Курсовая работа (проект) по операционным системам и архитектуре компьютера (2023-2024 уч. г.)

0. На языке программирования C++ (стандарт C++14 и выше) реализуйте приложение, позволяющее выполнять операции над коллекциями данных заданных типов (типы обслуживаемых объектов данных определите самостоятельно) и контекстами их хранения (коллекциями данных).

Коллекция данных описывается набором строковых параметров (набор параметров однозначно идентифицирует коллекцию данных):

- название пула схем данных, хранящего схемы данных;
- название схемы данных, хранящей коллекции данных;
- название коллекции данных.

Коллекции данных, схемы данных и пулы схем данных представляют собой ассоциативный контейнер вида *B-tree*, в котором каждый объект данных соответствует некоторому уникальному ключу. Для ассоциативного контейнера необходимо вынести интерфейсную часть (в виде абстрактного класса C++) и реализовать этот интерфейс. Взаимодействие с коллекцией объектов происходит посредством выполнения одной из операций над ней:

- добавление новой записи по ключу;
- чтение записи по её ключу;
- чтение набора записей с ключами из диапазона $[minbound \ldots maxbound]$;
- обновление данных для записи по ключу;
- удаление существующей записи по ключу.

Во время работы приложения возможно выполнение также следующих операций:

- добавление/удаление пулов данных;
- добавление/удаление схем данных для заданного пула данных;
- добавление/удаление коллекций данных для заданной схемы данных заданного пула данных.

Поток команд, выполняемых в рамках работы приложения, поступает из файла, путь к которому подаётся в качестве аргумента командной строки. Формат команд в файле определите самостоятельно.

Приложение должно работать в двух режимах:

- структуры и данные размещаются в оперативной памяти приложения (in-memory cache);
- структуры и данные размещаются в файловой системе.

Выбор режима работы обеспечивается при помощи аргументов командной строки на этапе запуска приложения.

Потеря данных в системе хранения недопустима.

Дополнительные задания:

Перед описанием задания в квадратных скобках указываются номера заданий, которые необходимо выполнить для выполнения текущего. Частичная (неполная) реализация дополнительных заданий **не допускается** (за исключением заданий 6, 7).

1. [0] Реализуйте выполнение команд, вводимых пользователем в интерактивном диалоге. Пользователь при этом может вводить конкретные команды (формат ввода определите самостоятельно) и подавать на вход файлы с потоком команд.
2. [0] Реализуйте механизм персистентности, позволяющий выполнять запросы к данным в рамках коллекции данных на заданный момент времени (дата и время, для которых нужно вернуть актуальную версию данных, передаётся как параметр). Для реализации используйте поведенческие паттерны проектирования “Команда” и “Цепочка обязанностей”.
3. [0] Реализуйте механизм вторичного индексирования, позволяющий выполнять эффективный поиск по различным отношениям порядка на пространстве данных (дублирование объектов данных при этом запрещается). Обеспечьте поиск при помощи указания ключа отношения порядка (в виде строки, подаваемой как параметр поиска).
4. [0] Обеспечьте хранение объектов строк, размещённых в объектах данных, на основе структурного паттерна проектирования “Приспособленец”. Дублирования объектов строк для разных объектов (независимо от контекста хранения) при этом запрещены. Доступ к строковому пулу обеспечьте на основе порождающего паттерна проектирования “Одиночка”.
5. [0] Для режима in-memory cache реализуйте механизмы сохранения состояния системы хранения данных в файловую систему и восстановления состояния системы хранения данных из файловой системы.
6. [0] Добавьте три реализации ассоциативных контейнеров, репрезентирующих пулы схем данных, схемы данных и коллекции данных: B^+-дерево, B^*-дерево, B^{*+}-дерево. Обеспечьте возможность конфигурирования типа ассоциативного контейнера при создании.
7. [0] Реализуйте возможность кастомизации (для заданной коллекции данных) аллокаторов для размещения объектов данных: глобальные операторы new и delete; первый + лучший + худший подходящий + освобождение в рассортированном списке; первый + лучший + худший подходящий + освобождение с дескрипторами границ, первый + лучший + худший подходящий + система двойников; первый + лучший + худший подходящий + аллокатор на красно-чёрном дереве.
8. [0] Реализуйте функционал приложения в виде сервера, запросы на который поступают из клиентских приложений. При этом взаимодействие клиентских приложений с серверным должно быть реализовано либо посредством средств межпроцессного взаимодействия, либо посредством средств сетевого взаимодействия.
9. [0, 8] Реализуйте комплекс серверных приложений (далее: кластер), одно из которых (далее: entrypoint) обрабатывает входящие пользовательские запросы от клиентского приложения и делегирует их остальным серверам (далее: storage), а storage-сервера хранят данные. Количество storage-серверов может быть произвольным; данные должны быть распределены по storage-серверам примерно равномерно (по возможности) в произвольный момент времени; во время работы комплекса приложений количество storage-серверов может меняться посредством выполнения запросов на создание/удаление серверов из отдельного приложения.

