­Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы Данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Веб-приложение статистики матчей Dota2

БГУИР КП 1-40 01 01 013 ПЗ

Выполнил

студент: гр. 151004 Босько В.С.

Проверил: Новицкая Н.В.

Минск 2024

содержание

[Введение 3](#_Toc165985169)

[1 Аналитический обзор литературы 4](#_Toc165985170)

[1.1 Обзор существующих аналогов 4](#_Toc165985171)

[1.2 Применение 7](#_Toc165985172)

[1.3 Требования к проектируемому программному средству 7](#_Toc165985173)

[1.4 Постановка задачи 10](#_Toc165985174)

[2 Анализ требований к программному средству и разработка функциональных требований 11](#_Toc165985175)

[2.1 Обоснование выбора СУБД и иных средств разработки 11](#_Toc165985176)

[2.2 Спецификация функциональных требований 12](#_Toc165985177)

[2.3 Учитываемые функции и бизнес-процессы 12](#_Toc165985178)

[2.4 Ключевые особенности базы данных и программного средства 12](#_Toc165985179)

[2.5 Параметры поиска информации 13](#_Toc165985180)

[2.6 Алгоритмы обработки данных 13](#_Toc165985181)

[2.7 Бизнес-процессы, связанные с реализацией информационной системы 13](#_Toc165985182)

[2.8 Обоснование выбора метода взаимодействия между клиентом и сервером по передаче данных 14](#_Toc165985183)

[3 Инфологическая модель предметной области 15](#_Toc165985184)

[3.1 Описание модели 15](#_Toc165985185)

[3.2 Сущности и связи 18](#_Toc165985186)

[3.3 Особенности нормализации 26](#_Toc165985187)

[4 Подробное описание бизнес-логики 28](#_Toc165985188)

[4.1 Процедуры 28](#_Toc165985189)

[4.2 Триггеры 34](#_Toc165985190)

[5 Тестирование базы данных и приложений системы 36](#_Toc165985191)

[5.1 Тестирование функций 36](#_Toc165985192)

[5.2 Тестирование триггеров 36](#_Toc165985193)

[5.3 Тестирование программного средства 37](#_Toc165985194)

[Заключение 40](#_Toc165985195)

[Список использованной литературы 41](#_Toc165985196)

[Приложение А 42](#_Toc165985197)

# Введение

В современном мире компьютерных игр онлайн-игры становятся все более популярными, привлекая миллионы игроков по всему миру. Одной из таких популярных игр является Dota 2 — многопользовательская онлайн-игра в жанре MOBA (многопользовательская онлайн-боевая арена), разработанная и выпущенная компанией Valve Corporation. Стремительное развитие этой игры привело к появлению активного сообщества игроков, которые стремятся анализировать прошедшие матчи, извлекать уроки из ошибок и улучшать свою игру.

Актуальность выбранной темы:

Dota 2 является одной из самых популярных онлайн-игр в мире, с миллионами активных игроков. Стремительное развитие этой игры привело к созданию огромного объема данных о матчах, которые могут быть использованы для анализа и извлечения ценной информации. Среди игроков Dota 2 существует высокий интерес к анализу своей игры и улучшению своих навыков. Веб-приложение со статистикой матчей предоставит игрокам удобный инструмент для отслеживания своего прогресса, выявления областей для улучшения и изучения тактик и стратегий успешных игроков.

Особенности предметной области:

Предметная область приложения со статистикой матчей по Dota2 имеет свои особенности и требования. Dota 2 - это игра, в которой происходит огромное количество матчей ежедневно. Для создания базы данных необходимо рассмотреть масштабность и обширность данных, которые будут собираться и храниться. Данные о матчах Dota 2 могут быть довольно сложными из-за множества факторов, которые могут влиять на результаты игры. Это включает в себя информацию о героях, предметах, навыках, событиях матча и т. д. Так как игра постоянно обновляется и изменяется, база данных и веб-приложение должны быть готовы к постоянному обновлению и добавлению новой информации.

Учитывая эти особенности, разработчики должны тщательно спланировать структуру базы данных и функциональность веб-приложения, чтобы обеспечить эффективное хранение, обработку и представление данных для пользователей.

# Аналитический обзор литературы

## Обзор существующих аналогов

### Liquipedia.net

Liquipedia – это один из самых популярных веб-сайтов предоставляющий информацию киберспортивных дисциплинах, в том числе о Dota2.

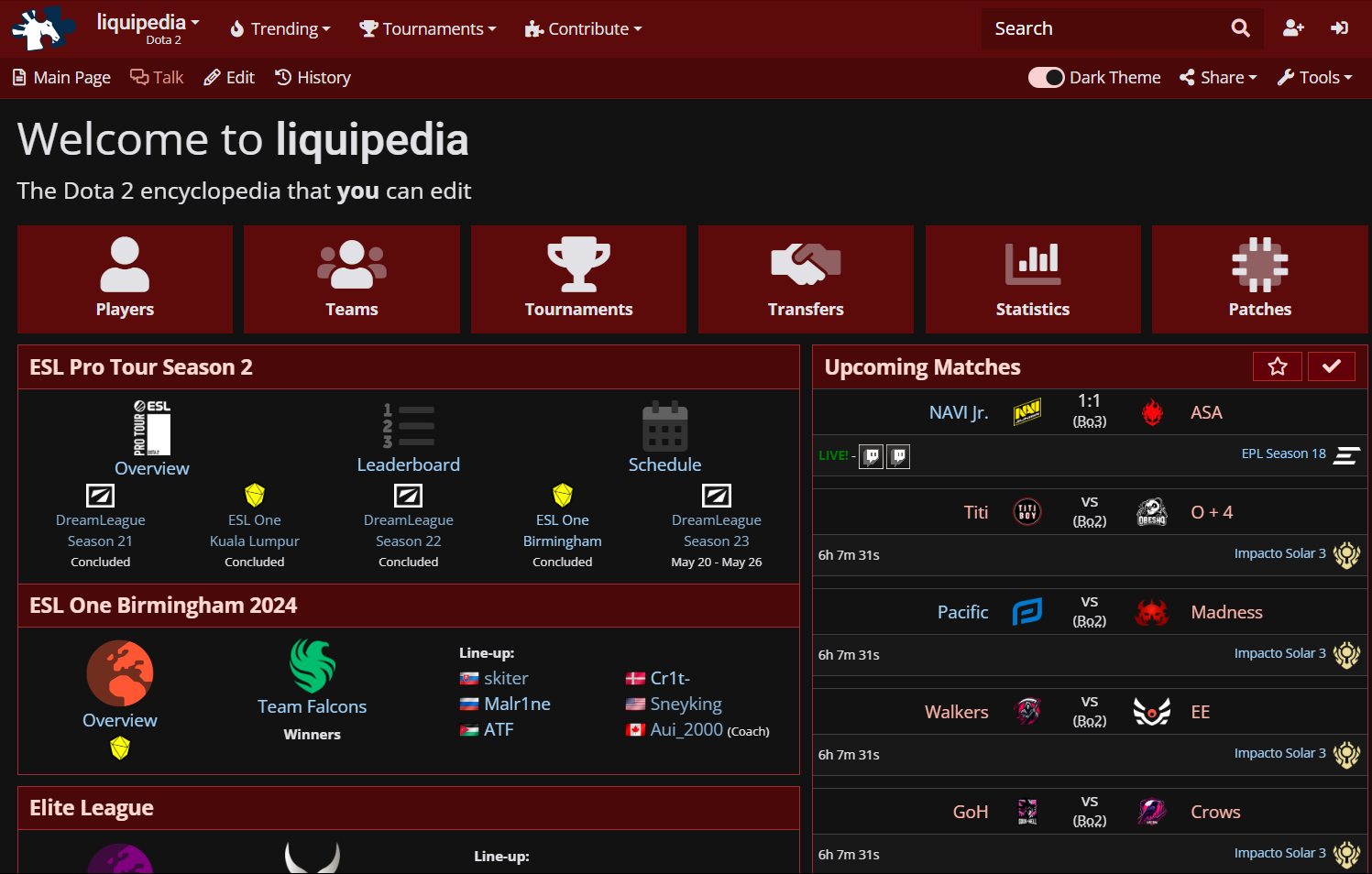


Рисунок . – Liquipedia.net

Достоинства:

* большое разнообразие дисциплин;
* актуальная информация;
* удобный интерфейс.

Недостатки:

* наличие рекламы;
* информация только о киберспортивной составляющей игры;
* зависимость от интернет-соединения;
* отсутствует русская локализация.

### Dota2.ru

Dota2.ru - это веб-сайт предоставляющий разнообразный контент в сфере Dota2. На сайте можно найти самые актуальные новости, а также статистику киберспортивных матчей.

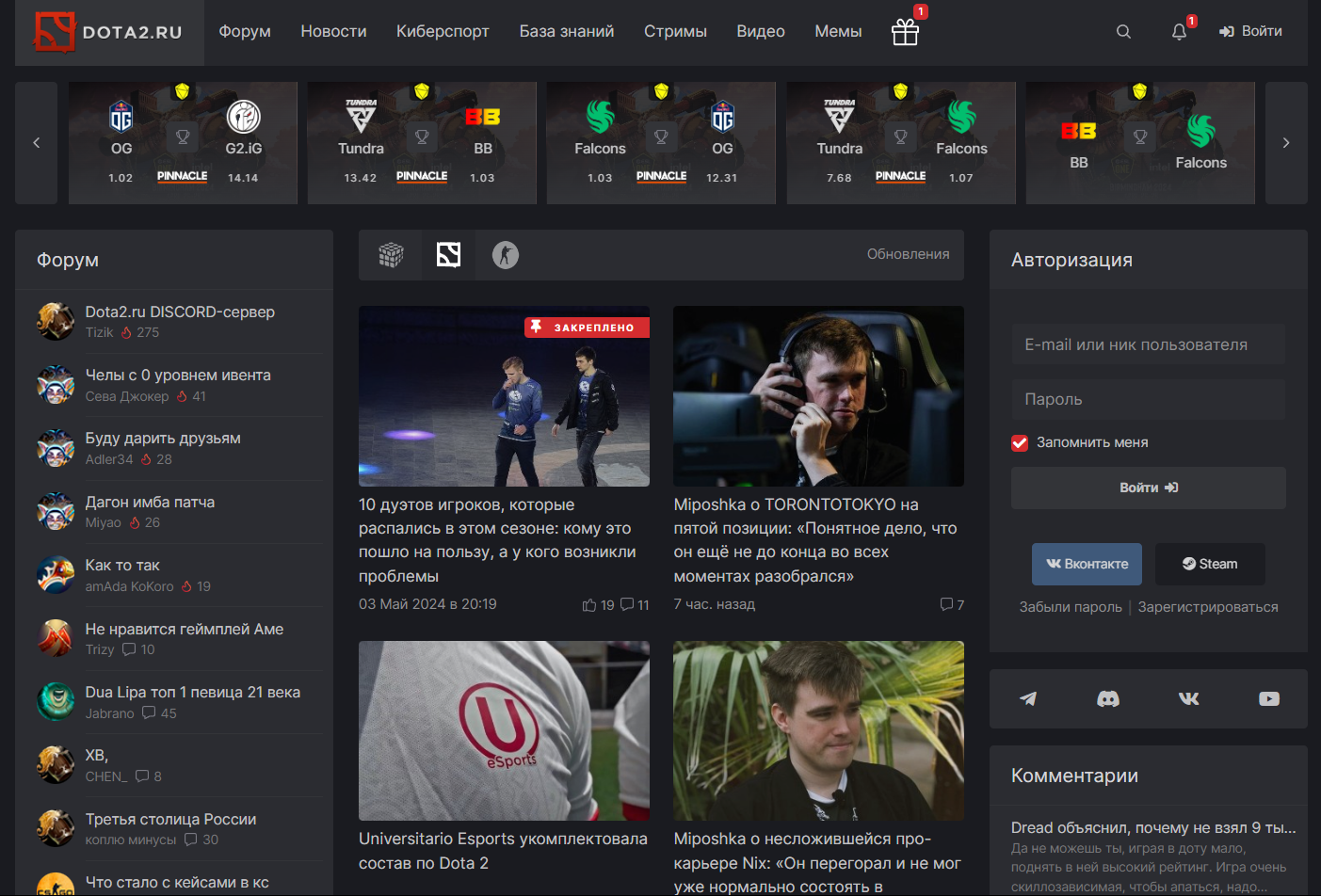


Рисунок . – Dota2.ru

Достоинства:

* большое разнообразие контента;
* русская локализация;
* актуальная информация.

Недостатки:

* обильное количество рекламы;
* мало статистики;
* зависимость от интернет-соединения;
* информация только о киберспортивной составляющей игры.

### Dotabaff.com

Dotabuff.com - это веб-сайт предоставляющий большое количество информации по Dota2, помимо киберспортивной составляющий сферы на сайте также можно получить статистику своих матчей и общеигровую статистику.



Рисунок . – Dotabuff.com

Достоинства:

* разнообразие статистических данных;
* русская локализация;
* актуальная информация;
* удобный интерфейс.

Недостатки:

* тяжело получить узконаправленную статистику;
* наличие рекламы;
* зависимость от интернет-соединения.

## Применение

База данных, созданная для анализа статистики матчей по игре Dota 2, имеет широкий спектр применений и открывает множество возможностей для пользователей.

1. Анализ личной игры: Игроки могут использовать базу данных для отслеживания своего прогресса, изучения своих сильных и слабых сторон, а также определения областей, в которых им необходимо улучшить свои навыки. Например, они могут анализировать свою статистику побед и поражений, эффективность игры на разных героях, использование предметов и тактики в различных ситуациях.

2. Изучение стратегий и тактик: Пользователи могут изучать статистику успешных игроков и профессиональных команд, чтобы понять, какие стратегии и тактики приводят к победам. Это может включать в себя анализ выбора героев, распределение ресурсов, временные моменты для совершения определенных действий и т. д.

3. Отслеживание изменений в мете игры: Игровая мета постоянно меняется из-за обновлений игры, патчей и изменений в игровом балансе. Пользователи могут использовать базу данных для отслеживания этих изменений, анализа их влияния на результаты матчей и адаптации своей игры под текущие условия.

