Оглавление

1 Аналитическая часть

В данном разделе приведена формализация задачи и данных, рассмотрены типы пользователей и требуемый функционал. Представлен анализ способов хранения данных и систем управления базами данных, а также произведен выбор оптимальной для решения поставленной задачи системы управления базой данных.

1.1 Формализация задачи

Необходимо спроектировать и реализовать базу данных для онлайн-мониторинга состояния трасс и подъемников горнолыжного курорта. Также необходимо разработать интерфейс, позволяющий работать с данной базой для получения и изменения хранящейся в ней информации и мониторинга очередей к подъемникам в онлайн-режиме. Требуется реализовать, как минимум, три вида ролей – пользователь, сотрудник лыжного патруля и администратор.

1.2 Формализация данных

База данных должна хранить информацию о:

- трассах;
- подъемниках;
- связях трасс и подъемников (на одном подъемнике можно добраться до нескольких трасс, и до одной трассы можно добраться на нескольких подъемниках);
- турникетах;
- проездных картах;
- считываниях карт на турникетах подъемников;

- сообщениях о происшествиях;
- пользователях;
- группах пользователей.

В таблице 1.1 приведены категории и сведения о данных.

Таблица 1.1: Категории и сведения о данных

Категория	**Сведения**
Трассы	ID трассы, название трассы, уровень сложности, открытость/закрытость.
Подъемники	ID подъемника, название подъемника, открытость/закрытость, количество мест, время подъема, время в очереди.
Связи трасс и подъемников	ID записи, ID подъемника, ID трассы.
Турникеты	ID турникета, ID подъемника, открытость/закрытость.
Проездные карты	ID карты, дата и время активации, тип.
Считывания карт на турникетах подъемников	ID записи, ID турникета, ID карты, дата и время считывания.
Сообщения о происшествиях	ID сообщения, ID отправителя, ID прочитавшего, текст сообщения.
Пользователи	ID пользователя, ID карты, email (логин), пароль, ID группы пользователей.
Группы пользователей	ID группы пользователей, права доступа.

1.3 Типы пользователей

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.

Для каждого типа пользователя предусмотрен свой набор функций:

- неавторизованный пользователь:
  - регистрация,

  - аутентификация,
  - просмотр информации о состоянии трасс и подъемников,
  - просмотр информации о связях трасс и подъемников;
- авторизованный пользователь:
  - выход,
  - просмотр информации о состоянии трасс и подъемников,
  - просмотр информации о связях трасс и подъемников,
  - отправка сообщений о происшествиях;
- сотрудник 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;
- администратор:
  - выход,
  - просмотр и изменение всей информации, доступной в базе данных, в том числе права доступа групп и отдельных пользователей.

1.4 Анализ баз данных и систем управления базами данных

Для реализации поставленной задачи необходимо выбрать подходящую базу данных (БД) и систему управления базой данных (СУБД).

БД – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. СУБД – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [1].

1.4.1 Классификация баз данных по месту хранения информации

По месту хранения информации БД можно разделить на [2]:

- 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.

IMDB появились как ответ традиционным БД в связи со снижением стоимости оперативной памяти, что позволяет хранить весь набор операционных данных непосредственно в памяти, увеличивая тем самым скорость их обработки более чем в 1000 раз [3].

Ключевыми преимуществами IMDB, в сравнении с традиционными БД, считаются следующие [4]:

- быстрота выполнения операций;
- эффективное сохранение зафиксированных данных, которые используются не часто, на жестком диске;
- высокая пропускная способность систем, критичных к производительности.

Обратной стороной этих достоинств являются следующие недостатки:

- однопоточность и эффективная утилизация только одного ядра ЦП, что не позволяет в полной мере воспользоваться возможностями современных многоядерных серверов;

- энергозависимость и привязка к размеру оперативной памяти.

В практическом плане IMDB-системы особенно востребованы в тех приложениях работы с данными в реальном времени, где требуется минимальное время отклика [5].

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 in-memory БД. И поскольку в современных СУБД существуют надежные и достаточно простые способы устранения указанных недостатков IMDB, было принято решение использовать именно этот подход к хранению данных.

