## 2.

Целью данной курсовой работы является разработка базы данных для онлайн-мониторинга состояния трасс и подъемников горнолыжного курорта.

Чтобы достигнуть поставленной цели, требуется решить следующие за-

дачи:

* проанализировать существующие решения;
* формализовать задачу и данные;
* проанализировать способы хранения данных и системы управления базами данных;
* выбрать наиболее подходящие для задачи тип БД и СУБД;
* спроектировать и разработать базу данных;
* реализовать программное обеспечение для доступа к данным посредством REST API;
* провести исследование зависимости времени обработки данных от их объема и от распределения вычислений между базой данных и приложением.

## 3.

Прежде всего, был проведен анализ существующих решений. Приложения горнолыжных курортов Газпром, куршавель и Роза Хутор оценивались по следующим критериям:

* наличие информации об открытости/закрытости трасс и подъемников; (1 столбец)
* наличие информации об очередях на подъемниках; (2 столбец)
* наличие сводной информации о связях трасс и подъемников; (3 столбец)
* возможность получить информацию о том, до каких трасс можно добраться на конкретном подъемнике/на каких подъемниках можно добраться до конкретной трассы. (4 столбец)

Как видно из таблицы, ни одно из решений не отображает всю информацию, которую предполагается предоставлять в разрабатываемом приложении, поэтому его разработка целесообразна.

## 4.

На слайде приведена ER-диаграмма БД «Горнолыжный курорт» в нотации Чена  
Была выделена такая сущность, как пользователь, который может отправлять и читать сообщения, просматривать информацию о трассах и подъемниках, при этом трассы и подъемники связаны между собой отношением многие-ко-многим: один подъемник может вести к нескольким трассам, а к одной трассе можно добраться на разных подъемниках. Также пользователь может обладать проездной картой, с помощью которой он может проходить на подъемник, прикладывая ее к турникету подъемника (то есть производя считывание)

## 5.

В соответствии с поставленной задачей необходимо разработать приложение с возможностью аутентификации пользователей, что делит их, прежде всего, на авторизованных и неавторизованных. Для управления приложением необходима ролевая модель: авторизованный (обычный) пользователь, сотрудник лыжного патруля и администратор.

Набор функций неавторизованного пользователя:

* регистрация,
* аутентификация,
* просмотр информации о состоянии трасс и подъемников,
* просмотр информации о связях трасс и подъемников.

Набор функций авторизованного пользователя:

* выход,
* просмотр информации о состоянии трасс и подъемников,
* просмотр информации о связях трасс и подъемников,
* отправка сообщений о происшествиях.

## 6.

Набор функций сотрудника лыжного патруля:

* выход,
* просмотр и изменение информации о состоянии трасс и подъемников,
* просмотр и изменение информации о связях трасс и подъемников,
* просмотр сообщений о происшествиях.

Набор функций администратора:

* выход,
* просмотр и изменение всей информации, доступной в базе данных, в том числе права доступа групп и отдельных пользователей.

## 7.

Для разработки БД прежде всего необходимо выбрать, как будут храниться данные.

По месту хранения информации БД можно разделить на [4]:

* традиционные, которые хранят информацию на жестком диске или другом постоянном носителе;
* in-memory databases (IMDB) (резидентные базы данных), которые хранят информацию непосредственно в оперативной памяти.
* Преимуществами традиционных БД, в сравнении с резидентными, является энергонезависимость, надежность хранения данных, а также отсутствие ограничения размера объемом оперативной памяти.
* Однако традиционные БД уступают in-memory в скорости обработки данных и пропускной способности.

Основным требованием к разрабатываемой БД является предоставление возможности **онлайн-мониторинга** состояния объектов горнолыжного курорта. Задача предполагает постоянное добавление и изменение данных. С особо высокой частотой будут добавляться считывания карт на турникетах подъемников. При этом время в очереди на подъемник должно регулярно пересчитываться и обновляться, чтобы пользователь не получил устаревшую информацию. Решение также предполагает быструю отзывчивость на запросы пользователя.