4. Статистические исследования: База данных может быть использована для проведения различных статистических исследований о игровой активности и поведении игроков. Например, исследование предпочтений игроков по выбору героев, анализ распределения побед между разными режимами игры или изучение влияния обновлений на популярность определенных стратегий.

5. Разработка обучающих материалов и руководств: На основе данных из базы можно создавать обучающие материалы, руководства и видеоуроки для новичков и опытных игроков, помогая им лучше понять игровые механики, стратегии и тактики.

Удобство использования базы данных заключается в доступности информации, ее структурированности и возможности быстрого поиска и анализа нужных данных с помощью специализированного веб-приложения. Пользователи могут легко находить интересующую их информацию, используя различные фильтры, сортировки и визуализации, предоставляемые приложением.

## Требования к проектируемому программному средству

### Назначение разработки

Назначение программного средства, создаваемого для анализа статистики матчей по игре Dota 2, состоит в предоставлении игрокам удобного инструмента для изучения и анализа их игровой активности, а также для получения ценной информации о текущем состоянии игровой меты. Вот основные аспекты назначения этого программного средства:

Программное средство позволяет игрокам анализировать свою игру, выявлять ошибки, определять области для улучшения и разрабатывать стратегии для достижения лучших результатов. Это помогает игрокам повысить свой уровень мастерства и эффективность в игре.

Пользователи могут использовать программное средство для отслеживания текущих трендов и изменений в игровой мете Dota 2. Анализ статистики матчей помогает понять, какие герои, стратегии и тактики в настоящее время наиболее эффективны, что позволяет игрокам адаптировать свою игру под текущие условия.

Создание программного средства способствует формированию сообщества игроков Dota 2, которые могут обмениваться опытом, стратегиями и советами. Пользователи могут делиться своими анализами и наблюдениями, что способствует распространению знаний и повышению общего уровня игрового сообщества.

Профессиональные игроки и тренеры могут использовать программное средство для анализа матчей, изучения игровых стратегий и разработки тактик для своих команд. Это помогает им улучшить свою подготовку к соревнованиям и достичь лучших результатов на турнирах.

В целом, целью программного средства является повышение качества игры игроков Dota 2 путем предоставления им доступа к полезной и актуальной информации о матчах и игровой мете, а также поддержка их в процессе анализа и улучшения игровых навыков.

### Состав выполняемых функций

Учёт профессиональных игроков: БД хранит информацию обо всех профессиональных игроках в Dota2. ПС позволяет просматривать и добавлять информацию.

Учёт киберспортивных команд: БД хранит информацию обо всех командах и организациях причастных к дисциплине. ПС позволяет просматривать и добавлять информацию. ПС позволяет просматривать и добавлять информацию.

Учёт турниров: БД хранит информацию обо всех наиболее популярных турнирах и их организаторах. ПС позволяет просматривать и добавлять информацию.

Учёт матчей: БД хранит всю информацию об матчах, в том числе их статистику. ПС позволяет просматривать и добавлять информацию.

Учёт внутриигровой информации: БД хранит внутриигровую информацию о героях, предметах, механиках и т.д.

### Входные данные

Информация о:

* Игроках;
* Командах;
* Турнирах;
* Организаторах турниров;
* Студиях освещения турниров;
* Матчах;
* Игровых героях;
* Игровых предметов.

### Выходные данные

Составы команд в разные временные промежутки.

Информация о турнирах, участниках, результатах команд, статистика турнира.

Подробная информация о матчах, статистика.

Информация о внутриигровых механиках: предметах, героях, сущностей.

### Требования к составу и параметрам технических и программных средств

В состав технических средств входит сервер, включающий в себя:

* процессор Intel i3 и выше;
* 512 Мбайт оперативной памяти;
* 1 Гбайт свободного пространства на жестком диске;
* Хорошее подключение к интернету.

### Требования к информационной и программной совместимости

Программа рассчитывается на работу на сервере и требует иметь .NET 8 на устройстве сервера. А для клиентской стороны достаточно наличия браузера.

### Обоснование выбора языка и сред разработки

Для разработки программного средства были выбраны язык C# и среда разработки Visual Studio 2022, так как - это мощный и популярный инструментарий для разработки приложений. C# - современный язык программирования с понятным синтаксисом, а Visual Studio 2022 предлагает удобную интегрированную среду разработки с множеством функций. Они обеспечивают разработчикам доступ к богатой экосистеме, высокую производительность и надежность приложений, а также поддержку и обновления со стороны Microsoft.

## Постановка задачи

Разработать приложение для просмотра информации о матчах по игре Dota2 с использованием базы данных. Приложение должно предоставлять пользователям возможность искать информацию по различным критериям, просматривать подробную информацию о матчах. База данных должна содержать информацию о командах, игроках, турнирах, матчах, внутриигровых механиках.

# Анализ требований к программному средству и разработка функциональных требований

## Обоснование выбора СУБД и иных средств разработки

В качестве СУБД была выбрана MySQL по следующим причинам: MySQL является одной из самых популярных коммерческих реляционных баз данных, обладает надежностью, стабильностью и поддерживает ACID-свойства, обеспечивая целостность данных и надежность операций. MySQL предоставляет широкий набор встроенных функций и типов данных, позволяющих эффективно моделировать и хранить различные типы данных, включая географические данные, JSON, XML и другие. Это обеспечивает гибкость и возможность адаптации к различным потребностям проекта.

Для создания API на основе структуры и данных, хранящихся в MySQL, был выбран ASP.NET Core. ASP.NET Core является мощным фреймворком для разработки веб-приложений, который поддерживает создание RESTful API. Он обладает высокой производительностью, масштабируемостью и безопасностью. С помощью ASP.NET Core, разработчики могут легко создавать и настраивать контроллеры и маршруты API, определять модели данных и взаимодействовать с базой данных MySQL.

Для документации API был использован Swagger. Swagger является популярным инструментом для автоматической генерации интерактивной документации API. Он позволяет описывать и документировать эндпоинты, модели данных, параметры запросов и другую информацию о API. В ASP.NET Core, Swagger интегрируется с помощью пакета Swashbuckle, который автоматически генерирует Swagger-спецификацию на основе атрибутов, определенных в коде API. Это упрощает понимание и использование API разработчиками и другими участниками проекта.  
 Для создания более сложных и интерактивных пользовательских интерфейсов на клиентской стороне был выбран фреймворк MVC. MVC предоставляет мощные инструменты для создания компонентов пользовательского интерфейса, обновления только необходимых частей страницы при изменении данных и обработки пользовательских событий. Это позволяет создавать реактивные и отзывчивые веб-приложения, которые быстро отображают изменения и обеспечивают более плавное взаимодействие с пользователем.

Использование MySQL в сочетании с ASP.NET Core, Swagger и MVC предоставляет надежное и гибкое решение для создания базы данных, разработки API и генерации документации API. Это позволяет сократить время разработки, упростить поддержку и обеспечить эффективное использование данных.

## Спецификация функциональных требований

Хранение информации о Dota2: База данных должна предоставлять возможность хранения информации о матчах, командах, игроках, принадлежности игроков команде и их роли, турнирах, местах проведения турниров, игровых механиках: информация о героях, предметах, сущностях.

Управление рейтингом: База данных должна автоматически обновлять рейтинг команд и игроков получая информацию о результатах команд и игроков на турнирах.

Составление статистики: База данных должна хранить подробную статистику матчей и предоставлять ее пользователям.

## Учитываемые функции и бизнес-процессы

База данных для приложения со статистикой по Dota2 должна предоставлять возможность хранения информации о матчах, командах, игроках, принадлежности игроков команде и их роли, турнирах, местах проведения турниров, игровых механиках: информация о героях, предметах, сущностях.

Программное средство будет иметь собственную систему рейтинга команд и должно автоматически ее обновлять.

Программное средство должно позволять просматривать статистику матчей, турниров, результаты команд, предстоящие события.

## Ключевые особенности базы данных и программного средства

Программное средство должно позволять работать одновременно большому числу пользователей, что требует хорошей оптимизации как самого программного средства так и базы данных. База данных должна содержать в себе сущности для хранения таких важных данных, как статистика матчей, рейтинг команд, внутриигровая инфомрация. Для обеспечения эффективной работы программного средства и базы данных в условиях большой нагрузки, можно применить следующие оптимизационные подходы:

1. Индексирование: Создание индексов на часто используемых полях в базе данных позволяет ускорить поиск и фильтрацию данных. Например, можно создать индексы для полей, используемых при поиске команд, игроков, турниров, героев, предметов и т.д.  
    2) Оптимизация запросов: Анализ и оптимизация запросов к базе данных могут значительно повысить производительность системы. Использование индексов, правильное написание запросов с использованием оптимальных операторов и избегание излишних запросов могут сократить время выполнения запросов и уменьшить нагрузку на базу данных.

## Параметры поиска информации

Данные поступают в базу данных через API, который реализован с использованием контроллеров в ASP.NET Core. Контроллеры служат посредниками между клиентскими запросами и сервисами, которые выполняют операции с базой данных MySQL.

Когда клиент отправляет запрос через API, соответствующий контроллер принимает этот запрос и передает его соответствующему сервису. Сервис содержит логику обработки запроса и взаимодействует с базой данных, отправляя соответствующие SQL-запросы для получения или изменения данных.

После получения данных из базы данных, сервис возвращает их контроллеру, который в свою очередь формирует ответ и отправляет его обратно клиенту. Клиентский интерфейс отображает полученные данные пользователю.

## Алгоритмы обработки данных

Информация о игре, матчах и результатах может быть получена из различных источников, через API Dota2, от базы данных турнирных операторов или ручной ввод. Операции, выполняемые над этими данными, включают добавление новой информации об игре и матчах, редактирование информации о прошедших событиях, удаление записей и просмотр данных для анализа и отчетности.

## Бизнес-процессы, связанные с реализацией информационной системы

Множество запросов на чтение: предполагается, что пользователи базы данных часто выполняют запросы на чтение данных, например, для аналитических целей или отображения информации, это может потребовать оптимизации процесса чтения данных.

Большое количество параллельных запросов: база данных получает большое количество параллельных запросов от разных пользователей. Поэтому следует разработать план оптимизации как запросов к базе данных, так и саму базу данных. Также для программного средства тоже следует оптимизировать архитектуру, для более быстрого выполнения запросов.

## Обоснование выбора метода взаимодействия между клиентом и сервером по передаче данных

Выбор REST API в качестве метода взаимодействия между клиентом и сервером для передачи данных обусловлен рядом преимуществ. Во-первых, REST обеспечивает архитектурную гибкость, позволяя разделить клиентскую и серверную части приложения, что упрощает масштабирование и обновление системы. Во-вторых, REST API основан на простых и понятных принципах, использующих стандартные HTTP-методы для выполнения различных операций с данными, что делает его понятным и удобным для разработчиков. Третье преимущество состоит в универсальности и доступности REST API, так как оно широко поддерживается различными языками программирования, фреймворками и платформами. Кроме того, REST API поддерживает возможности кэширования HTTP, что позволяет улучшить производительность и снизить нагрузку на сервер. Наконец, REST API поддерживает гибкую передачу данных в различных форматах, таких как JSON, XML или файлы, обеспечивая удобство обмена данными между клиентом и сервером.

В качестве фреймворка для организации клиент-серверного соединения используется ASP.NET. Он был выбран, так как обеспечивает высокую производительность и масштабируемость, позволяя обрабатывать большое количество запросов от клиентов. Он предлагает встроенную поддержку для различных протоколов и форматов данных, таких как JSON и XML, для удобного обмена данными между клиентами и сервером. ASP.NET также предоставляет обширную библиотеку классов, которая упрощает разработку API, включая функциональности для аутентификации, авторизации и управления состоянием. Он поддерживает различные способы маршрутизации запросов и предоставляет инструменты для тестирования, обнаружения и исправления ошибок API до их выпуска в продакшн. ASP.NET обеспечивает безопасность API с помощью встроенных механизмов защиты от уязвимостей, таких как атаки CSRF и XSS. Он также легко интегрируется с другими сервисами и инфраструктурой, такими как базы данных и облачные платформы. ASP.NET предоставляет инструменты для мониторинга и отладки API, помогающие выявлять и устранять проблемы производительности и неполадки. Он имеет большое сообщество разработчиков, обеспечивающее доступ к обширной базе знаний, документации и поддержке. Кроме того, ASP.NET является частью экосистемы .NET, что облегчает интеграцию с другими компонентами и сервисами, разработанными с использованием .NET.

