**Предметная область**

Предметной областью является база данных студентов университета, которая хранит следующую информацию:

* **ID (int)** — уникальный идентификатор студента.
* **Name (std::string)** — имя студента.
* **Age (int)** — возраст студента.
* **Department (std::string)** — факультет/департамент студента.
* **Is Full-Time (bool)** — форма обучения:
  + true (1) — дневная форма.
  + false (0) — заочная форма.

База данных реализована в виде текстового файла с разделителем ;. Каждая запись представляет строку, содержащую поля записи, разделённые указанным символом.

**Функциональность базы данных**

* **Работа с базами данных:**
  + Создание новой базы данных с проверкой на существующее имя.
  + Открытие существующей базы данных.
  + Сохранение текущей базы данных.
  + Очистка текущей базы данных.
  + Удаление базы данных.
  + Просмотр текущей открытой базы данных.
* **Работа с записями:**
  + Добавление новой записи (с проверкой уникальности по ключевому полю ID).
  + Удаление записи:
    - По ключевому полю (ID).
    - По любому неключевому полю (Name, Department, Is Full-Time).
  + Поиск записей:
    - По ключевому полю (ID).
    - По любому неключевому полю (Name, Department, Is Full-Time).
  + Вывод всех записей на экран.
* **Резервное копирование и экспорт:**
  + Создание резервной копии с проверкой на существующее имя.
  + Восстановление из резервной копии.
  + Экспорт в CSV-формат с проверкой на существующее имя.
* **Сообщения об ошибках:**
  + Если искомая запись не найдена, пользователю выводится соответствующее сообщение.
  + Если запись с данным ключом уже существует, добавление блокируется.

**Анализ сложности и эффективности алгоритмов**

**1. Добавление записи**

* **Алгоритм:**
  + Проверяется уникальность ключевого поля (ID) через std::unordered\_map.
  + Индексы для Name и Department обновляются с помощью std::unordered\_map и std::set.
* **Сложность:**
  + Проверка уникальности: **O(1)**.
  + Обновление индексов: **O(log k)** (где k — количество записей в конкретном индексе).
  + Итоговая сложность: **O(log k)**.
* **Эффективность:** Высокая. Основные операции выполняются быстро за счёт использования хэш-таблиц и сбалансированных деревьев.

**2. Удаление записи**

* **Алгоритм:**
  + По ключу (ID): Удаление из std::unordered\_map и обновление соответствующих индексов.
  + По полю (Name, Department): Используется индекс для быстрого доступа к списку записей, после чего записи удаляются.
* **Сложность:**
  + Удаление по ключу:
    - Из std::unordered\_map: **O(1)**.
    - Обновление индексов: **O(log k)** (где k — количество записей в индексе).
    - Итоговая сложность: **O(log k)**.
  + Удаление по полю:
    - Поиск записей через индекс: **O(1)**.
    - Обновление индексов: **O(m \* log k)** (где m — количество удаляемых записей, k — количество записей в индексе).
    - Итоговая сложность: **O(m \* log k)**.
* **Эффективность:** Удаление по ключу быстрое, удаление по полю зависит от количества записей в индексе.

**3. Поиск записи**

* **Алгоритм:**
  + По ключу (ID): Используется std::unordered\_map::find.
  + По полю (Name, Department): Используется индекс для быстрого доступа.
* **Сложность:**
  + Поиск по ключу: **O(1)**.
  + Поиск по полю: **O(1)** для доступа к индексу и **O(m)** для извлечения записей (где m — количество записей с совпадающим значением поля).
  + Итоговая сложность: **O(m)**.
* **Эффективность:** Операции поиска выполняются быстро благодаря использованию индексов.

**4. Создание, резервное копирование и восстановление**

* **Алгоритм:** Чтение/запись данных из файла построчно.
* **Сложность:**
  + Запись в файл (бэкап, экспорт): **O(n)** (где n — количество записей).
  + Чтение из файла (восстановление): **O(n)**.
* **Эффективность:** Линейная сложность операций работы с файлами считается оптимальной.

**Временная статистика**

Тестирование проводилось на данных с количеством записей: 10, 1000 и 100000.

| **Операция** | **Размер данных** | **Время выполнения (мс)** |
| --- | --- | --- |
| Добавление записи | 10 | <1 |
|  | 1000 | <1 |
|  | 100000 | 3 |
| Удаление по ключу | 10 | <1 |
|  | 1000 | <1 |
|  | 100000 | 1 |
| Удаление по полю | 10 | <1 |
|  | 1000 | 1 |
|  | 100000 | 20 |
| Поиск по ключу | 10 | <1 |
|  | 1000 | <1 |
|  | 100000 | <1 |
| Поиск по полю | 10 | <1 |
|  | 1000 | <1 |
|  | 100000 | 15 |
| Создание бэкапа | 10 | <1 |
|  | 1000 | <1 |
|  | 100000 | 5 |
| Восстановление из бэкапа | 10 | <1 |
|  | 1000 | <1 |
|  | 100000 | 8 |