Таким образом, задача является типовым примером использования IMDB.

И поскольку в современных in-memory СУБД существуют надежные и достаточно простые способы устранения указанных недостатков этих БД, бы-

ло принято решение использовать именно этот подход к хранению данных.

## 8.

Далее необходимо выбрать in-memory СУБД. Сравнивались такие СУБД, как Memcached, Redis и Tarantool.

для решения поставленной задачи СУБД должна удовлетворять нескольким требованиям:

1. Для удобного хранения данных, указанных на ER-диаграмме, СУБД должна предоставлять, как минимум , два типа данных – строки и целые числа.
2. Для предоставления пользователям приведенного функционала, в СУБД должна быть реализована возможность создания вторичных индексов, а также триггеров или хранимых процедур для выполнения сложных вычислений.
3. Необходимо надежное хранение данных, без риска их потери даже в случае сбоя в системе (то есть поддержка репликации данных).

Из рассмотренных in-memory СУБД всем перечисленным требованиям удовлетворяет Tarantool, поэтому именно он был выбран для использования в данной работе.

## 9.

На слайде приведена диаграмма БД «Горнолыжный курорт»

БД хранит следующие таблицы:

* трассы slopes;
* подъемники lifts;
* связи трасс и подъемников lifts\_slopes;
* турникеты turnstiles;
* проездные карты cards;
* считывания карт на турникетах подъемников card\_readings;
* сообщения messages;
* пользователи users.

## 10.

Для онлайн-мониторинга очередей на подъемниках необходимо реализовать функцию, которая будет обновлять время в очереди к каждому из них. Эта функция будет периодически, через некоторый заданный интервал времени, вызываться из приложения для каждого подъемника из БД.

На слайде представлены схемы алгоритмов двух функций:

* вспомогательной функции count\_card\_readings, которая
  + принимает id подъемника, время предыдущего обновления и время запроса,
  + и возвращает количество считываний на указанном подъемнике в заданный промежуток времени.
  + Для этого сначала счетчик принимается равным 0, осуществляется поиск связанных с подъемником турникетов, затем в цикле для каждого турникета осуществляется поиск считываний на нем, и к счетчику прибавляется количество считываний на этом турникете, которые произошли в указанный промежуток времени.
* основная функция – update\_queue\_time
  + принимает те же параметры
  + обновляет время в очереди к указанному подъемнику и возвращает новое время
  + Для этого от предыдущего времени в очереди необходимо
    - отнять прошедшее с последнего обновления время
    - прибавить количество считываний на данном подъемнике в указанный промежуток времени, умноженное на время подъема, умноженное на 2 (так как каждому креслу необходимо проделать путь дважды – подняться и спуститься), и деленное на количество мест на подъемнике

## 11.

На слайде представлена архитектура приложения.

Интерфейс для работы с базой данных представляет собой многокомпонентное desktop-приложение. На базовом уровне выделены три компонента:

1. Компонент бизнес-логики спроектирован по схеме MVP (Model-View-Presenter) и является главным (независимым). Он отвечает за логику обработки запросов и данных.
2. Компонент доступа к данным отвечает за получение данных из БД, их изменение, добавление и удаление. Для его реализации использован паттерн «репозиторий».
3. Компонент реализации пользовательского интерфейса предоставляет реализацию UI для выбранного технологического стека.

Также на слайде представлен стек технологий:

1. Как уже упоминалось, в качестве СУБД был выбран Tarantool
2. В качестве языка программирования выбран язык C#
   1. он объектно-ориентирован, что позволит использовать наследование, интерфейсы, абстракции и так далее.
   2. имеет коннектор progaudi.tarantool для платформы in-memory вычислений Tarantool.
3. В качестве среды разработки была выбрана «Microsoft Visual Studio 2022»
   1. имеет множество удобств для написания и отладки кода;
   2. является бесплатной для студентов;
   3. обеспечивает работу с фреймворком Windows Forms[18], который был выбран для реализации пользовательского интерфейса из-за своей простоты в реализации.
4. Для упаковки приложения в готовый продукт была выбрана система контейнеризации Docker [19].
   1. С его помощью можно создать изолированную среду для программного обеспечения, которое можно будет развернуть на различных устройствах без дополнительных настроек.