# Инфологическая модель предметной области

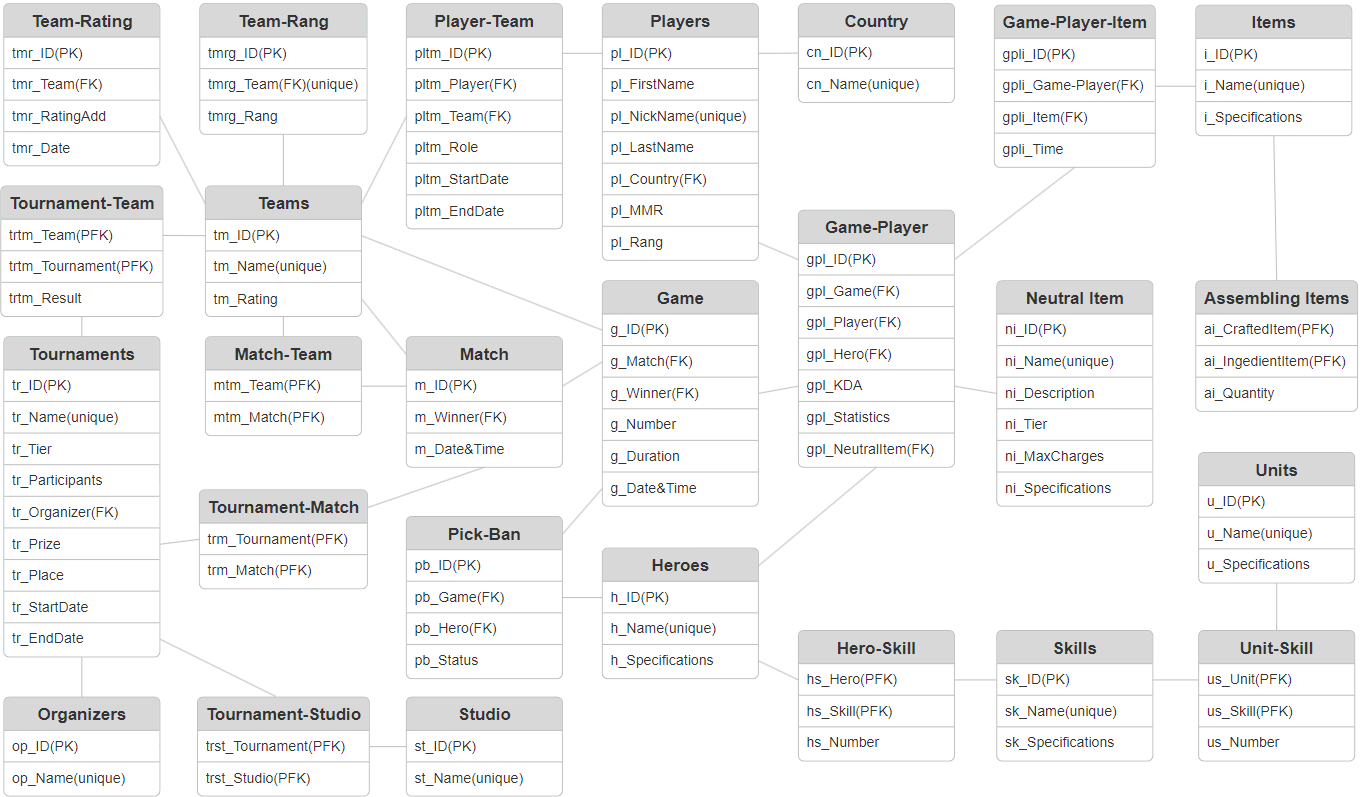


Рисунок . – Инфологическая модель базы данных

## Описание модели

База данных должна содержать в себе информацию о командах, игроках, турнирах, матчах, внутриигровых механиках: героях, предметах, сущностях.

При проектировании базы данных на инфологическом уровне можно выделить следующие сущности и их атрибуты:

1. Таблица, содержащая информацию о странах. Хранит идентификатор страны и название.
2. Таблица, содержащая информацию о игроках. Хранит идентификатор игрока, имя, фамилию, игровое имя, страну игрока, внутриигровой рейтинг игрока, ранг игрока в мировом топе.
3. Таблица, содержащая информацию о командах. Хранит идентификатор команда, название команды, рейтинг команды.
4. Таблица, содержащая информацию о принадлежности игрока к команде. Хранит идентификатор записи, идентификатор игрока, идентификатор команды, роль игрока в команде, дата начала выступления игрока за команду, дата окончания выступления игрока за команду.
5. Таблица, содержащая информацию о ранге команды. Хранит идентификатор записи, идентификатор команды, ранг команды.
6. Таблица, содержащая информацию об обновлении рейтинга команд. Хранит идентификатор записи, идентификатор команды, полученный рейтинг, дата получения рейтинга.
7. Таблица, содержащая информацию о организаторах турниров. Хранит идентификатор организатора, название организатора.
8. Таблица, содержащая информацию о турнирах. Хранит идентификатор турнира, название турнира, уровень турнира, число участников турнира, идентификатор организатора турнира, призовой фонд турнира, место проведения турнира, дата начала турнира, дата окончания турнира.
9. Таблица, содержащая информацию об участии команд на турнирах. Хранит идентификатор турнира, идентификатор команды, участвующей в турнире, результат команды на турнире.
10. Таблица, содержащая информацию о студиях, транслирующих турниры. Хранит идентификатор студии, название студии.
11. Таблица, содержащая информацию о том, какие студии транслируют турнир. Хранит идентификатор турнира, идентификатор студии.
12. Таблица, содержащая информацию о матчах. Хранит идентификатор матча, идентификатор команды-победителя, дата и время начала матча.
13. Таблица, содержащая информацию о принадлежности матча к турниру. Хранит идентификатор турнира, идентификатор матча.
14. Таблица, содержащая информацию о командах участвующих в матче. Хранит идентификатор матча, идентификатор команды.
15. Таблица, содержащая информацию об играх проходящих в рамках матча. Хранит идентификатор игры, идентификатор матча, идентификатор команды-победителя в игре, номер игры в рамках матча, длительность игры, дата и время начала игры.
16. Таблица, содержащая информацию о героях. Хранит идентификатор героя, название героя, основной атрибут героя, описание героя, базовое число очков здоровья, базовое число очков маны, базовое восстановление очков здоровья, базовое восстановление очков маны, базовая сила, базовая ловкость, базовый интеллект, прирост силы за уровень, прирост ловкости за уровень, прирост интеллекта за уровень, базовый урон, базовая скорость передвижения, базовая броня, базовая скорость атаки, тип атаки, базовый интервал между атаками, базовая дальность атаки, усиление заклинаний, базовое сопротивление магии, базовое сопротивление эффектам, базовое уклонение, описание Aghanim’s Scepter, описание Aghanim’s Shard.
17. Таблица, содержащая информацию о выборе и запрете героев в рамках игры. Хранит идентификатор записи, идентификатор игры, идентификатор героя, статус героя.
18. Таблица, содержащая информацию о сущностях. Хранит идентификатор сущности, имя сущности, описание сущности, тип сущности, базовое число очков здоровья, базовое число очков маны, базовое восстановление очков здоровья, базовое восстановление очков маны, базовый урон, базовая скорость передвижения, базовая броня, базовая скорость атаки, тип атаки, базовый интервал между атаками, базовая дальность атаки, базовое сопротивление магии, базовое сопротивление эффектам, золото за убийство, опыт за убийство.
19. Таблица, содержащая информацию о способностях. Хранит идентификатор способности, название способности, описание способности, характеристики способности, стоимость маны, стоимость здоровья, перезарядка, максимальный уровень.
20. Таблица, содержащая информацию о способностях сущностей. Хранит идентификатор сущности, идентификатор способности, номер способности.
21. Таблица, содержащая информацию о способностях героев. Хранит идентификатор героя, идентификатор способности, номер способности.
22. Таблица, содержащая информацию о предметах. Хранит идентификатор предмета, название предмета, описание предмета, цена предмета, тип предмета, максимальное число зарядов, максимальное число предметов в одном слоте, можно ли разобрать предмет, собирается ли предмет, характеристики предмета.
23. Таблица, содержащая информацию о сборке предметов. Хранит идентификатор предмета, который собирается, идентификатор предмета из которого собирается, количество предметов для сборки.
24. Таблица, содержащая информацию о нейтральных предметах. Хранит идентификатор нейтрального предмета, название нейтрального предмета, описание нейтрального предмета, уровень нейтрального предмета, максимальное число зарядов, характеристики.
25. Таблица, содержащая информацию о игроке в игре. Хранит идентификатор записи, идентификатор игры, идентификатор игрока, идентификатор героя, число убийств, число смертей, число помощей, число добитых вражеских крипов, число добитых союзных крипов, общая ценность героя, количество золота в минуту, количество опыта в минуту, идентификатор нейтрального предмета.
26. Таблица, содержащая информацию о покупке игроком предметов в игре. Хранит идентификатор записи, идентификатор записи о игроке в игре, идентификатор предмета, время покупки предмета.

## Сущности и связи

Таблица – Сущности и связи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Отношение (имя на русском языке и латиницей) | Описание | Основные атрибуты | Краткое описание связей с другими отношениями | Особенности |
| county (страна) | Содержит информацию о стране | cn\_id (INT): идентификатор страны  (первичный ключ)  cn\_name (VARCHAR(50)):  название страны | Связано с отношениями: players | cn\_Name является уникальным идентификатором |
| players (игроки) | Содержит информацию о игроке | pl\_id (INT): идентификатор игрока (первичный ключ)  pl\_FirstName (VARCHAR(50)): имя игрока  pl\_NickName (VARCHAR(50)): внутриигровое имя игрока  pl\_LastName (VARCHAR(50)): фамилия игрока  pl\_Country (INT): идентификатор страны  pl\_MMR (INT): рейтинг игрока  pl\_Rang (INT): ранг игрока | Связано с отношениями: country, game-player, player-team | pl\_NickName является уникальным идентификатором |
| teams (команды) | Содержит информацию о командах | tm\_ID (INT): идентификатор команды (первичный ключ)  tm\_Name (VARCHAR(50)): название команды  tm\_Rating (INT): оценка фильма | Связано с отношениями: player-team, team-rang, team-rating, tournament-team, match, match-team, game | tm\_Name является уникальным идентификатором |
| player-team(игрок-команда) | Содержит информацию о принадлежности игрока к команде | pltm\_ID (INT): идентификатор записи (первичный ключ)  pltm\_Player (INT): идентификатор игрока  pltm\_Team (INT): идентификатор команды  pltm\_Role (INT): роль игрока в команде  pltm\_StartDate (DATE): дата начала выступления игрока за команду  pltm\_EndDate (DATE): дата конца выступления игрока за команду | Связано с отношениями: players, teams |  |
| team-rang (команда-ранг) | Содержит инфомацию о ранге команды | tmrg\_ID (INT): идентификатор записи (первичный ключ)  tmrg\_Team (INT): идентификатор команды  tmrg\_Rang (INT): ранг команды | Связано с отношениями: teams | tmrg\_Team является уникальным идентификатором |
| team-rating (команда-рейтинг) | Содержит информацию об обновлении рейтинга команд | tmr\_ID (INT): идентификатор записи (первичный ключ)  tmr\_Team (INT): идентификатор команды  tmr\_RatingAdd (INT): полученный рейтинг  tmr\_Date (DATE): дата получения рейтинга | Связано с отношениями:  teams |  |
| organizers (организоторы) | Содержит информацию о организаторах турниров | op\_ID (INT): идентификатор организатора (первичный ключ)  op\_Name (VARCHAR(50)): название организатора | Связано с отношениями: tournaments | op\_Name является уникальным идентификатором |
| tournaments (турниры) | Содержит информацию о турнирах | tr\_ID (INT): идентификатор турнира (первичный ключ)  tr\_Name (VARCHAR(50)): название турнира  tr\_Tier (INT): уровень турнира  tr\_Participants (INT): число участников турнира  tr\_Organizer (INT): идентификатор организатора турнира  tr\_Prize (INT): призовой фонд турнира  tr\_Place (VARCHAR(50)): место проведения турнира  tr\_StartDate (DATE): дата начала турнира  tr\_EndDate (DATE): дата окончания турнира | Связано с отношениями: tournament-team, tournament-match, tournament-studio, organizers | tr\_Name является уникальным идентификатором |
| Tournament-team (турнир-команда) | Содержит информацию об участии команд на турнирах | trtm\_Tournament (INT): идентификатор турнира (первычный ключ)  trtm\_Team (INT): идентификатор команды (первичный ключ) | Связано с отношениями: teams, tournaments | trmt\_Tournament и trtm\_Team являются составным первичным ключом |
| studio (студия) | Содержит информацию о студиях | st\_ID (INT): идентификатор студии (первичный ключ)  st\_Name (VARCHAR(50)): название студии | Связано с отношениями: tournament-studio | st\_Name является уникальным идентификатором |
| tournament-studio (турнир-студия) | Содержит информацию о том, какие студии транслируют турнир | trst\_Tournament (INT): идентификатор турнира (первичный ключ)  trst\_Studio (INT): идентификатор студии (первичный ключ) | Связано с отношениями: studio, tournaments | trst\_Tournament и trst\_Studio являются составным первичным ключом |
| match (матч) | Содержит информацию о матчах | m\_ID (INT): идентификатор матча (первичный ключ)  m\_Winner (INT): идентификатор команды-победителя  m\_Date&Time (DATETIME): дата и время начала матча | Связано с отношениями: teams, game, tournament-match, match-team |  |
| tournament-match (турнир-матч) | Содержит информацию о принадлежности матча к турниру | trm\_Tournament (INT): идентификатор турнира (первичный ключ)  trm\_Match (INT): идентификатор матча (первичный ключ) | Связано с отношениями: tournaments, match | trm\_Tournament и trm\_Match являются составным первичным ключом |
| match-team(матч-команда) | Содержит информацию о командах участвующих в матче | mtm\_Match (INT): идентификатор матча (первичный ключ)  mtm\_Team (INT): идентификатор команды (первичный ключ) | Связано с отношениями: match, teams | mtm\_Match и mtm\_Team являются составным первичным ключом |
| game (игра) | Содержит информацию об играх проходящих в рамках матча | g\_ID (INT): идентификатор игры (первичный ключ)  g\_Match (INT): идентификатор матча  g\_Number (INT): номер игры в рамках матча  g\_Winner (INT): идентификатор команды-победителя  g\_Duration (TIME): длительность игры  g\_Date&Time (DATETIME): дата и время начала игры | Связано с отношениями:  teams, match, pick-ban, game-player |  |
| heroes (герои) | Содержит информации о героях | h\_ID (INT): идентификатор героя (первичный ключ)  h\_Name (VARCHAR(50)): название героя  h\_Attribute (ENUM): основной атрибут героя  h\_Description (VARCHAR(100)): описание героя  h\_HP (INT): базовое число очков здоровья  h\_MP (INT): базовое число очков маны  h\_HPR (FLOAT): базовое восстановление очков здоровья  h\_MPR (FLOAT): базовое восстановление очков маны  h\_Strength (FLOAT): базовая сила  h\_Agillity (FLOAT): базовая ловкость  h\_Intelligence (FLOAT): базовый интеллект  h\_StrengthPerLVL (FLOAT): прирост силы за уровень  h\_AgillityPerLVL (FLOAT): прирост ловкости за уровень  h\_IntelligencePerLVL (FLOAT): прирост интеллекта за уровень  h\_Damage (INT): базовый урон  h\_MoveSpeed (INT): базовая скорость передвижения  h\_Armor (INT): базовая броня  h\_AttackSpeed (INT): базовая скорость атаки  h\_AttackInterval (FLOAT): базовый интервал между атаками  h\_AttackType (ENUM): тип атаки  h\_AttackRange (INT): базовая дальность атаки  h\_SpellAMP (FLOAT): усиление заклинаний  h\_MagicResist (FLOAT): базовое сопротивление магии  h\_StatusResist (FLOAT): базовое сопротивление эффектам  h\_Evasion (FLOAT): базовое уклонение  h\_Scepter (VARCHAR(5000)): описание Aghanim’s Scepter  h\_Shard: (VARCHAR(5000)): описание Aghanim’s Shard | Связано с отношениями: pick-ban, game-player, hero-skill | h\_Name является уникальным идентификатором |
| units (сущности) | Содержит информацию о сущностях | u\_ID (INT): идентификатор сущности (первичный ключ)  u\_Name (VARCHAR(50)): имя сущности  u\_Description (VARCHAR(100)): описание сущности  u\_Type (ENUM): тип сущности  u\_HP (INT): базовое число очков здоровья  u\_MP (INT): базовое число очков маны  u\_HPR (FLOAT): базовое восстановление очков здоровья  u\_MPR (FLOAT): базовое восстановление очков маны  u\_Damage (INT): базовый урон  u\_MoveSpeed (INT): базовая скорость передвижения  u\_Armor (INT): базовая броня  u\_AttackSpeed (INT): базовая скорость атаки  u\_AttackInterval (FLOAT): базовый интервал между атаками  u\_AttackType (ENUM): тип атаки  u\_AttackRange (INT): базовая дальность атаки  u\_SpellAMP (FLOAT): усиление заклинаний  u\_MagicResist (FLOAT): базовое сопротивление магии  u\_StatusResist (FLOAT): базовое сопротивление эффекта  u\_Gold (INT): золото за убийство  u\_Expirience (INT): опыт за убийство | Связано с отношениями: unit-skill | u\_Name является уникальным идентификатором |
| pick-ban (выбор-запрет) | Содержит информацию о выборе и запрете героев | pb\_ID (INT): идентификатор записи (первичный ключ)  pb\_Game (INT): идентификатор игры  pb\_Hero (INT): идентификатор героя  pb\_Status (ENUM): статус героя | Связано с отношениями: game, heroes, |  |
| skills (способности) | Содержит информацию о способностях | sk\_ID (INT): идентификатор способности (первичный ключ)  sk\_Name (VARCHAR(50)): название способности  sk\_Description (VARCHAR(5000)): описание способности  sk\_Specifications (VARCHAR(500)): характеристики способности  sk\_ManaCost (INT): стоимость маны sk\_HealthCost (INT): стоимость здоровья  sk\_Reaload (FLOAT): перезарядка  sk\_MaxLVL (INT): максимальный уровень | Связано с отношениями: unit-skill, hero-skill | sk\_Name является уникальным идентификатором |
| unit-skill (сущность-способность) | Содержит информацию о способностях сущностей | us\_Unit (INT): идентификатор сущности (первичный ключ)  us\_Skill (INT): идентификатор способности (первичный ключ)  us\_Numer (INT): номер способности | Связано с отношениями: units, skills | us\_Unit и us\_Skill являются составным первичным ключом |
| hero-skill (герой-способность) | Содержит информацию о способностях героев | hs\_Hero (INT): идентификатор героя (первичный ключ)  hs\_Skill (INT): идентификатор способности  hs\_Number (INT): номер способности | Связано с отношениями: heroes, skills | hs\_Hero и hs\_Skill являются составным первичным ключом |
| items (предметы) | Содержит информацию о предметах | i\_ID (INT): идентификатор предмета (первичный ключ)  i\_Name (VARCHAR(50)): название предмета  i\_Description (VARCHAR(500)): описание предмета  i\_Price (INT): цена предмета  i\_Type (ENUM): тип предмета  i\_MaxCharges (INT): максимальное число зарядов  i\_MaxStack (INT): максимальное число предметов в одном слоте  i\_Disassemble (BOOL): можно ли разобрать предмет  i\_Assemble (BOOL): собирается ли предмет  i\_Specifications (VARCHAR(500)): характеристики предмета | Связано с отношениями: assembling items, game-player-item | i\_Name является уникальным идентификатором |
| assembling items (сборка предметов) | Содержит информацию о сборке предметов | ai\_CraftItem (INT): идентификатор предмета, который собирается (первичный ключ)  ai\_IngredientItem (INT): идентификатор предмета из которого собирается (первичный ключ)  ai\_Quantity (INT): количество предметов для сборки | Связано с отношениями: items | ai\_CraftItem и ai\_IngredientItem являются составным первичным ключом |
| neutral item (нейтральные предметы) | Содержит информацию о нейтральных предметах | ni\_ID (INT): идентификатор нейтрального предмета (первичный ключ)  ni\_Name (VARCHAR(50)): название нейтрального предмета  ni\_Description (VARCHAR(500)): описание нейтрального предмета  ni\_Tier (INT): уровень нейтрального предмета  ni\_MaxCharges (INT): максимальное число зарядов ni\_Specifications (VARCHAR(500)): характеристики | Связано с отношениями: game-player | ni\_Name является уникальным идентификатором |
| game-player (игра-игрок) | Содержит информацию о игроке в игре | gpl\_ID (INT): идентификатор записи (первичный ключ)  gpl\_Game (INT): идентификатор игры  gpl\_Player (INT): идентификатор игрока  gpl\_Hero (INT): идентификатор героя  gpl\_Kill (INT): число убийств  gpl\_Dead (INT): число смертей  gpl\_Support (INT): число помощей  gpl\_EnemyCreeps (INT): число добитых вражеских крипов  gpl\_AlliedCreeps (INT): число добитых союзных крипов  gpl\_Nethworth (INT): общая ценность героя  gpl\_GPM (INT): количество золота в минуту  gpl\_EPM (INT): количество опыта в минуту  gpl\_Neutralitem (INT): идентификатор нейтрального предмета | Связано с отношениями: players, neutral item, game, heroes, game-player-item |  |
| game-player-item (игра-игрок-предмет) | Содержит информацию о покупке игроком предметов в игре | gpli\_ID (INT): идентификатор записи (первичный ключ)  gpli\_Game-Player (INT): идентификатор записи о игроке в игре  gpli\_Item (INT): идентификатор предмета  gpli\_Time (TIME): время покупки предмета | Связано с отношениями: game-player, items |  |

