. Многопользовательский доступ

- Поддержка одновременной работы нескольких клиентов
- Блокировки на уровне базы данных
- Очередь запросов на модификацию данных

---

3. Техническая реализация

3.1. Серверная часть

Архитектура сервера:

- **Основной поток:** Принимает новые подключения
- **Рабочие потоки/процессы:** Обрабатывают запросы клиентов
- **Менеджер блокировок:** Управляет доступом к базам данных
- **Пул соединений:** Управление сетевыми соединениями

Обработка запросов:

1. Принятие и парсинг JSON запроса
2. Проверка корректности запроса
3. Получение блокировки (для операций записи)
4. Выполнение операции с данными
5. Формирование и отправка ответа
6. Снятие блокировки

3.2. Клиентская часть

Интерактивный режим:

- REPL (Read-Eval-Print Loop) для ввода команд

Режим одного запроса:

```
./db_client --host localhost --port 8080 --database my_database

> INSERT users {'name': 'Alice', 'age': 25}
> FIND users {'age': {'$gt': 20}}
> DELETE users {'name': 'Alice'}
```

3.3. Механизм блокировок

Реализация:

- Использование мьютексов для каждой базы данных
- Захват мьютекса на начало выполнение запроса, отпускание мьютекса после выполнения запроса
- Таймауты для избежание deadlock'ов

3.4. Обработка ошибок

Типовые ошибки:

- Сетевые проблемы (разрыв соединения, таймауты)
- Некорректные запросы (синтаксические ошибки)

4. Обязательные требования

4.1. Сетевое взаимодействие

- Реализация TCP-сервера с использованием сокетов
- Сериализация/десериализация JSON сообщений
- Обработка разрывов соединения
- Таймауты на операции

4.2. Конкурентность

- Обработка минимум 5 одновременных клиентов
- Блокировки через мьютекс для операций модификации данных
- Потокобезопасные операции с базой данных

---

5. Рекомендации по выполнению

1. **Этап 1:** Реализовать базовый TCP-сервер и простой клиент
2. **Этап 2:** Разработать формат сетевых сообщений
3. **Этап 3:** Интегрировать сетевое API с существующей СУБД
4. **Этап 4:** Реализовать механизм блокировок и конкурентность
5. **Этап 5:** Добавить интерактивный режим клиента

Тестирование:

- Проверка работы нескольких клиентов одновременно
- Тестирование обработки сетевых ошибок
- Проверка целостности данных при параллельных операциях
- Нагрузочное тестирование

Материалы:

- https://ru.wikipedia.org/wiki/TCP/IP
- https://en.wikipedia.org/wiki/Network_socket
- https://en.wikipedia.org/wiki/Lock_(computer_science)
- https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)