## 12.

Был проведен эксперимент, целью которого являлось исследование зависимости времени обновления длительности ожидания в очередях к подъемникам от их количества и от того, где производятся основные шаги вычислений: на стороне БД или на стороне приложения.

Количество подъемников варьировалось от 50 до 800

При этом сравнивалось 4 алгоритма, пронумерованных в порядке убывания объема вычислений, производимых на стороне БД.

1. В первом алгоритме б ́ольшая часть вычислений осуществляется на стороне БД.
   1. После получения всех подъемников
   2. в цикле для каждого из них вызывается функция update\_queue\_time, которая на уровне БД производит все вычисления:
      1. поиск связанных с подъемником турникетов,
      2. подсчет считываний на этих турникетах в заданный промежуток времени,
      3. расчет нового времени ожидания
      4. обновление соответствующего кортежа в спейсе lifts.
2. Во втором алгоритме происходит примерно все то же самое, но на сторону приложения переносится сам расчет нового времени (при этом количество подходящих считываний все равно вычисляется на стороне БД): часть с вычитанием и сложением времен
   1. *после получения всех подъемников*
   2. *в цикле для каждого из них:*
      1. *вызывается функция count\_card\_readings, которая на уровне БД подсчитывает количество считываний карт в заданный промежуток времени на турникетах, связанных с данным подъемником*
      2. *на уровне приложения проводится расчет нового времени ожидания в очереди;*
      3. *осуществляется вызов функции для обновления соответствующего кортежа в спейсе lifts.*
3. В третьем алгоритме
   1. после получения всех подъемников и считываний карт
   2. происходит отбрасывание считываний, которые не попали в указанный промежуток времени,
   3. а затем в цикле для каждого подъемника:
      1. путем обращения к БД получаются все турникеты, связанные с данным подъемником;
      2. на уровне приложения подсчитывается количество считываний карт в заданный промежуток времени на этих турникетах
      3. осуществляется расчет нового времени ожидания в очереди;
      4. производится вызов функции для обновления соответствующего кортежа в спейсе lifts.
4. В четвертом алгоритме б ́ольшая часть вычислений осуществляется на уровне приложения, и обращение к БД происходит только в начале для получение всех подъемников, считываний и турникетов из БД и в конце для обновления соответствующего кортежа в спейсе lifts.
   1. *Сначала происходит получение всех подъемников, считываний и турникетов из БД.*
   2. *Затем отбрасываются считывания, которые не попали в указанный промежуток времени.*
   3. *Далее в цикле для каждого подъемника:* 
      1. *на уровне приложения определяются связанные с ним турникеты;*
      2. *подсчитываются чтения карт на этих турникетах;*
      3. *осуществляется расчет нового времени ожидания в очереди; производится вызов функции дя обновления соответствующего кортежа в спейсе lifts.*

В результате поставленного эксперимента была выявлена линейная зависимость времени обновления длительности ожидания в очередях к подъемникам от их количества.

Исследование также показало, что при фиксированном количестве подъемников на время выполнения обновления б ́ольшеевлияние оказывает не то, где производятся основные шаги вычислений (на стороне БД или на стороне приложения), а объем данных, передаваемых

между приложением и БД.

## 13.

Цель курсовой работы достигнута, все поставленные задачи решены:

* проанализированы существующие решения;
* формализованы задача и данные;
* проанализирована способы хранения данных и системы управления базами данных;
* выбраны наиболее подходящие для задачи тип БД и СУБД;
* спроектирована и разработана база данных;
* реализовано программное обеспечение для доступа к данным посредством REST API;
* проведено исследование зависимости времени обработки данных от их объема и от распределения вычислений между базой данных и приложением.

Возможное дальнейшее развитие:

* использование данных с видеокамер;
* добавление информации о погодных условиях.