## Особенности нормализации

Вынесение информации о стране игрока в отдельную таблицу country позволяет контролировать добавление и изменение данных, так как теперь в поле pl\_Country нельзя записать данные, не находящиеся в таблице country.

Вынесение информации о ранге команды в отдельную таблицу team-rang позволяет обновлять ранги команд, при обновлении рейтинга.

Вынесение информации об организаторах турнира в отдельную таблицу organizers позволяет контролировать добавление и изменение данных, так как теперь в поле tr\_organizer нельзя записать данные, не находящиеся в таблице country.

Вынесение информации о нейтральных предметах в отдельную таблицу Neutral Item позволяет хранить больше информации и контролировать добавление и изменение данных, так как теперь в поле gpl\_NeutralItem нельзя записать данные, не находящиеся в таблице Neutral Item.

# Подробное описание бизнес-логики

## Процедуры

### Обновление уровня турнира

Процедура UpdatePournamentTierProcedure создана для обновления уровня(tr\_Tier) турнира из таблицы tournaments. При вызове, происходит обновление уровня турнира исходя из рангов команд, участвующих на турнире.

CREATE PROCEDURE UpdateTournamentTierProcedure(IN tournamentID INT)

BEGIN

DECLARE participantCount INT;

DECLARE tier1Count INT;

DECLARE tier2Count INT;

DECLARE tier3Count INT;

SELECT `tr\_Participants` INTO participantCount FROM Tournaments WHERE `tr\_ID` = tournamentID;

SELECT COUNT(\*) INTO tier1Count FROM `team-rang` WHERE `tm\_Rang` <= participantCount;

SELECT COUNT(\*) INTO tier2Count FROM `team-rang`WHERE `tm\_Rang` <= (participantCount \* 2);

SELECT COUNT(\*) INTO tier3Count FROM `team-rang`WHERE `tm\_Rang` <= (participantCount \* 4);

IF (tier1Count >= participantCount / 2) THEN

UPDATE Tournaments SET `tr\_Tier` = 1 WHERE `tr\_ID` = tournamentID;

ELSEIF (tier2Count >= participantCount / 2) THEN

UPDATE Tournaments SET `tr\_Tier` = 2 WHERE `tr\_ID` = tournamentID;

ELSEIF (tier3Count >= participantCount / 2) THEN

UPDATE Tournaments SET `tr\_Tier` = 3 WHERE `tr\_ID` = tournamentID;

ELSE

UPDATE Tournaments SET `tr\_Tier` = 4 WHERE `tr\_ID` = tournamentID;

END IF;

END //

### Вычитание рейтинга команд

Процедура RatingDecrease создана для обновления рейтинга команд из таблицы teams. При вызове, у команд, получивших очки рейтинга 1, 2, 3 и 4 месяца назад, вычитается 25% полученных очков рейтинга.

CREATE PROCEDURE RatingDecrease()

BEGIN

DECLARE period1\_start DATE;

DECLARE period1\_end DATE;

DECLARE period2\_start DATE;

DECLARE period2\_end DATE;

DECLARE period3\_start DATE;

DECLARE period3\_end DATE;

DECLARE period4\_start DATE;

DECLARE period4\_end DATE;

SET period1\_start = DATE\_SUB(CURDATE(), INTERVAL 35 DAY);

SET period1\_end = DATE\_SUB(CURDATE(), INTERVAL 29 DAY);

SET period2\_start = DATE\_SUB(CURDATE(), INTERVAL 70 DAY);

SET period2\_end = DATE\_SUB(CURDATE(), INTERVAL 61 DAY);

SET period3\_start = DATE\_SUB(CURDATE(), INTERVAL 105 DAY);

SET period3\_end = DATE\_SUB(CURDATE(), INTERVAL 99 DAY);

SET period4\_start = DATE\_SUB(CURDATE(), INTERVAL 140 DAY);

SET period4\_end = DATE\_SUB(CURDATE(), INTERVAL 134 DAY);

UPDATE `Teams` t

SET `tm\_Rating` = `tm\_Rating` - (0.25 \* (

SELECT SUM(`tmr\_RatingAdd`)

FROM `Team\_Rating` tr

WHERE tr.`tmr\_Team` = t.`tm\_ID`

AND (

(tr.`tmr\_Date` BETWEEN period1\_start AND period1\_end)

OR (tr.`tmr\_Date` BETWEEN period2\_start AND period2\_end)

OR (tr.`tmr\_Date` BETWEEN period3\_start AND period3\_end)

OR (tr.`tmr\_Date` BETWEEN period4\_start AND period4\_end)

)

));

END //

### Прибавление рейтинга команд

Процедура AddRatingPoints создана для обновления рейтинга команд из таблицы teams. При вызове, переданным командам прибавляется рейтинг в зависимости он их результата на турнире.

CREATE PROCEDURE AddRatingPoints(IN team\_id INT, IN tournament\_id INT)

BEGIN

DECLARE place INT;

DECLARE tier INT;

DECLARE rating\_points INT;

SELECT `trtm\_Resul`t INTO place

FROM `m2m\_Tournament-Team`

WHERE `trtm\_Team` = team\_id AND `trtm\_Tournament` = tournament\_id;

SELECT `tr\_Tier` INTO tier

FROM `Tournaments`

WHERE `tr\_ID` = tournament\_id;

CASE tier

WHEN 1 THEN

CASE place

WHEN 1 THEN SET rating\_points = 300;

WHEN 2 THEN SET rating\_points = 180;

WHEN 3 THEN SET rating\_points = 120;

WHEN 4 THEN SET rating\_points = 60;

WHEN 5 THEN SET rating\_points = 40;

WHEN 6 THEN SET rating\_points = 40;

WHEN 7 THEN SET rating\_points = 24;

WHEN 8 THEN SET rating\_points = 24;

WHEN 9 THEN SET rating\_points = 15;

WHEN 10 THEN SET rating\_points = 15;

WHEN 11 THEN SET rating\_points = 15;

WHEN 12 THEN SET rating\_points = 15;

WHEN 13 THEN SET rating\_points = 10;

WHEN 14 THEN SET rating\_points = 10;

WHEN 15 THEN SET rating\_points = 10;

WHEN 16 THEN SET rating\_points = 10;

WHEN 17 THEN SET rating\_points = 6;

WHEN 18 THEN SET rating\_points = 6;

WHEN 19 THEN SET rating\_points = 6;

WHEN 20 THEN SET rating\_points = 6;

ELSE SET rating\_points = 0;

END CASE;

WHEN 2 THEN

CASE place

WHEN 1 THEN SET rating\_points = 100;

WHEN 2 THEN SET rating\_points = 60;

WHEN 3 THEN SET rating\_points = 40;

WHEN 4 THEN SET rating\_points = 24;

WHEN 5 THEN SET rating\_points = 15;

WHEN 6 THEN SET rating\_points = 15;

WHEN 7 THEN SET rating\_points = 10;

WHEN 8 THEN SET rating\_points = 10;

WHEN 9 THEN SET rating\_points = 6;

WHEN 10 THEN SET rating\_points = 6;

WHEN 11 THEN SET rating\_points = 6;

WHEN 12 THEN SET rating\_points = 6;

WHEN 13 THEN SET rating\_points = 4;

WHEN 14 THEN SET rating\_points = 4;

WHEN 15 THEN SET rating\_points = 4;

WHEN 16 THEN SET rating\_points = 4;

WHEN 17 THEN SET rating\_points = 2;

WHEN 18 THEN SET rating\_points = 2;

WHEN 19 THEN SET rating\_points = 2;

WHEN 20 THEN SET rating\_points = 2;

ELSE SET rating\_points = 0;

END CASE;

WHEN 3 THEN

CASE place

WHEN 1 THEN SET rating\_points = 50;

WHEN 2 THEN SET rating\_points = 30;

WHEN 3 THEN SET rating\_points = 20;

WHEN 4 THEN SET rating\_points = 12;

WHEN 5 THEN SET rating\_points = 8;

WHEN 6 THEN SET rating\_points = 8;

WHEN 7 THEN SET rating\_points = 4;

WHEN 8 THEN SET rating\_points = 4;

WHEN 9 THEN SET rating\_points = 2;

WHEN 10 THEN SET rating\_points = 2;

WHEN 11 THEN SET rating\_points = 2;

WHEN 12 THEN SET rating\_points = 2;

WHEN 13 THEN SET rating\_points = 1;

WHEN 14 THEN SET rating\_points = 1;

WHEN 15 THEN SET rating\_points = 1;

WHEN 16 THEN SET rating\_points = 1;

ELSE SET rating\_points = 0;

END CASE;

WHEN 4 THEN

CASE place

WHEN 1 THEN SET rating\_points = 20;

WHEN 2 THEN SET rating\_points = 12;

WHEN 3 THEN SET rating\_points = 8;

WHEN 4 THEN SET rating\_points = 4;

WHEN 5 THEN SET rating\_points = 2;

WHEN 6 THEN SET rating\_points = 2;

WHEN 7 THEN SET rating\_points = 1;

WHEN 8 THEN SET rating\_points = 1;

ELSE SET rating\_points = 0;

END CASE;

END CASE;

UPDATE `Teams`

SET `tm\_Rating` = `tm\_Rating` + rating\_points

WHERE `tm\_ID` = team\_id;

END //

### Получение команд для обновления рейтинга

Процедура GetTeamsToUpdateRating находит команды, участвовавшие в турнире, для обновления их рейтинга.