1.4.2 Обзор in-memory СУБД

Memcached

Memcached - это высокопроизводительная система кэширования данных в оперативной памяти, предназначенная для использования в ускорении динамических веб-приложений за счет уменьшения нагрузки на базу данных. Memcached относится к семейству решений для управления данными NoSQL и основана на модели данных с ключевыми значениями [6].

Данная СУБД спроектирована так, чтобы все операции имели алгоритмическую сложность O(1), то есть время выполнения любой операции не зависит от количества хранящихся ключей. Это означает, что некоторые операции или возможности, реализация которых требует всего лишь линейного (O(n)) времени, в ней отсутствуют. Так, например, в Memcached отсутствует возможность группировки ключей.

Memcached не является надежным хранилищем – возможна ситуация,

когда ключ будет удален из кэша раньше окончания его срока жизни.

Управление внутренней памятью Memcached более эффективно в простейших случаях использования (при кэшировании относительно небольших и статических данных), поскольку оно потребляет сравнительно мало ресурсов памяти для метаданных. Строки (единственный тип данных, поддерживаемый Memcached) идеально подходят для хранения данных, которые только читаются, потому что строки не требуют дальнейшей обработки. Тем не менее, эффективность управления памятью Memcached быстро уменьшается, когда размер данных является динамическим, после чего память Memcached может стать фрагментированной [7].

Еще одно преимущество Memcached – достаточно простая масштабируемость: поскольку данная система многопоточна, ее можно увеличить, просто предоставив больше вычислительных ресурсов. Однако это может привести к потере части или всех кэшированных данных (в зависимости от того, используется ли постоянное хеширование).

Redis

Redis [8] – резидентная система управлениями базами данных класса NoSQL с открытым исходным кодом.

Основной структурой данных, с которой работает Redis является структура типа «ключ-значение», причем значения могут быть пяти различных типов. Данная СУБД используется как для хранения данных, так и для реализации кэшей и брокеров сообщений.

Redis хранит данные в оперативной памяти и снабжена механизмом «снимков» и журналирования, что обеспечивает постоянное хранение данных. Существует поддержка репликации данных типа master-slave, транзакций и пакетной обработки комманд.

Redis позволяет осуществлять мелкомасштабный контроль за вытеснением данных, предоставляя выбор из шести различных политик вытеснения [9].

К недостаткам Redis можно отнести отсутствие вторичных индексов и триггеров. Также транзакции в данной СУБД не удовлетворяют свойствам ACID (Atomicity – Атомарность, Consistency – Согласованность, Isolation –

Изолированность, Durability – Долговечность) [10].

Tarantool

Tarantool [11] – это платформа in-memory вычислений с гибкой схемой хранения данных для эффективного создания высоконагруженных приложений. Включает себя базу данных и сервер приложений на языке программирования Lua [12].

Записи в Tarantool хранятся в пространствах (space) – аналог таблицы в реляционной базе данных SQL. Внутри пространства находятся кортежи (tuples), которые похожи на строку в таблице SQL.

Tarantool объединяет в себе преимущества, характерные для кэша:

- «горячие данные»;
- оптимальная работа при высокой параллельной нагрузке;
- низкая задержка (99% запросов $<$ 1 мс, 99,9% запросов $<$ 3 мс);
- поддерживаемая загрузка на запись — до 1 миллиона транзакций в секунду на одном ядре ЦПУ;
- система работает постоянно, не нужно делать перерыв на профилактические работы,

и достоинства классических СУБД:

- персистентность;
- транзакции со свойствами ACID;
- наличие репликации (master-slave и master-master);
- наличие хранимых процедуры.
- поддержка первичных и вторичных индексов (в том числе, составных).

В Tarantool реализован механизм «снимков» текущего состояния хранилища и журналирования всех операций, что позволяет восстановить состояние базы данных после ее перезагрузки.

К недостаткам данной СУБД можно отнести относительно малое количество поддерживаемых языков (C, C#, C++, Erlang, Go, Java, JavaScript, Lua, Perl, PHP, Python, Rust) [11], а также более высокий порог входа по сравнению с ранее рассмотренными СУБД [9].

1.4.3 Выбор СУБД для решения задачи

В данной работе не предполагается хранение в БД очень большого количества информации, и в этом случае требованию о коротком времени отклика удовлетворяют все рассмотренные СУБД.

Для удобного хранения данных, указанных в пункте 1.2, СУБД должна предоставлять несколько типов данных, таких как строки, целые числа, числа с плавающей запятой. Этому требованию не удовлетворяет Memcached.

