Задачи для 2 семестра 2 курса

Задачи на взаимодействие клиент–сервер.

Требуется реализовать сетевое взаимодействие между сервером, моделирующем базу данных, и несколькими клиентами, выполняющими запросы к серверу.

Принципы сетевого программирования и примеры реализаций UDP и TCP клиентов и серверов будут обсуждаться на лекциях и также должны закрепляться на семинарах. Я постараюсь регулярно рассылать информацию о темах, рассмотренных на текущих лекциях.

Для контроля за процессом работы в семестре нужно обозначить контрольные сроки по заданиям и их отдельным этапам. Также предполагается провести пару контрольных работ на простейшие клиент-серверные реализации как тренировку зачетной процедуры.

С этого года наполовину сокращен лекционный курс. Поэтому теоретический материал скорее всего не будет централизовано выноситься на зачет. Возможно, я сформулирую некоторое количество вопросов, которые можно спрашивать в рабочем порядке, но дать какие либо рекомендации сейчас я не готов.

Задание 1 предполагает простое взаимодействие клиента с сервером в “режиме чтения”. Это означает, что сервер при запуске загружает свою базу данных из заданного файла, а клиенты обращаются к серверу только для получения информации по своим запросам. При этом сама логика взаимодействия предполагает “однократный” принцип работы клиента. То есть соединение создается отдельно под каждый конкретный запрос и разрывается после его обслуживания.

Обычно результатом запроса является несколько записей, либо таблица, построенная на основе нескольких записей. Эти данные должны быть отсортированы клиентом в зависимости от сути конкретной задачи, например, по алфавиту, по возрастанию, по датам и т.п. (обсуждается в каждом конкретном случае).

Сроки и этапы

— генерация случайного массива исходных данных для тестирования — 1 неделя

— реализация базы данных без сетевой части (загрузка, функции поиска, функции добавления, удаления, изменения) — 2 недели

— реализация языка запросов к базе — 2 недели

— реализация сетевого взаимодействия — 2 недели

Задание 2 предполагает развитие задания 1 в направлении пакетного режима работы и возможности редактирования базы данных со стороны клиента. Редактирование базы предполагает изменение существующей записи, добавление и удаление записей. При этом следует реализовать систему взаимоисключения, так чтобы запись, редактируемая одним клиентом, не могла быть изменена другим клиентом, а также чтобы редактирование базы данных не препятствовало обработке запросов от других клиентов. Конкретные операции редактирования записей обсуждаются в рабочем порядке для каждой конкретной задачи. Пакетный режим означает, что клиент может автономно выполнить серию запросов, записанных, например, в текстовом файле, и обработать полученные ответы, сохранив их также в каких-то файлах.

Сроки и этапы

— реализация механизма блокировки записей базы при выполнении запросов — 1 неделя

— реализация пекетного режима работы клиента по сценарию из файла — 2 недели

Контрольная работа — простейшее взаимодействие клиента и сервера путем многогратных запросов и ответов. (Варианты заданий будут позднее). Как пример задания — сервер “загадывает” целое число, а клиент пытается его “отгадать” методом деления пополам (сервер отвечает больше или меньше кго число, чем то, которое предложил клиент).

Задание 3 предполагает активное взаимодействие нескольких серверов и клиентов. Имеется в виду перенаправление запросов, регистрация клиентов на сервере, рассылка сообщений, моделирование почты, формума, чата и т.п.

Сроки и этапы

— реализация регистрации и авторизации клиентов на сервере — 2 недели

— реализация распределенной базы данных — 2 недели

Конкретные задания (стратегии обработки запросов) будут определены позднее.

Контрольная работа — простейшее взаимодействие нескольких клиентов и серверов по типу почты, формума, чата, DNS и т.п. (Варианты заданий будут позднее).

Итого с контрольными работами получается как раз около 16 недель. В действительности, все сделать в полном объеме вряд ли удастся, но можно хотя бы попытаться.