CREATE PROCEDURE GetTeamsToUpdateRating(IN tournament\_id INT)

BEGIN

DECLARE team\_id INT;

DECLARE done INT DEFAULT 0;

DECLARE teams\_cursor CURSOR FOR

SELECT DISTINCT `trtm\_Team`

FROM `m2m\_Tournament-Team`

WHERE `trtm\_Tournament` = tournament\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN teams\_cursor;

read\_teams: LOOP

FETCH teams\_cursor INTO team\_id;

IF (team\_id IS NULL) THEN

LEAVE read\_teams;

END IF;

CALL AddRatingPoints(team\_id, tournament\_id);

END LOOP;

CLOSE teams\_cursor;

END //

### Получение турниров

Процедура GetTournamentsToUpdateRating находит турниры закончившиеся за указанный промежуток времени, для обновления рейтинга участвовавших команд.

CREATE PROCEDURE GetTournamentsToUpdateRating(IN last\_week\_start DATE, IN last\_week\_end DATE)

BEGIN

DECLARE tournament\_id INT;

DECLARE done INT DEFAULT 0;

DECLARE tournaments\_cursor CURSOR FOR

SELECT `tr\_ID`

FROM `Tournaments`

WHERE `tr\_EndDate` BETWEEN last\_week\_start AND last\_week\_end;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN tournaments\_cursor;

read\_tournaments: LOOP

FETCH tournaments\_cursor INTO tournament\_id;

IF done THEN

LEAVE read\_tournaments;

END IF;

CALL GetTeamsToUpdateRating(tournament\_id);

END LOOP;

CLOSE tournaments\_cursor;

END //

### Обновление рейтинга команд

Процедура UpdateRatingPoints находит находит начало и конец промежутка времени для нахождения турниров.

CREATE PROCEDURE UpdateRatingPoints()

BEGIN

DECLARE last\_week\_start DATE;

DECLARE last\_week\_end DATE;

SET last\_week\_start = DATE\_SUB(CURDATE(), INTERVAL 7 DAY);

SET last\_week\_end = DATE\_SUB(CURDATE(), INTERVAL 1 DAY);

CALL GetTournamentsToUpdateRating(last\_week\_start, last\_week\_end);

END //

## Триггеры

### Сохранение информации об обновлении рейтинга

Триггер TRG\_Logging\_Rating\_Update срабатывает после обновления команды. При обновлении рейтинга команды, проверяется увеличивается ли рейтинг, если да, то в таблицу team-rating заносится информация о команде, рейтинге и дате изменения.

CREATE TRIGGER `TRG\_Logging\_Rating\_Update`

AFTER UPDATE ON `Teams`

FOR EACH ROW

BEGIN

IF NEW.`tm\_Rating` > OLD.`tm\_Rating` THEN

INSERT INTO `Team-Rating` (`tmr\_Team`, `tmr\_RatingAdd`, `tmr\_Date`)

VALUES (NEW.`tm\_ID`, NEW.`tm\_Rating` - OLD.`tm\_Rating`, NOW());

END IF;

END //

### Проверка на максимальный рейтинг

Триггер TRG\_Teams\_Max\_Rating\_Update срабатывает после обновления команды. При обновлении рейтинга команды, если рейтинг превышает 1000, то рейтинг команды становится равен 1000.

CREATE TRIGGER `TRG\_Teams\_Max\_Rating\_Update`

BEFORE UPDATE ON `Teams`

FOR EACH ROW

BEGIN

IF NEW.`tm\_Rating` > 1000 THEN

SET NEW.`tm\_Rating` = 1000;

END IF;

END //

### Обновление рангов команд

Триггер TRG\_Team\_RangUpdate\_Update срабатывает после обновления команды. При обновлении рейтинга команды, в таблице team-rating пересчитываются ранги команд.

CREATE TRIGGER `TRG\_Team\_RangUpdate\_Update`

AFTER UPDATE ON Teams

FOR EACH ROW

BEGIN

UPDATE `Team-Rang` AS tr

JOIN (

SELECT tm\_ID, RANK() OVER (ORDER BY tm\_Rating DESC) AS new\_rank

FROM Teams

) AS ranked\_teams ON tr.tmrg\_Team = ranked\_teams.tm\_ID

SET tr.tmrg\_Rang = ranked\_teams.new\_rank;

END //

### Обновление уровня турнира

Триггер TRG\_m2m\_Tournament-Team\_TeamCount срабатывает после создания записи в таблице m2m\_Tournament-Team. При создании записи проверяется, внесены ли все команды, участвующие в турнире, если да, то уровень турнира обновляется в зависимости от рангов участвующих команд.

CREATE TRIGGER `TRG\_m2m\_Tournament-Team\_TeamCount`

AFTER INSERT ON `m2m\_Tournament-Team`

FOR EACH ROW

BEGIN

DECLARE tournamentID INT;

DECLARE teamCount INT;

SELECT NEW.`trtm\_Tournament`, `tr\_Participants` INTO tournamentID, teamCount

FROM `Tournaments`

WHERE `tr\_ID` = NEW.`trtm\_Tournament`;

IF teamCount = (SELECT COUNT(\*) FROM `m2m\_Tournament-Team` WHERE `trtm\_Tournament` = tournamentID) THEN

CALL UpdateTournamentTierProcedure(tournamentID);

END IF;

END //

# Тестирование базы данных и приложений системы

## Тестирование функций

### Функция: UpdateTournamentTierProcedure (IN tournamentID INT)

Исходные данные: В турнире участвует 10 команд. Ранги команд: 1, 2, 3, 5, 6, 7, 8, 10, 11, 20.

Ожидаемый результат: tr\_Tier = 1

Полученный результат: tr\_Tier = 1

### Функция: AddRatingPoints (IN team\_id INT, IN tournament\_id INT)

Исходные данные: Команда заняла 1 место на tier 1 турнире

Ожидаемый результат: Начисление команде 300 очков рейтинга

Полученный результат: Начисление команде 300 очков рейтинга

### Функция: UpdateRatingPoints()

Исходные данные: запуск процедуры

Ожидаемый результат: обновление рейтинга команд

Полученный результат: обновление рейтинга команд

## Тестирование триггеров

### Триггер: TRG\_Logging\_Rating\_Update

Исходные данные: обновление рейтинга команды

Ожидаемый результат: запись в таблице team-rating о дате и количестве прибавленного рейтинга

Полученный результат: запись в таблице team-rating о дате и количестве прибавленного рейтинга

### Триггер: TRG\_Teams\_Max\_Rating\_Update

Исходные данные: попытка установить команде рейтинг 1100

Ожидаемый результат: рейтинг команды станет равен 1000

Полученный результат: рейтинг команды станет равен 1000

### Триггер: TRG\_Team\_RangUpdate\_Update

Исходные данные: обновление рейтинга команды

Ожидаемый результат: обновление ранга всех команд в таблице team-rang

Полученный результат: обновление ранга всех команд в таблице team-rang

### Триггер: TRG\_m2m\_Tournament-Team\_TeamCount

Исходные данные: добавление последней команды в Tournament-Team

Ожидаемый результат: обновление уровня турнира

Полученный результат: обновление уровня турнира

## Тестирование программного средства

### Загрузка списка команд

Исходный набор данных: запрос на получение команд из бд

Ожидаемый результат: отображение команд

Полученный результат:

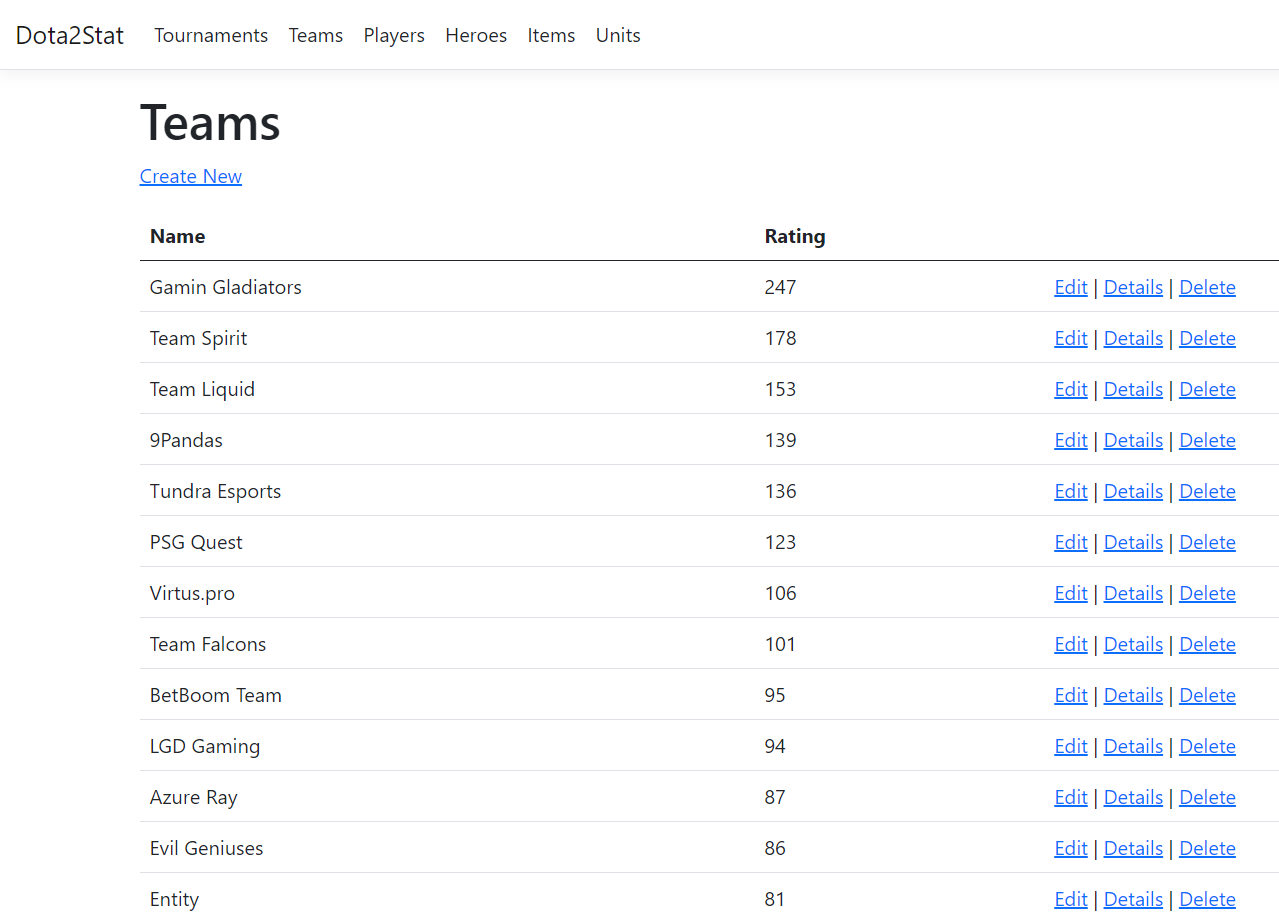


Рисунок 5.3.1 – загрузка списка команд

### Загрузка информации о команде

Исходный набор данных: запрос на получение информации о команде

Ожидаемый результат: отображение информации о команде

Полученный результат:

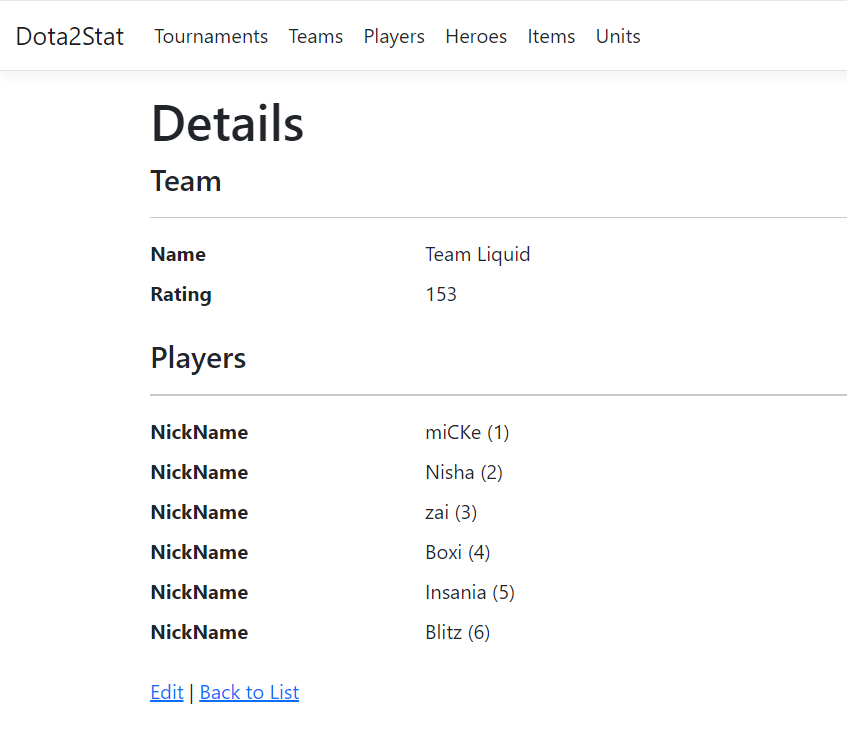


Рисунок 5.3.2 – загрузка информации о команде

### Загрузка списка игроков

Исходный набор данных: запрос на получение списков игроков

Ожидаемый результат: отображение списка игроков

Полученный результат:

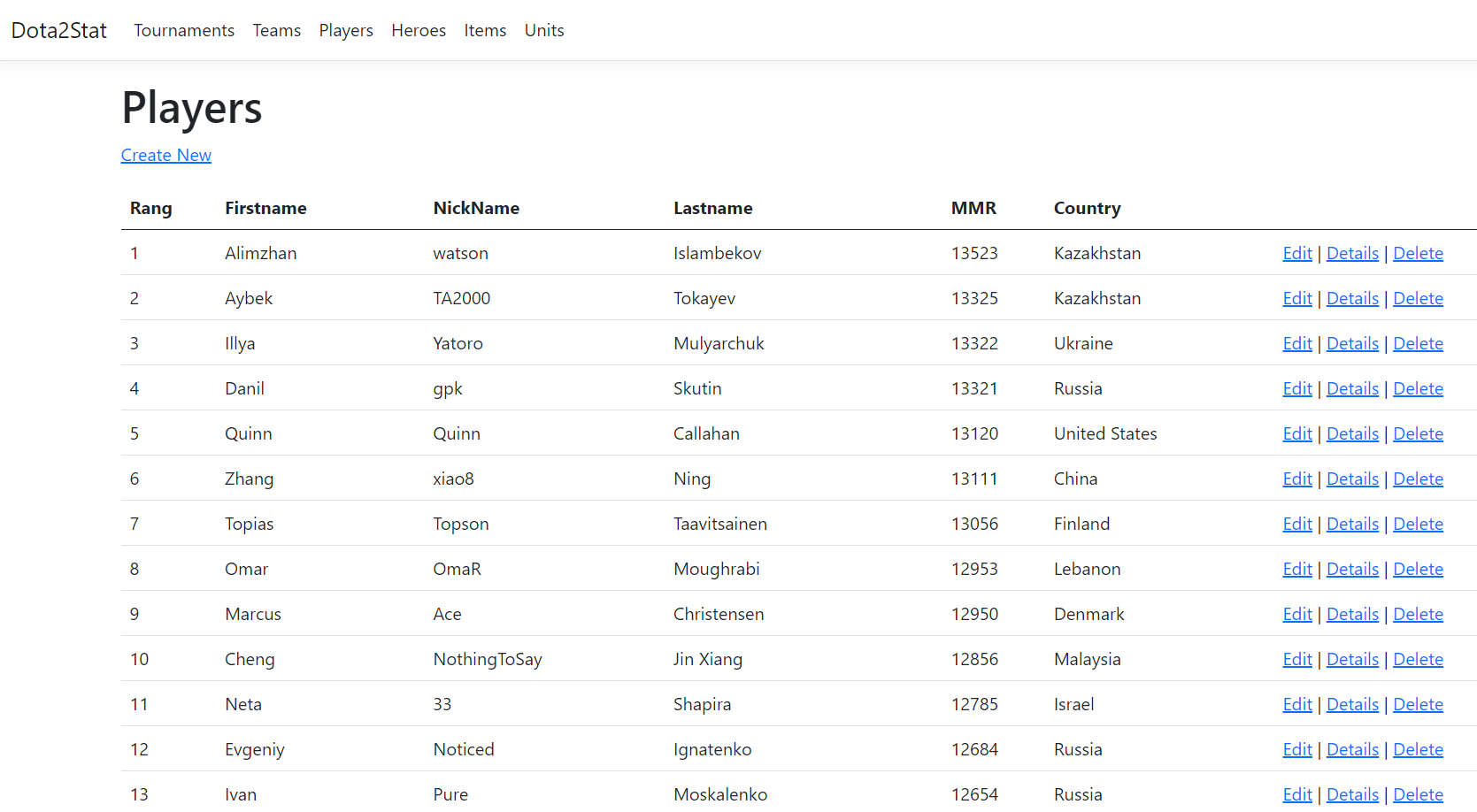


Рисунок 5.3.3 – загрузка информации о игроках

# Заключение

В рамках курсового проекта была разработана база данных для приложения со статистикой матчей по игре Dota2. Данная разработка позволяет хранить и обеспечивать управление информацией о командах, игроках, турнирах, матчах, героях, предметах и других связанных с Dota2 сущностях. Была обеспечена функциональность по вводу, изменению, удалению и извлечению данных, а также поддерживать связи и целостность данных.

Также было разработано приложение для взаимодействия с базой данных. Пользователи могут просматривать информацию о командах, игроках, турнирах, матчах, героях и предметах, получать статистику матчей. Приложение обеспечивает удобный пользовательский интерфейс для просмотра интересующей их информации.

В процессе разработки исследовались различные варианты решения задач, связанных с приложением со статистикой матчей по Dota2. Была изучена модель базы данных, специфичная для Dota2, а также язык запросов для работы с базой данных. В данном проекте в качестве системы управления базами данных (СУБД) был применен MySQL Server, который обеспечивает надежное хранение и эффективное извлечение данных для сервиса просмотра фильмов. MySQL Server предоставляет широкие возможности для моделирования, хранения и запросов данных, что позволяет эффективно работать с информацией о Dota2 и связанных с ними сущностях в базе данных.

В результате были достигнуты поставленные цели, и приложение взаимодействует успешно с базой данных, обеспечивая надежное хранение и управление информацией о Dota2, а также удобный интерфейс для пользователей. Проект прошел все тестирования и готов для использования.

# Список использованной литературы

1. Куликов, Святослав. Реляционные базы данных в примерах. EPAM Systems, 2020–2023. – 424 с.
2. Куликов, Святослав. Работа с MySQL, MS SQL Server и Oracle в примерах. EPAM Systems, 2020–2023. – 590 с.
3. Б. Новиков, Е. Горшкова, Н. Графеева. Основы технологий баз данных: учебное пособие. ДМК Пресс, 2020. – 582 с.
4. Руководство по SQL Server [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/sql/relational-databases/sql-server-guides?view=sql-server-ver16>
5. Руководство по Entity Framework 8 [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/efcore>
6. Руководство по ASP.NET Core 8 [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/aspnet6/>
7. Руководство по ASP.NET Core MVC [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/aspnetmvc/>
8. Обучающие ресурсы по SQL [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/sql/sql-server/educational-sql-resources?view=sql-server-ver16>
9. [Dusan Petkovic](https://www.amazon.com/Dusan-Petkovic/e/B004LREPYA/ref=dp_byline_cont_ebooks_1). Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition. McGraw Hill, 2022. – 865с.

Приложение А

(обязательное)

Схема базы данных на языке SQL

DELIMITER //

SET GLOBAL event\_scheduler = ON//

CREATE EVENT Update\_Rating\_Event

ON SCHEDULE EVERY 7 DAY

STARTS '2001-01-01 00:00:00'

DO

BEGIN

CALL UpdateRatingPoints();

CALL RatingDecrease();

END //

DELIMITER ;

/\* Create Tables \*/

CREATE TABLE `Assembling Items`

(

`ai\_CraftedItem` INT UNSIGNED NOT NULL,

`ai\_IngredientItem` INT UNSIGNED NOT NULL,

`ai\_Quantity` TINYINT UNSIGNED NULL,

CONSTRAINT `PK\_Assembling Items` PRIMARY KEY (`ai\_CraftedItem` ASC, `ai\_IngredientItem` ASC)

);

CREATE TABLE `Country`

(

`cn\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`cn\_Name` VARCHAR(50) NULL,

CONSTRAINT `PK\_Country` PRIMARY KEY (`cn\_ID` ASC)

);

CREATE TABLE `Game`

(

`g\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`g\_Match` INT UNSIGNED NOT NULL,

`g\_Number` TINYINT UNSIGNED NOT NULL,

`g\_Winner` INT UNSIGNED NULL,

`g\_Duration` TIME NULL,

`g\_Date&Time` DATETIME NULL,

CONSTRAINT `PK\_Game` PRIMARY KEY (`g\_ID` ASC)

);

CREATE TABLE `Game-Player`

(

`gpl\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`gpl\_Game` INT UNSIGNED NOT NULL,

`gpl\_Player` INT UNSIGNED NOT NULL,

`gpl\_Hero` INT UNSIGNED NULL,

`gpl\_Kill` INT UNSIGNED NULL,

`gpl\_Dead` INT UNSIGNED NULL,

`gpl\_Support` INT UNSIGNED NULL,

`gpl\_EnemyCreeps` INT UNSIGNED NULL,

`gpl\_AlliedCreeps` INT UNSIGNED NULL,

`gpl\_Nethworth` INT UNSIGNED NULL,

`gpl\_GPM` INT UNSIGNED NULL,

`gpl\_EPM` INT UNSIGNED NULL,

`gpl\_NeutralItem` INT UNSIGNED NULL,

CONSTRAINT `PK\_Game-Player` PRIMARY KEY (`gpl\_ID` ASC)

);

CREATE TABLE `Game-Player-Item`

(

`gpli\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`gpli\_Game-Player` INT UNSIGNED NOT NULL,

`gpli\_Item` INT UNSIGNED NOT NULL,

`gpli\_Time` TIME NOT NULL,

CONSTRAINT `PK\_Game-Gammer-Item` PRIMARY KEY (`gpli\_ID` ASC)

);

CREATE TABLE `Heroes`

(

`h\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'ID героя',

`h\_Name` VARCHAR(50) NULL,

`h\_Attribute` ENUM('strength', 'agility', 'intelligence', 'universal') NULL,

`h\_Description` VARCHAR(100) NULL,

`h\_HP` SMALLINT UNSIGNED NULL,

`h\_MP` SMALLINT UNSIGNED NULL,

`h\_HPR` FLOAT(5,1) UNSIGNED NULL,

`h\_MPR` FLOAT(5,1) UNSIGNED NULL,

`h\_Strength` FLOAT(5,1) UNSIGNED NULL,

`h\_StrengthPerLVL` FLOAT(5,1) UNSIGNED NULL,

`h\_Agility` FLOAT(5,1) UNSIGNED NULL,

`h\_AgilityPerLVL` FLOAT(5,1) UNSIGNED NULL,

`h\_Intelligence` FLOAT(5,1) UNSIGNED NULL,

`h\_IntelligencePerLVL` FLOAT(5,1) UNSIGNED NULL,

`h\_Damage` SMALLINT UNSIGNED NULL,

`h\_MoveSpeed` SMALLINT UNSIGNED NULL,

`h\_Armor` SMALLINT UNSIGNED NULL,

`h\_AttackSpeed` SMALLINT UNSIGNED NULL,

`h\_AttackInterval` FLOAT(5,3) UNSIGNED NULL,

`h\_AttackType` ENUM('melee', 'range') NULL,

`h\_AttackRange` SMALLINT UNSIGNED NULL,

`h\_SpellAMP` FLOAT(4,3) UNSIGNED NULL,

`h\_MagicResist` FLOAT(4,3) UNSIGNED NULL,

`h\_StatusResist` FLOAT(4,3) UNSIGNED NULL,

`h\_Evasion` FLOAT(4,3) UNSIGNED NULL,

`h\_Scepter` VARCHAR(5000) NULL,

`h\_Shard` VARCHAR(5000) NULL,

CONSTRAINT `PK\_Heroes` PRIMARY KEY (`h\_ID` ASC)

);

CREATE TABLE `Items`

(

`i\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'ID предмета',

`i\_Name` VARCHAR(50) NULL,

`i\_Description` VARCHAR(500) NULL,

`i\_Price` SMALLINT UNSIGNED NULL,

`i\_Type` ENUM('Consumables', 'Attributes', 'Equipment', 'Miscellaneous', 'Secret shop', 'Accessories', 'Support', 'Magical', 'Armor', 'Weapons', 'Artifacts') NULL,

`i\_MaxCharges` TINYINT UNSIGNED NULL,

`i\_MaxStack` TINYINT UNSIGNED NULL,

`i\_Disassemble` BOOL NULL,

`i\_Assemble` BOOL NULL,

`i\_Specifications` VARCHAR(500) NULL,

CONSTRAINT `PK\_Items` PRIMARY KEY (`i\_ID` ASC)

);

CREATE TABLE `m2m\_Hero-Skill`

(

`hs\_Hero` INT UNSIGNED NOT NULL,

`hs\_Skill` INT UNSIGNED NOT NULL,

`hs\_Number` SMALLINT UNSIGNED NULL,

CONSTRAINT `PK\_m2m\_Hero-Skill` PRIMARY KEY (`hs\_Hero` ASC, `hs\_Skill` ASC)

);

CREATE TABLE `m2m\_Match-Team`

(

`mtm\_Match` INT UNSIGNED NOT NULL,

`mtm\_Team` INT UNSIGNED NOT NULL,

CONSTRAINT `PK\_m2m\_Match-Team` PRIMARY KEY (`mtm\_Match` ASC, `mtm\_Team` ASC)

);

CREATE TABLE `m2m\_Player-Team`

(

`pltm\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`pltm\_Player` INT UNSIGNED NOT NULL,

`pltm\_Team` INT UNSIGNED NOT NULL,

`pltm\_Role` TINYINT UNSIGNED NULL,

`pltm\_StartDate` DATE NULL,

`pltm\_EndDate` DATE NULL,

CONSTRAINT `PK\_m2m\_Player-Team` PRIMARY KEY (`pltm\_ID` ASC)

);

CREATE TABLE `m2m\_Tournament-Match`

(

`trm\_Tounament` INT UNSIGNED NOT NULL,

`trm\_Match` INT UNSIGNED NOT NULL,

CONSTRAINT `PK\_m2m\_Tournament-Match` PRIMARY KEY (`trm\_Tounament` ASC, `trm\_Match` ASC)

);

CREATE TABLE `m2m\_Tournament-Studio`

(

`trst\_Tournament` INT UNSIGNED NOT NULL,

`trst\_Studio` INT UNSIGNED NOT NULL,

CONSTRAINT `PK\_m2m\_Tournament-Studio` PRIMARY KEY (`trst\_Tournament` ASC, `trst\_Studio` ASC)

);

