

Задание 13

Условие

Разработать асинхронный сервер выполняющий операции пересечения и симметрической разницы над множествами.

Внутренняя реализация должна предоставить возможность работать с двумя таблицами идентичной и фиксированной структуры. Одна таблица имеет название A, вторая - B и общую структуру:

```
{
    int id;
    std::string name;
}
```

id является первичным ключом и не может дублироваться в одной таблице.

Для содержимого таблицы A:

```
id | name
---+-----
0  | lean
1  | sweater
2  | frank
3  | violation
4  | quality
5  | precision
```

и содержимого таблицы B:

```
id | name
---+-----
3  | proposal
4  | example
5  | lake
6  | flour
7  | wonder
8  | selection
```

Необходимо уметь формировать пересечение данных из двух таблиц в виде коллекции:

```
id | A          | B
---+-----+-----
3  | violation  | proposal
4  | quality    | example
5  | precision  | lake
```

Необходимо уметь формировать симметрическую разность данных из двух таблиц в виде коллекции:

id	A	B
0	lean	
1	sweater	
2	frank	
6		flour
7		wonder
8		selection

Строки должны следовать в порядке возрастания поля *id*.

Для пополнения коллекции и выполнения операций над множествами необходимо реализовать следующий протокол:

```
INSERT table id name
TRUNCATE table
INTERSECTION
SYMMETRIC_DIFFERENCE
```

Протокол

Команды отправляются по сети в слушающий порт сервера. Признаком завершения команды является символ `\n`. В ответ на любую команду сервер присылает последовательность `OK\n` в случае успеха и `ERR message\n` в случае ошибки с расшифровкой.

Для формирования таблиц из условия будут выполнены последовательно следующие команды:

```
> INSERT A 0 lean
< OK
> INSERT A 0 understand
< ERR duplicate 0
> INSERT A 1 sweater
< OK
> INSERT A 2 frank
< OK
...
> INSERT B 6 flour
< OK
> INSERT B 7 wonder
< OK
> INSERT B 8 selection
< OK
```

В качестве разделителя используется строго один символ пробела.

Команды TRUNCATE A и TRUNCATE B очищают соответствующие таблицы.

```
> TRUNCATE A
< OK
```

INTERSECTION и SYMMETRIC_DIFFERENCE имеют идентичный формат вывода

```
> INTERSECTION
< 3,violation,proposal
< 4,quality,example
< 5,precision,lake
< OK
```

и

```
> SYMMETRIC_DIFFERENCE
< 0,lean,
< 1,sweater,
< 2,frank,
< 6,,flour
< 7,,wonder
< 8,,selection
< OK
```

Порядок запуска:

```
# join_server <port>
```

где

- *port* – номер tcp порта для входящих соединений. Соединения должны обслуживаться с произвольных интерфейсов.

Требования к реализации

Результатом работы должен стать исполняемый файл `join_server` и находиться в пакете `join_server`.

Специальных усилий по демонизации сервера выполнять не следует, это значит, что сразу после запуска сервер не возвращает управление до принудительного завершения.

Результат работы должен быть опубликован на bintray.

Проверка

Задание считается выполненным успешно, если после установки пакета и запуска с тестовыми данными вывод соответствует ожиданию.

Будет отмечена оценка сложности добавления новых записей в таблицы, очистки и операций над множествами. Отсутствие блокировки при длительной отправке результирующего блока клиенту. Отсутствие блокировки при длительном выполнении операций над множествами. Устойчивость к выполнению операций INSERT во время работы операций INTERSECTION и SYMMETRIC_DIFFERENCE или отправки результата этих операций.