База данных на стороне сервера представляет собой простой массив записей, соответствующих тематике задачи. При запуске сервер считывает записи базы данных из текстового файла, где также для простоты может быть заранее указано общее количество записей. Все операции поиска выполяются для простоты последовательным просмотром данного массива. Использование более сложных структур для хранения данных, как правило, не предполагается, но может быть добавлено по желанию в отдельных случаях и в отдельных задачах. На сервере должа быть предусмотрена операция сохранения его базы данных в файл в аналогичном текстовом виде для последующего использования.

Ориентировочные параметры для тестирования:
объем базы данных — несколько сотен тысяч записей;
количество клиентов в тесте — до нескольких десятков;
количество запросов от одного клиента — до нескольких сотен.

Варианты 1 задания смотри на след. стр.

Варианты баз данных (задание 1)

Также можно добавить по желанию любые свои “темы” и структуры

---

(1) **S1 Студент.**

Запись о студенте содержит 4 поля:

```
struct { char name[64]; int group; double raiting; char phone[16];};
```

Запрос:

```
select name=[min,max] group=[min,max] rating=[min,max]
```

должен возвращать все записи, удовлетворяющие указанным критериям.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(2) **S2 Студент.**

Запись о студенте содержит 4 поля:

```
struct { char name[64]; int group; int marks[30]; };
```

для простоты считаем, что студент за время обучения сдает 30 экзаменов.

Запросы:

```
select name=[min,max] group=[min,max] rating=[min,max]
select all exam=N mark=[min,max]
```

должны возвращать все записи, удовлетворяющие указанным критериям.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(3) **D1 Ежедневник.**

Запись в ежедневнике состоит из трех полей:

```
struct { char date[11]; double time; char *event; };
```

дата гггг.мм.дд, время события, описание события в виде строки производьной длины.

Запрос:

```
select date=[min,max] time=[min,max]
```

должен возвращать все строки `event`, удовлетворяющие указанным критериям.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(4) **D2 Ежедневник.**

Запись в ежедневнике состоит из трех полей:

```
struct { char date[11]; double time; char *event; };
```

дата гггг.мм.дд, время события, описание события в виде строки произвольной длины.

Запрос:

```
select date=[min,max] in_event=[строка]
```

должен возвращать полностью все записи, в которых указанная строка содержится в строке `event` и дата соответствует запросу.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(5) **R1 Расписание.**

Запись в расписании лекций состоит из 5 полей:

```
struct { int day; int pair; int room; char *teacher; char *course; };
```

день недели 1–7, учебная пара 1–6, номер аудитории, преподаватель, название курса.

Запросы:

```
select teacher=[строка]
select subject=[строка]
```

должны составлять таблицу расписания для конкретного преподавателя или лекционного курса.

---

(6) **R2 Расписание.**

Запись в расписании состоит из 6 полей:

```
struct { int day; int pair; int room; int group; char *teacher; char *course; };
```

день недели 1–7, учебная пара 1–6, номер аудитории, группа, преподаватель, название курса.

Запросы:

```
select day=X group-pair
select day=X room-pair
```

должны для указанного дня недели составлять таблицу расписания, у которой строки соответствуют номеру пары, а столбцы — группе или номеру аудитории. Остальная информация записывается в ячейки таблицы.

(7) **L1 Библиотека.**

Запись в каталоге библиотеки журналов состоит из 5 полей:

`struct { char journal[64]; int year, number; char author[32]; char article[128];};`

Запрос:

`select journal=[строка] year=[min,max] number=[min,max]`

должен предоставлять списокк статей (содержание) указанного журнала за указанный период времени.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(8) **L2 Библиотека.**

Запись в каталоге библиотеки журналов состоит из 5 полей:

`struct { char journal[64]; int year, number; char author[32]; char article[128];};`

Запрос:

`select author=[строка] year=[min,max] number=[min,max]`

должен предоставлять списокк статей указанного автора за указанный период времени.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(9) **L3 Библиотека.**

Запись в каталоге библиотеки журналов состоит из 5 полей:

`struct { char journal[64]; int year; int number; char author[32]; char *article;};`

Запрос:

`select in_title=[строка] year=[min,max] number=[min,max]`

должен предоставлять списокк статей за указанный период времени, в названии которых встречается указанная строка.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(10) **B1 Биллинг.**

Запись в базе данных оплаты мобильной связи состоит из 4 полей:

`struct { char phone[12]; int service; char datetime[20]; double sum; };`

данный телефон обслуживался данным сервисом (условный номер — 1,2,3 ...) в данный момент гггг.мм.дд.чч.мм.сс, начислено sum рублей.