Для того, чтобы предоставить пользователям перечисленные в пункте 1.3 функции, в СУБД должна быть реализована возможность создания вторичных индексов, а также триггеров или хранимых процедур (для выполнения сложных вычислений). Данному требованию не удовлетворяет Redis.

При этом необходимо надежное хранение данных, без риска их потери даже в случае сбоя в системе, а также реализация ACID транзакций.

Всем перечисленным требованиям удовлетворяет Tarantool, поэтому именно эта СУБД была выбрана для использования в данной работе.

Вывод

В данном разделе была проведена формализация задачи и данных, рассмотрены типы пользователей и требуемый функционал. В результате анализа способов хранения данных и рассмотрения IMDB-систем, в качестве СУБД для данной работы был выбран Tarantool.

2 Конструкторская часть

2.1 Функциональная модель

На рисунке 2.1 изображена функциональная модель, отображающая структуру и функции системы.

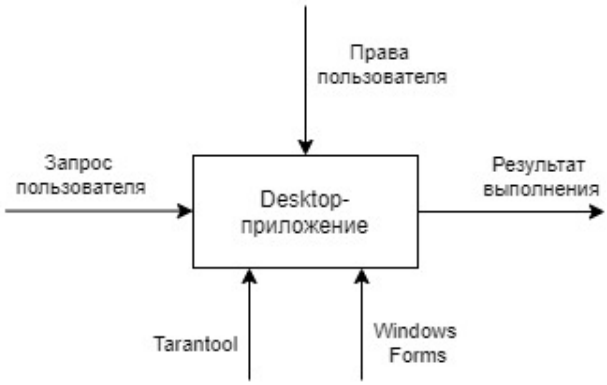

Рис. 2.1: Функциональная модель приложения

2.2 Сценарии использования

Сценарии использования описываются с помощью Use Case Diagram (диаграммы прецедентов). Она состоит из графической диаграммы, описывающей действующие лица и прецеденты – конкретные действия, которые выполняет пользователь при работе с системой.

На рисунке 2.2 представлена Use Case Diagram для действий неавторизованного пользователя.

Рис. 2.2: Use Case Diagram действий неавторизованного пользователя

На рисунке 2.3 представлена Use Case Diagram для действий авторизованного пользователя.

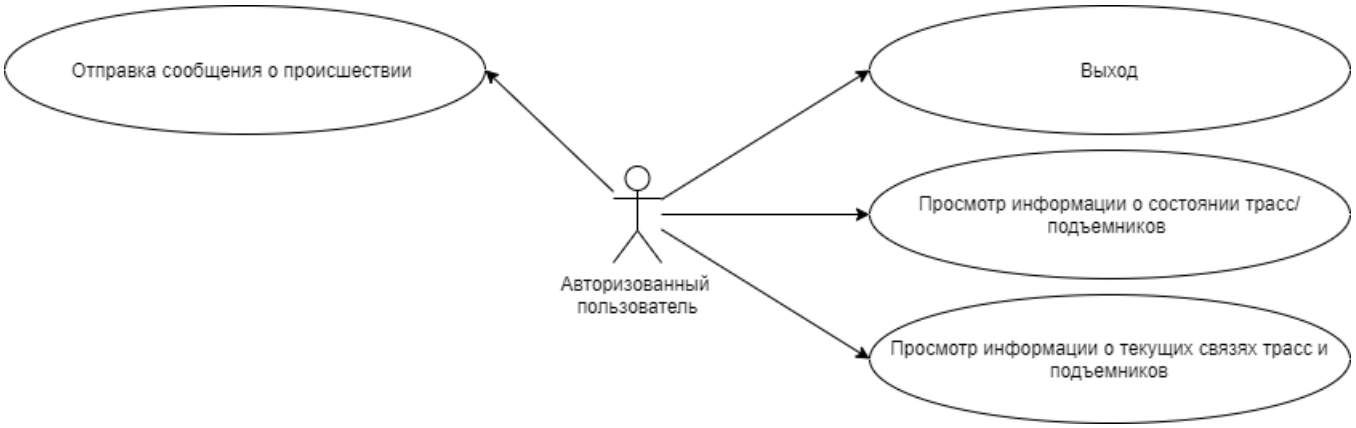

Рис. 2.3: Use Case Diagram действий авторизованного пользователя

На рисунке 2.4 представлена Use Case Diagram для действий сотрудника лыжного патруля.