CREATE TABLE `m2m\_Tournament-Team`

(

`trtm\_Team` INT UNSIGNED NOT NULL,

`trtm\_Tournament` INT UNSIGNED NOT NULL,

`trtm\_Result` TINYINT UNSIGNED NULL,

CONSTRAINT `PK\_m2m\_Tournament-Team` PRIMARY KEY (`trtm\_Team` ASC, `trtm\_Tournament` ASC)

);

CREATE TABLE `Match`

(

`m\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'ID матча',

`m\_Winner` INT UNSIGNED NULL,

`m\_Data&Time` DATETIME NULL,

CONSTRAINT `PK\_Match` PRIMARY KEY (`m\_ID` ASC)

);

CREATE TABLE `Neutral Item`

(

`ni\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`ni\_Name` VARCHAR(50) NULL,

`ni\_Description` VARCHAR(500) NULL,

`ni\_Tier` TINYINT UNSIGNED NULL,

`ni\_MaxCharges` TINYINT UNSIGNED NULL,

`ni\_Specifications` VARCHAR(500) NULL,

CONSTRAINT `PK\_Neutral Item` PRIMARY KEY (`ni\_ID` ASC)

);

CREATE TABLE `Organizers`

(

`op\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`op\_Name` VARCHAR(50) NULL,

CONSTRAINT `PK\_Organizers` PRIMARY KEY (`op\_ID` ASC)

);

CREATE TABLE `Pick-Ban`

(

`pb\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`pb\_Game` INT UNSIGNED NOT NULL,

`pb\_Hero` INT UNSIGNED NOT NULL,

`pb\_Status` ENUM('pick', 'ban') NULL,

CONSTRAINT `PK\_Pick-Ban` PRIMARY KEY (`pb\_ID` ASC)

);

CREATE TABLE `Player`

(

`pl\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`pl\_FirstName` VARCHAR(50) NULL,

`pl\_LastName` VARCHAR(50) NULL,

`pl\_Nickname` VARCHAR(50) NULL,

`pl\_Country` INT UNSIGNED NOT NULL,

`pl\_MMR` INT UNSIGNED NULL,

`pl\_Rang` INT UNSIGNED NULL,

CONSTRAINT `PK\_Player` PRIMARY KEY (`pl\_ID` ASC)

);

CREATE TABLE `Skills`

(

`sk\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`sk\_Name` VARCHAR(50) NULL,

`sk\_Description` VARCHAR(5000) NULL,

`sk\_Specifications` VARCHAR(500) NULL,

`sk\_ManaCost` INT UNSIGNED NULL,

`sk\_HealthCost` INT UNSIGNED NULL,

`sk\_Reaload` FLOAT(4,1) UNSIGNED NULL,

`sk\_MaxLVL` TINYINT UNSIGNED NULL,

CONSTRAINT `PK\_Skills` PRIMARY KEY (`sk\_ID` ASC)

);

CREATE TABLE `Studio`

(

`st\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'ID студии освещения',

`st\_Name` VARCHAR(50) NULL,

CONSTRAINT `PK\_Studio` PRIMARY KEY (`st\_ID` ASC)

);

CREATE TABLE `Team-Rating`

(

`tmr\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`tmr\_Team` INT UNSIGNED NULL,

`tmr\_RatingAdd` SMALLINT UNSIGNED NULL,

`tmr\_Date` DATE NULL,

CONSTRAINT `PK\_Team-Rating` PRIMARY KEY (`tmr\_ID` ASC)

);

CREATE TABLE `Team-Rang`

(

`tmrg\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`tmrg\_Team` INT UNSIGNED NULL,

`tmrg\_Rang` INT UNSIGNED NULL,

CONSTRAINT `PK\_Team-Rang` PRIMARY KEY (`tmrg\_ID` ASC)

);

CREATE TABLE `Teams`

(

`tm\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`tm\_Name` VARCHAR(50) NULL,

`tm\_Rating` INT UNSIGNED NULL,

CONSTRAINT `PK\_Teams` PRIMARY KEY (`tm\_ID` ASC)

);

CREATE TABLE `Tournaments`

(

`tr\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'ID турнира',

`tr\_Name` VARCHAR(50) NULL,

`tr\_Tier` TINYINT UNSIGNED NULL,

`tr\_Participants` SMALLINT UNSIGNED NULL,

`tr\_Organizer` INT UNSIGNED NOT NULL,

`tr\_Prize` INT UNSIGNED NULL,

`tr\_Place` VARCHAR(50) NULL,

`tr\_StartDate` DATE NULL,

`tr\_EndDate` DATE NULL,

CONSTRAINT `PK\_Tournaments` PRIMARY KEY (`tr\_ID` ASC)

);

CREATE TABLE `Units`

(

`u\_ID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`u\_Name` VARCHAR(50) NULL,

`u\_Description` VARCHAR(100) NULL,

`u\_Type` ENUM('structures', 'lane creeps', 'small creeps', 'medium creeps', 'large creeps', 'ancient creeps', 'roshan', 'tormentor', 'observer') NULL,

`u\_HP` SMALLINT UNSIGNED NULL,

`u\_MP` SMALLINT UNSIGNED NULL,

`u\_HPR` FLOAT(5,1) UNSIGNED NULL,

`u\_MPR` FLOAT(5,1) UNSIGNED NULL,

`u\_Damage` SMALLINT UNSIGNED NULL,

`u\_MoveSpeed` SMALLINT UNSIGNED NULL,

`u\_Armor` SMALLINT UNSIGNED NULL,

`u\_AttackSpeed` SMALLINT UNSIGNED NULL,

`u\_AttackInterval` FLOAT(5,3) UNSIGNED NULL,

`u\_AttackType` ENUM('melee', 'reange') NULL,

`u\_AttackRange` SMALLINT UNSIGNED NULL,

`u\_SpellAMP` FLOAT(4,3) UNSIGNED NULL,

`u\_MagicResist` FLOAT(4,3) UNSIGNED NULL,

`u\_StatusResist` FLOAT(4,3) UNSIGNED NULL,

`u\_Gold` INT UNSIGNED NULL,

`u\_Experience` INT UNSIGNED NULL COMMENT 'Опыт за убийство',

CONSTRAINT `PK\_Units` PRIMARY KEY (`u\_ID` ASC)

);

CREATE TABLE `Unit-Skill`

(

`us\_Unit` INT UNSIGNED NOT NULL,

`us\_Skill` INT UNSIGNED NOT NULL,

`us\_Numer` TINYINT UNSIGNED NULL,

CONSTRAINT `PK\_Unit-Skill` PRIMARY KEY (`us\_Skill` ASC, `us\_Unit` ASC)

);

/\* Create Primary Keys, Indexes, Uniques, Checks \*/

ALTER TABLE `Assembling Items`

ADD INDEX `IXFK\_Assembling Items\_Items` (`ai\_CraftedItem` ASC)

;

ALTER TABLE `Assembling Items`

ADD INDEX `IXFK\_Assembling Items\_Items\_02` (`ai\_IngredientItem` ASC)

;

ALTER TABLE `Country`

ADD CONSTRAINT `UNQ\_cn\_Name` UNIQUE (`cn\_Name`)

;

ALTER TABLE `Game`

ADD INDEX `IXFK\_Game\_Match` (`g\_Match` ASC)

;

ALTER TABLE `Game`

ADD INDEX `IXFK\_Game\_Teams` (`g\_Winner` ASC)

;

DELIMITER //

CREATE TRIGGER `TRG\_Game\_CheckWinner\_Insert`

BEFORE INSERT ON `Game`

FOR EACH ROW

BEGIN

DECLARE match\_count INT;

SELECT COUNT(\*) INTO match\_count

FROM `m2m\_Match-Team`

WHERE `mtm\_Match` = NEW.`g\_Match` AND `mtm\_Team` = NEW.`g\_Winner`;

IF match\_count = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'No matching record in m2m\_Match-Team for the specified match and team.';

END IF;

END //

CREATE TRIGGER `TRG\_Game\_CheckWinner\_Update`

BEFORE UPDATE ON `Game`

FOR EACH ROW

BEGIN

DECLARE match\_count INT;

SELECT COUNT(\*) INTO match\_count

FROM `m2m\_Match-Team`

WHERE `mtm\_Match` = NEW.`g\_Match` AND `mtm\_Team` = NEW.`g\_Winner`;

IF match\_count = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'No matching record in m2m\_Match-Team for the specified match and team.';

END IF;

END //

DELIMITER ;

ALTER TABLE `Game-Player`

ADD INDEX `IXFK\_Game-Player\_Game` (`gpl\_Game` ASC)

;

ALTER TABLE `Game-Player`

ADD INDEX `IXFK\_Game-Player\_Hero` (`gpl\_Hero` ASC)

;

ALTER TABLE `Game-Player`

ADD INDEX `IXFK\_Game-Player\_NeutralItem` (`gpl\_NeutralItem` ASC)

;

ALTER TABLE `Game-Player`

ADD INDEX `IXFK\_Game-Player\_Player` (`gpl\_Player` ASC)

;

DELIMITER //

CREATE TRIGGER `TRG\_Game-Player\_CheckHero\_Insert`

BEFORE INSERT ON `Game-Player`

FOR EACH ROW

BEGIN

DECLARE status\_check INT;

SELECT COUNT(\*) INTO status\_check

FROM `Pick\_Ban`

WHERE `pb\_Status` = 'pick' AND `pb\_Hero` = NEW.`gpl\_Hero`;

IF status\_check = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid value for gpl\_Hero. Hero is not picked.';

END IF;

END //

CREATE TRIGGER `TRG\_Game-Player\_CheckHero\_Update`

BEFORE UPDATE ON `Game-Player`

FOR EACH ROW

BEGIN

DECLARE status\_check INT;

SELECT COUNT(\*) INTO status\_check

FROM `Pick\_Ban`

WHERE `pb\_Status` = 'pick' AND `pb\_Hero` = NEW.`gpl\_Hero`;

IF status\_check = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid value for gpl\_Hero. Hero is not picked.';

END IF;

END //

DELIMITER ;

ALTER TABLE `Game-Player-Item`

ADD INDEX `IXFK\_Game-Player-Item\_Game-Player` (`gpli\_Game-Player` ASC)

;

ALTER TABLE `Game-Player-Item`

ADD INDEX `IXFK\_Game-Player-Item\_Items` (`gpli\_Item` ASC)

;

ALTER TABLE `m2m\_Hero-Skill`

ADD INDEX `IXFK\_m2m\_Hero-Skill\_Heroes` (`hs\_Hero` ASC)

;

ALTER TABLE `m2m\_Hero-Skill`

ADD INDEX `IXFK\_m2m\_Hero-Skill\_Skills` (`hs\_Skill` ASC)

;

ALTER TABLE `m2m\_Match-Team`

ADD INDEX `IXFK\_m2m\_Match-Team\_Match` (`mtm\_Match` ASC)

;

ALTER TABLE `m2m\_Match-Team`

ADD INDEX `IXFK\_m2m\_Match-Team\_Teams` (`mtm\_Team` ASC)

;

ALTER TABLE `m2m\_Player-Team`

ADD INDEX `IXFK\_m2m\_Player-Team\_Player` (`pltm\_Player` ASC)

;

ALTER TABLE `m2m\_Player-Team`

ADD INDEX `IXFK\_m2m\_Player-Team\_Teams` (`pltm\_Team` ASC)

;

DELIMITER //

CREATE TRIGGER `TRG\_m2m\_Player-Team\_Check\_Dates\_Insert`

BEFORE INSERT ON `m2m\_Player-Team`

FOR EACH ROW

BEGIN

IF NEW.`pltm\_StartDate` > NEW.`pltm\_EndDate` THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'StartDate cannot be greater than EndDate';

END IF;

END //

CREATE TRIGGER `TRG\_m2m\_Player-Team\_Check\_Dates\_Update`

BEFORE UPDATE ON `m2m\_Player-Team`

FOR EACH ROW

BEGIN

IF NEW.`pltm\_StartDate` > NEW.`pltm\_EndDate` THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'StartDate cannot be greater than EndDate';

END IF;

END //

DELIMITER ;

ALTER TABLE `m2m\_Tournament-Match`

ADD INDEX `IXFK\_m2m\_Tournament-Match\_Match` (`trm\_Match` ASC)

;

ALTER TABLE `m2m\_Tournament-Match`

ADD INDEX `IXFK\_m2m\_Tournament-Match\_Tournaments` (`trm\_Tounament` ASC)

;

ALTER TABLE `m2m\_Tournament-Studio`

ADD INDEX `IXFK\_m2m\_Tournament-Studio\_Studio` (`trst\_Studio` ASC)

;

ALTER TABLE `m2m\_Tournament-Studio`

ADD INDEX `IXFK\_m2m\_Tournament-Studio\_Tournaments` (`trst\_Tournament` ASC)

;

ALTER TABLE `m2m\_Tournament-Team`

ADD INDEX `IXFK\_m2m\_Tournament-Team\_Teams` (`trtm\_Team` ASC)

;

ALTER TABLE `m2m\_Tournament-Team`

ADD INDEX `IXFK\_m2m\_Tournament-Team\_Tournaments` (`trtm\_Tournament` ASC)

;

DELIMITER //

CREATE TRIGGER `TRG\_m2m\_Tournament-Team\_TeamCount`

AFTER INSERT ON `m2m\_Tournament-Team`

FOR EACH ROW

BEGIN

DECLARE tournamentID INT;

DECLARE teamCount INT;

SELECT NEW.`trtm\_Tournament`, `tr\_Participants` INTO tournamentID, teamCount

FROM `Tournaments`

WHERE `tr\_ID` = NEW.`trtm\_Tournament`;

IF teamCount = (SELECT COUNT(\*) FROM `m2m\_Tournament-Team` WHERE `trtm\_Tournament` = tournamentID) THEN

CALL UpdateTournamentTierProcedure(tournamentID);

END IF;

END //

DELIMITER ;

ALTER TABLE `Match`

ADD INDEX `IXFK\_Match\_Teams` (`m\_Winner` ASC)

;

DELIMITER //

CREATE TRIGGER `TRG\_Match\_CheckWinner\_Insert`

BEFORE INSERT ON `Match`

FOR EACH ROW

BEGIN

DECLARE match\_count INT;

SELECT COUNT(\*) INTO match\_count

FROM `m2m\_Match-Team`

WHERE `mtm\_Match` = NEW.`m\_ID` AND `mtm\_Team` = NEW.`m\_Winner`;