10. [0, 8] Обеспечьте механизм сердцебиения (heartbeat) для проверки работоспособности сервера. При отсутствии ответа на heartbeat-запрос, целевой сервер должен быть подвергнут перезапуску.
11. [0, 8, 9] Обеспечьте горизонтальное шардирование данных в системе хранения.
12. [0, 8, 9] Обеспечьте вертикальное шардирование данных в системе хранения.
13. [0, 8] Реализуйте механизм асинхронной обработки запросов (результатом запроса на выполнение операции должен являться идентификатор запроса (используйте формат GUID v4), по которому впоследствии должна иметься возможность получения результатов запроса).
14. [0, 8, 9] Реализуйте децентрализованную систему обработки запросов (любой запрос может быть отправлен на любой из узлов кластера; результат асинхронного запроса также может быть получен из любого узла кластера).
15. [0] Реализуйте серверное приложение, собирающее логи клиентской (и, если есть, серверной) части приложения в файловые потоки вывода. Конфигурирование серверного логгера обеспечьте на основе файла со структурой JSON.

Для получения положительной (3 и выше) оценки за курсовую работу необходимо подготовить и сдать на кафедру пояснительную записку. Требования к оформлению пояснительной записки указаны в приложении 1. Во время защиты курсовой работы необходимо уметь ориентироваться в коде, демонстрировать работу реализованного комплекса приложений, быть готовым отвечать на вопросы по языку программирования C++ и по алгоритмам и структурам данных.
Оценивается **только** финальная версия реализованного комплекса приложений.

Приложение 1. Требования к оформлению пояснительной записки.

В основной части пояснительной записки необходимо для каждого реализованного задания описать алгоритм решения задания, использованные средства языка программирования C++, а также использованные при разработке алгоритмы и структуры данных.

Структура пояснительной записки:

- Титульный лист
- Содержание
- Введение
- Основная часть
- Вывод
- Список использованных источников
- Приложения

Оформление пояснительной записки:

- Поля: левое 20мм, остальные 15мм
- Нумерация страниц: начиная с титульного листа, индексация инкрементальная начиная с 1, по центру страницы; на титульном листе номер страницы не указывается
- Заголовки и подзаголовки разделов: шрифт Times New Roman 16pt, междустрочный интервал 1.5pt, выравнивание по левому краю
- Основной текст: шрифт Times New Roman 14pt, междустрочный интервал 1.15pt, выравнивание по ширине; абзацные отступы
- Рисунки: выравнивание по центру; под рисунком должна находиться подпись в формате

  Рисунок #. <Описание рисунка>

  , где # - номер рисунка при сквозной нумерации рисунков по всей пояснительной записке, индексация инкрементальная начиная с 1. Оформление подписи к рисунку: шрифт Times New Roman 12pt, курсивный, междустрочный интервал 1pt, выравнивание по центру; рисунок и подпись к нему должны находиться на одной странице
- Таблицы: Выравнивание по центру; над таблицей должна находиться подпись в формате

  Таблица #. <Описание таблицы>

  , где # - номер таблицы при сквозной нумерации таблиц по всей пояснительной записке, индексация инкрементальная начиная с 1. Оформление подписи к таблице: шрифт Times New Roman 12pt, курсивный, междустрочный интервал 1pt, выравнивание по левому краю; таблица и подпись к ней должны находиться на одной странице
- Листинги: Шрифт Consolas 12pt, междустрочный интервал 1pt, выравнивание по левому краю; над листингом должна находиться подпись в формате

  Листинг #. <Описание листинга>

  , где # - номер листинга при сквозной нумерации листингов по всей пояснительной записке, индексация инкрементальная начиная с 1. Оформление подписи к листингу: шрифт Times New Roman 12pt, курсивный, междустрочный интервал 1pt, выравнивание по левому краю; листинг и подпись к нему должны находиться на одной странице
- Список использованных источников: оформление по ГОСТ 7.0.100-2018