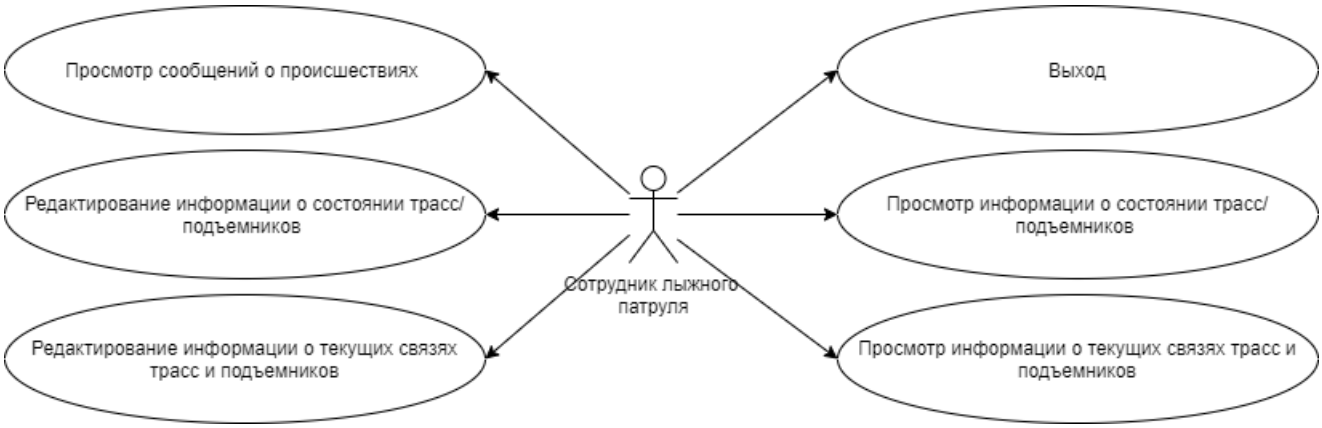

Рис. 2.4: Use Case Diagram действий сотрудника лыжного патруля

На рисунке 2.5 представлена Use Case Diagram для действий администратора.

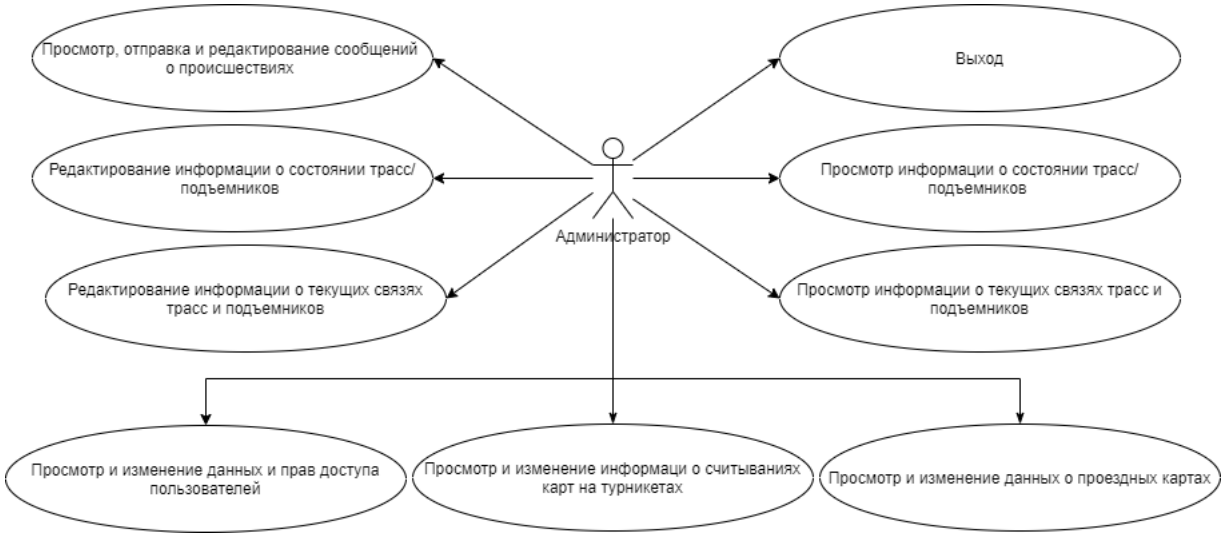

Рис. 2.5: Use Case Diagram действий администратора

2.3 Проектирование базы данных

На рисунке 2.6 отображена диаграмма сущностей системы. Данная схема построена на основе категорий и сведений о данных, отображенных в таблице 1.1.