Запрос:

`select phone=[строка] period=[min,max]`

должен предоставлять сводку суммарных начислений за указанный период по каждому типу сервиса.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(11) **B2 Биллинг.**

Запись в базе данных оплаты мобильной связи состоит из 4 полей:

`struct { char phone[12]; int service; char datetime[20]; double sum; };`

данный телефон обслуживался данным сервисом (условный номер — 1,2,3 ... разговор, SMS, роуминг и пр.) в данный момент гггг.мм.дд.чч.мм.сс, начислено sum рублей.

Запрос:

`select service=X period=[min,max]`

должен предоставлять список телефонов с их начисленными суммами за указанный период по указанному типу сервиса.

---

(12) **N1 СМИ.**

Запись в базе электронного СМИ содержит записи из 3 полей:

`struct { char *keywords; char *article; char datetime[20]; };`

строка с ключевыми словами, текст статьи, дата-время в некотором наглядном формате.

Запрос:

`select keywords=[строка] period=[min,max]`

должен предоставлять список статей из указанного диапазона дат, в ключевых словах которых встречается хотя бы одно слово из keywords.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

(13) **N2 СМИ.**

Запись в базе электронного СМИ содержит записи из 3 полей:

```
struct { char *keywords; char *article; char datetime[20]; };
```

строка с ключевыми словами, текст статьи, дата-время в некотором наглядном формате.

Запрос:

```
select keywords=[строка] period=[min,max]
```

должен предоставлять список статей из указанного диапазона дат, в ключевых словах которых встречается все слова из keywords.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(14) **F1 Банк.**

Запись в базе банка содержит записи из полей:

```
struct { int account_id; double sum; int currency; char date[12]; };
```

номер счета, сумма на счете, тип валюты, дата открытия счета. Кроме этого банк имеет таблицу коэффициентов для конвертации валют.

Запрос:

```
select sum=[min,max] period=[min,max] currency=X
```

должен предоставить все счета и их суммы в пересчете на указанный тип валюты, открытые за указанный период.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(15) **M1 Магазин.**

Запись в базе интернет-магазина содержит записи из полей:

```
struct { int user_id; char *item; double price; char date[12]; };
```

идентификатор покупателяб название товара, цена покупки, дата.

Запрос:

```
select user=X period=[min,max]
```

должен предоставить список товаров, купленных данным покупателем за указанный период с указанием цены каждого товара и суммарной цены.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(16) **M2 Магазин.**

Запись в базе интернет-магазина содержит записи из полей:

```
struct { int user_id; char *item; double price; char date[12]; };
```

идентификатор покупателя, название товара, цена покупки, дата.

Запрос:

```
select item=[строка] period=[min,max]
```

должен предоставить список товаров, в названии коорых встречается указанная строка, со списком покупателей, купивших эти товары за указанный период с указанием цены товаров.

Отдельные поля могут отсутствовать в запросе. Тогда они не проверяются при выборке.

---

(17) **M3 Магазин.**

Запись в базе интернет-магазина содержит записи из полей:

```
struct { int user_id; char *item; double price; char date[12]; };
```

идентификатор покупателя, название товара, цена покупки, дата.

Запрос:

```
select period=[min,max]
```

должен выставить "счет" каждому покупателю, делавшему покупки за указанный период, т.е. для каждого покупателя указать набор товаров с их ценами, а также суммарную цену.

Варианты редактирования баз данных (задание 2)

Редактирование существующих записей, добавление новых записей, удаление записей. Сохранение текущей базы в файле.

Операции изменения базы должны поддерживать взаимоисключение. То есть несколько клиентов не могут одновременно редактировать одну и ту же запись.

Для пакетного редактирования базы должен быть разработан и реализован соответствующий набор команд-запросов к серверу.