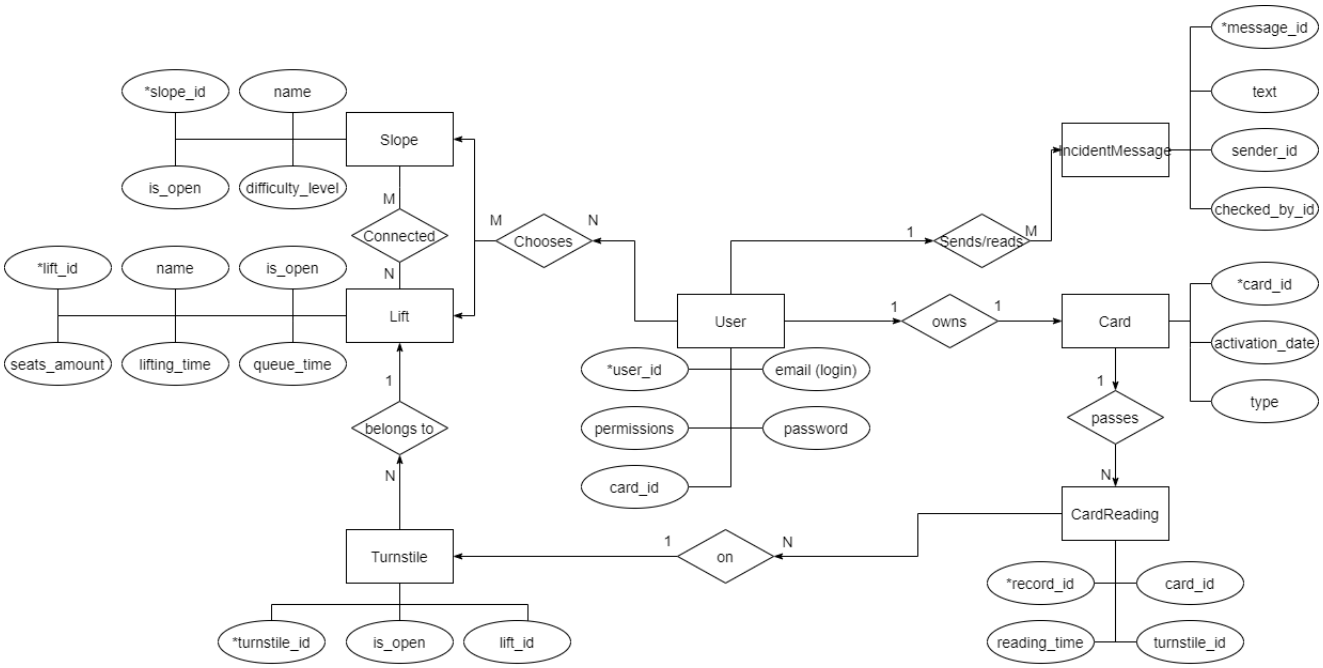

Рис. 2.6: Диаграмма сущностей системы

В соответствии с этой диаграммой база данных должна хранить следующие таблицы:

- таблица трасс slopes;
- таблица подъемников lifts;
- таблица связей трасс и подъемников lifts_slopes;
- таблица турникетов turnstiles;
- таблица проездных карт cards;
- таблица считываний карт на турникетах подъемников card_readings;
- таблица сообщений о происшествиях messages;
- таблица пользователей users.

Таблица slopes хранит информацию о трассах и содержит следующие поля:

- slope_id – уникальный идентификатор трассы, PK;
- slope_name – уникальное название;

- difficulty_level – уровень сложности;
- is_open – открыта или закрыта.

Таблица lifts хранит информацию о подъемниках и содержит следующие поля:

- lift_id – уникальный идентификатор подъемника, PK;
- lift_name – уникальное название;
- is_open – открыт или закрыт;
- seats_amount – количество мест;
- lifting_time – время подъема;
- queue_time – время в очереди;

Эти две таблицы связаны отношением многие-ко-многим. Таблица lifts_slopes хранит информацию об этих отношениях (связях трасс и подъемников) и содержит следующие поля:

- record_id – уникальный идентификатор записи, PK;
- lift_id – идентификатор подъемника, FK на поле lift_id таблицы lifts;
- slope_id – идентификатор трассы, FK на поле slope_id таблицы slopes.

Таблица turnstiles хранит информацию о турникетах и содержит следующие поля:

- turnstile_id – уникальный идентификатор турникета, PK;
- lift_id – идентификатор подъемника, FK на поле lift_id таблицы lifts;
- is_open – открыт или закрыт.

Таблица cards хранит информацию о проездных картах и содержит следующие поля:

- card_id – уникальный идентификатор карты, PK;

- activation_time – дата и время активации;
- type – тип карты (детская, взрослая, временная, ...).

Таблица card_readings хранит информацию о считываниях карт на турникетах подъемников и содержит следующие поля:

- record_id – уникальный идентификатор считывания, PK;
- turnstile_id – идентификатор турникета, FK на поле turnstile_id таблицы turnstiles;
- card_id – идентификатор проездной карты, FK на поле card_id таблицы cards;
- reading_time – дата и время считывания.

Таблица users хранит информацию о пользователях и содержит следующие поля:

- user_id – уникальный идентификатор пользователя, PK;
- card_id – идентификатор проездной карты (может отсутсвовать), FK на поле card_id таблицы cards;
- user_email – адрес электронной почты (он же будет использоавться как логин);
- password – пароль;
- permissions – права доступа (роль).

Таблица messages хранит информацию о сообщениях пользователей о происшествиях и содержит следующие поля:

- message_id – уникальный идентификатор сообщения, PK;
- sender_id – идентификатор отправителя, FK на поле user_id таблицы usres;
- checked_by_id – идентификатор прочитавшего, FK на поле user_id таблицы usres;

- text – текст сообщения.

На рисунке 2.7 предоставлена диаграмма реализованной базы данных.

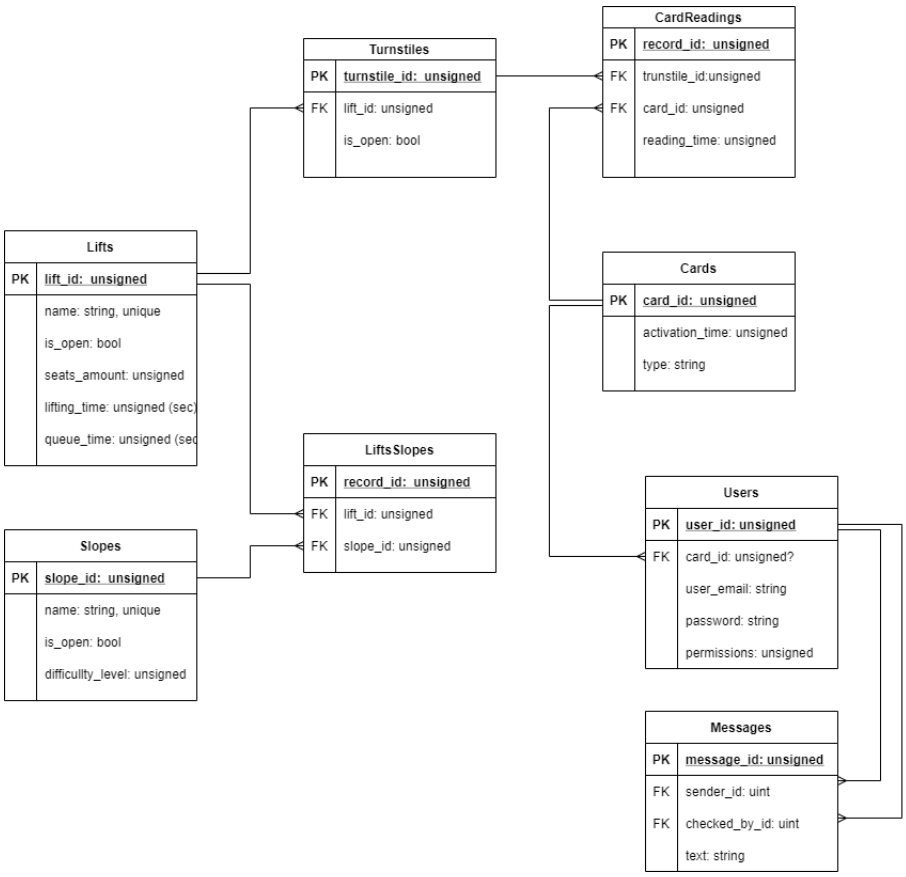

Рис. 2.7: Диаграмма базы данных

!!!!Кроме того, для каждой таблицы будет реализован триггер, срабатывающий после обновления или удаления данных из таблиц. Этот триггер будет посылать сигнал базе данных кэширования, с помощью языка plpython3u, о необходимости обновить или удалить информацию из кэша. С помощью таких триггеров можно решить проблему синхронизации данных в хранилище и кэше.

2.4 Проектирование базы данных кэширования

База данных кэширования будет реализована с помощью использования СУБД Tarantool. В базе данных будут полностью продублированны

таблицы (в виде спейсов) из хранилища рабочих программ дисциплин. Первичным ключом будет являться поле с уникальным идентификатором этих таблиц (`id`). Кроме того, для спейсов хранящих поле `discipline_id` будет добавлен вторичный ключ по этому полю, для удобного и быстрого сбора нужных данных по заданной дисциплине.

При запросе данных у приложения, будет проводиться проверка, присутствует ли запись в кэше. Если запись присутствует, запрос к базе данных рабочих программ дисциплин производиться не будет и будут возвращены данные из кэша. В противном случае, будет произведен запрос к базе данных хранящую информацию о дисциплинах.

Все спейсы будут созданы на основе движка `memtx`, хранящего все данные в оперативной памяти. Персистентность данных будет обеспечивается при помощи ведения журнала транзакция и системы «снимков» текущего состояния кэша. Эти технологии помогут решить проблему «холодного» старта базы данных кэширования.

Вывод

В данном разделе были представлены этапы проектирования баз данных и рассмотрены особенности используемых СУБД на архитектурном уровне.

Литература

[1] Что такое база данных | Oracle Россия и СНГ [Электронный ресурс]. Режим доступа: https://www.oracle.com/ru/database/what-is-database/ (дата обращения: 25.03.2022).

[2] Системы и технологии баз данных в памяти | Microsoft [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/sql/relational-databases/in-memory-database?view=sql-server-ver15 (дата обращения: 25.03.2022).

[3] In-Memory. База данных в оперативной памяти |ECM-Journal [Электронный ресурс]. Режим доступа: https://ecm-journal.ru/material/In-Memory-Baza-dannykh-v-operativnojj-pamjati (дата обращения: 27.03.2022).

[4] In-Memory Database [Электронный ресурс]. Режим доступа: https://ru.bmstu.wiki/IMDB_(In-memory_Database) (дата обращения: 27.03.2022).

[5] 4 крупных примера внедрения Tarantool | Big Data School [Электронный ресурс]. Режим доступа: https://www.bigdataschool.ru/blog/tarantool-use-cases-and-advantages.html (дата обращения: 27.03.2022).

[6] Memcached - a distributed memory object caching system [Электронный ресурс]. Режим доступа: https://memcached.org/ (дата обращения: 15.04.2022).

[7] Memcached | Распределенное хранилище данных типа «ключ-значение» [Электронный ресурс]. Режим доступа: https://memcached.org/ (дата обращения: 15.04.2022).

[8] Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker [Электронный ресурс]. Режим доступа: https://redis.io/ (дата обращения: 01.04.2022).

[9] Преимущества Redis [Электронный ресурс]. Режим доступа: https://aws.amazon.com/ru/redis/ (дата обращения: 01.04.2022).

[10] Транзакции, ACID, CAP | GeekBrains [Электронный ресурс]. Режим доступа: https://gb.ru/posts/acid_cap_transactions (дата обращения: 07.06.2021).

[11] Tarantool – Платформа In-memory вычислений [Электронный ресурс]. Режим доступа: https://www.tarantool.io/ru/ (дата обращения: 01.04.2022).

[12] The Programming Language Lua [Электронный ресурс]. Режим доступа: http://www.lua.org/ (дата обращения: 01.04.2022).