IF match\_count = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'No matching record in m2m\_Match-Team for the specified match and team.';

END IF;

END //

CREATE TRIGGER `TRG\_Match\_CheckWinner\_Update`

BEFORE UPDATE ON `Match`

FOR EACH ROW

BEGIN

DECLARE match\_count INT;

команды

SELECT COUNT(\*) INTO match\_count

FROM `m2m\_Match-Team`

WHERE `mtm\_Match` = NEW.`m\_ID` AND `mtm\_Team` = NEW.`m\_Winner`;

IF match\_count = 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'No matching record in m2m\_Match-Team for the specified match and team.';

END IF;

END //

DELIMITER ;

ALTER TABLE `Pick-Ban`

ADD INDEX `IXFK\_Pick-Ban\_Game` (`pb\_Game` ASC)

;

ALTER TABLE `Pick-Ban`

ADD INDEX `IXFK\_Pick-Ban\_Heroes` (`pb\_Hero` ASC)

;

ALTER TABLE `Player`

ADD INDEX `IXFK\_Player\_Country` (`pl\_Country` ASC)

;

ALTER TABLE `Player`

ADD CONSTRAINT `UNQ\_pl\_NickName` UNIQUE (`pl\_NickName`)

;

ALTER TABLE `Team-Rang`

ADD CONSTRAINT `UNQ\_tmrg\_Team` UNIQUE (`tmrg\_Team` ASC)

;

ALTER TABLE `Team-Rang`

ADD INDEX `IXFK\_Team-Rang\_Teams` (`tmrg\_Team` ASC)

;

ALTER TABLE `Team-Rang`

ADD CONSTRAINT `FK\_Team-Rang\_Teams`

FOREIGN KEY (`tmrg\_Team`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Team-Rating`

ADD INDEX `IXFK\_Team-Rating\_Teams` (`tmr\_Team` ASC)

;

ALTER TABLE `Teams`

ADD INDEX `UNQ\_tm\_Name` (`tm\_Name`)

;

DELIMITER //

CREATE TRIGGER `TRG\_Team\_RangUpdate\_Update`

AFTER UPDATE ON Teams

FOR EACH ROW

BEGIN

UPDATE `Team-Rang` AS tr

JOIN (

SELECT tm\_ID, RANK() OVER (ORDER BY tm\_Rating DESC) AS new\_rank

FROM Teams

) AS ranked\_teams ON tr.tmrg\_Team = ranked\_teams.tm\_ID

SET tr.tmrg\_Rang = ranked\_teams.new\_rank;

END //

CREATE TRIGGER `TRG\_Team\_RangUpdate\_Insert`

AFTER INSERT ON Teams

FOR EACH ROW

BEGIN

UPDATE `Team-Rang` AS tr

JOIN (

SELECT tm\_ID, RANK() OVER (ORDER BY tm\_Rating DESC) AS new\_rank

FROM Teams

) AS ranked\_teams ON tr.tmrg\_Team = ranked\_teams.tm\_ID

SET tr.tmrg\_Rang = ranked\_teams.new\_rank;

END //

CREATE TRIGGER `TRG\_Teams\_Max\_Rating\_Insert`

BEFORE INSERT ON `Teams`

FOR EACH ROW

BEGIN

IF NEW.`tm\_Rating` > 1000 THEN

SET NEW.`tm\_Rating` = 1000;

END IF;

END //

CREATE TRIGGER `TRG\_Teams\_Max\_Rating\_Update`

BEFORE UPDATE ON `Teams`

FOR EACH ROW

BEGIN

IF NEW.`tm\_Rating` > 1000 THEN

SET NEW.`tm\_Rating` = 1000;

END IF;

END //

CREATE TRIGGER `TRG\_Logging\_Rating\_Insert`

AFTER INSERT ON `Teams`

FOR EACH ROW

BEGIN

IF NEW.`tm\_Rating` > 0 THEN

INSERT INTO `Team-Rating` (`tmr\_Team`, `tmr\_RatingAdd`, `tmr\_Date`)

VALUES (NEW.`tm\_ID`, NEW.`tm\_Rating`, NOW());

END IF;

END //

CREATE TRIGGER `TRG\_Logging\_Rating\_Update`

AFTER UPDATE ON `Teams`

FOR EACH ROW

BEGIN

IF NEW.`tm\_Rating` > OLD.`tm\_Rating` THEN

INSERT INTO `Team-Rating` (`tmr\_Team`, `tmr\_RatingAdd`, `tmr\_Date`)

VALUES (NEW.`tm\_ID`, NEW.`tm\_Rating` - OLD.`tm\_Rating`, NOW());

END IF;

END //

DELIMITER ;

ALTER TABLE `Tournaments`

ADD INDEX `IXFK\_Tournaments\_Organizers` (`tr\_Organizer` ASC)

;

ALTER TABLE `Tournaments`

ADD INDEX `UNQ\_tr\_Name` (`tr\_Name`)

;

DELIMITER //

CREATE TRIGGER `TRG\_Tournaments\_Check\_Dates\_Insert`

BEFORE INSERT ON `Tournaments`

FOR EACH ROW

BEGIN

IF NEW.`tr\_StartDate` > NEW.`tr\_EndDate` THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'StartDate cannot be greater than EndDate';

END IF;

END //

CREATE TRIGGER `TRG\_Tournament\_Check\_Dates\_Update`

BEFORE UPDATE ON `Tournaments`

FOR EACH ROW

BEGIN

IF NEW.`tr\_StartDate` > NEW.`tr\_EndDate` THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'StartDate cannot be greater than EndDate';

END IF;

END //

DELIMITER ;

ALTER TABLE `Studio`

ADD INDEX `UNQ\_st\_Name` (`st\_Name`)

;

ALTER TABLE `Organizers`

ADD INDEX `UNQ\_op\_Name` (`op\_Name`)

;

ALTER TABLE `heroes`

ADD INDEX `UNQ\_h\_Name` (`h\_Name`)

;

ALTER TABLE `skills`

ADD INDEX `UNQ\_sk\_Name` (`sk\_Name`)

;

ALTER TABLE `units`

ADD INDEX `UNQ\_u\_Name` (`u\_Name`)

;

ALTER TABLE `items`

ADD INDEX `UNQ\_i\_Name` (`i\_Name`)

;

ALTER TABLE `neutral item`

ADD INDEX `UNQ\_ni\_Name` (`ni\_Name`)

;

ALTER TABLE `Unit-Skill`

ADD INDEX `IXFK\_Unit-Skill\_Skills` (`us\_Skill` ASC)

;

ALTER TABLE `Unit-Skill`

ADD INDEX `IXFK\_Unit-Skill\_Units` (`us\_Unit` ASC)

;

/\* Create Foreign Key Constraints \*/

ALTER TABLE `Assembling Items`

ADD CONSTRAINT `FK\_Assembling Items\_Items`

FOREIGN KEY (`ai\_CraftedItem`) REFERENCES `Items` (`i\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Assembling Items`

ADD CONSTRAINT `FK\_Assembling Items\_Items\_02`

FOREIGN KEY (`ai\_IngredientItem`) REFERENCES `Items` (`i\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game`

ADD CONSTRAINT `FK\_Game\_Match`

FOREIGN KEY (`g\_Match`) REFERENCES `Match` (`m\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game`

ADD CONSTRAINT `FK\_Game\_Teams`

FOREIGN KEY (`g\_Winner`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game-Player`

ADD CONSTRAINT `FK\_Game-Player\_Game`

FOREIGN KEY (`gpl\_Game`) REFERENCES `Game` (`g\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game-Player`

ADD CONSTRAINT `FK\_Game-Player\_Hero`

FOREIGN KEY (`gpl\_Hero`) REFERENCES `Heroes` (`h\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game-Player`

ADD CONSTRAINT `FK\_Game-Player\_Neutral Item`

FOREIGN KEY (`gpl\_NeutralItem`) REFERENCES `Neutral Item` (`ni\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game-Player`

ADD CONSTRAINT `FK\_Game-Player\_Player`

FOREIGN KEY (`gpl\_Player`) REFERENCES `Player` (`pl\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game-Player-Item`

ADD CONSTRAINT `FK\_Game-Player-Item\_Game-Player`

FOREIGN KEY (`gpli\_Game-Player`) REFERENCES `Game-Player` (`gpl\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Game-Player-Item`

ADD CONSTRAINT `FK\_Game-Player-Item\_Items`

FOREIGN KEY (`gpli\_Item`) REFERENCES `Items` (`i\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Hero-Skill`

ADD CONSTRAINT `FK\_m2m\_Hero-Skill\_Heroes`

FOREIGN KEY (`hs\_Hero`) REFERENCES `Heroes` (`h\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Hero-Skill`

ADD CONSTRAINT `FK\_m2m\_Hero-Skill\_Skills`

FOREIGN KEY (`hs\_Skill`) REFERENCES `Skills` (`sk\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Match-Team`

ADD CONSTRAINT `FK\_m2m\_Match-Team\_Match`

FOREIGN KEY (`mtm\_Match`) REFERENCES `Match` (`m\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Match-Team`

ADD CONSTRAINT `FK\_m2m\_Match-Team\_Teams`

FOREIGN KEY (`mtm\_Team`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Player-Team`

ADD CONSTRAINT `FK\_m2m\_Player-Team\_Player`

FOREIGN KEY (`pltm\_Player`) REFERENCES `Player` (`pl\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Player-Team`

ADD CONSTRAINT `FK\_m2m\_Player-Team\_Teams`

FOREIGN KEY (`pltm\_Team`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Tournament-Match`

ADD CONSTRAINT `FK\_m2m\_Tournament-Match\_Match`

FOREIGN KEY (`trm\_Match`) REFERENCES `Match` (`m\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Tournament-Match`

ADD CONSTRAINT `FK\_m2m\_Tournament-Match\_Tournaments`

FOREIGN KEY (`trm\_Tounament`) REFERENCES `Tournaments` (`tr\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Tournament-Studio`

ADD CONSTRAINT `FK\_m2m\_Tournament-Studio\_Studio`

FOREIGN KEY (`trst\_Studio`) REFERENCES `Studio` (`st\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Tournament-Studio`

ADD CONSTRAINT `FK\_m2m\_Tournament-Studio\_Tournaments`

FOREIGN KEY (`trst\_Tournament`) REFERENCES `Tournaments` (`tr\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Tournament-Team`

ADD CONSTRAINT `FK\_m2m\_Tournament-Team\_Teams`

FOREIGN KEY (`trtm\_Team`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `m2m\_Tournament-Team`

ADD CONSTRAINT `FK\_m2m\_Tournament-Team\_Tournaments`

FOREIGN KEY (`trtm\_Tournament`) REFERENCES `Tournaments` (`tr\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Match`

ADD CONSTRAINT `FK\_Match\_Teams`

FOREIGN KEY (`m\_Winner`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Pick-Ban`

ADD CONSTRAINT `FK\_Pick-Ban\_Game`

FOREIGN KEY (`pb\_Game`) REFERENCES `Game` (`g\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Pick-Ban`

ADD CONSTRAINT `FK\_Pick-Ban\_Heroes`

FOREIGN KEY (`pb\_Hero`) REFERENCES `Heroes` (`h\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Player`

ADD CONSTRAINT `FK\_Player\_Country`

FOREIGN KEY (`pl\_Country`) REFERENCES `Country` (`cn\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Team-Rating`

ADD CONSTRAINT `FK\_Team-Rating\_Teams`

FOREIGN KEY (`tmr\_Team`) REFERENCES `Teams` (`tm\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Tournaments`

ADD CONSTRAINT `FK\_Tournaments\_Organizers`

FOREIGN KEY (`tr\_Organizer`) REFERENCES `Organizers` (`op\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Unit-Skill`

ADD CONSTRAINT `FK\_Unit-Skill\_Skills`

FOREIGN KEY (`us\_Skill`) REFERENCES `Skills` (`sk\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

ALTER TABLE `Unit-Skill`

ADD CONSTRAINT `FK\_Unit-Skill\_Units`

FOREIGN KEY (`us\_Unit`) REFERENCES `Units` (`u\_ID`) ON DELETE Cascade ON UPDATE Cascade

;

SET FOREIGN\_KEY\_CHECKS=1

;

/\* Create Stored Procedures \*/

DELIMITER //

CREATE PROCEDURE AddRatingPoints(IN team\_id INT, IN tournament\_id INT)

BEGIN

DECLARE place INT;

DECLARE tier INT;

DECLARE rating\_points INT;

SELECT `trtm\_Resul`t INTO place

FROM `m2m\_Tournament-Team`

WHERE `trtm\_Team` = team\_id AND `trtm\_Tournament` = tournament\_id;

SELECT `tr\_Tier` INTO tier

FROM `Tournaments`

WHERE `tr\_ID` = tournament\_id;

CASE tier

WHEN 1 THEN

CASE place

WHEN 1 THEN SET rating\_points = 300;

WHEN 2 THEN SET rating\_points = 180;

WHEN 3 THEN SET rating\_points = 120;

WHEN 4 THEN SET rating\_points = 60;

WHEN 5 THEN SET rating\_points = 40;

WHEN 6 THEN SET rating\_points = 40;

WHEN 7 THEN SET rating\_points = 24;

WHEN 8 THEN SET rating\_points = 24;

WHEN 9 THEN SET rating\_points = 15;

WHEN 10 THEN SET rating\_points = 15;

WHEN 11 THEN SET rating\_points = 15;

WHEN 12 THEN SET rating\_points = 15;

WHEN 13 THEN SET rating\_points = 10;

WHEN 14 THEN SET rating\_points = 10;

WHEN 15 THEN SET rating\_points = 10;

WHEN 16 THEN SET rating\_points = 10;

WHEN 17 THEN SET rating\_points = 6;

WHEN 18 THEN SET rating\_points = 6;

WHEN 19 THEN SET rating\_points = 6;

WHEN 20 THEN SET rating\_points = 6;

ELSE SET rating\_points = 0;

END CASE;

WHEN 2 THEN

CASE place

WHEN 1 THEN SET rating\_points = 100;

WHEN 2 THEN SET rating\_points = 60;

WHEN 3 THEN SET rating\_points = 40;

WHEN 4 THEN SET rating\_points = 24;

WHEN 5 THEN SET rating\_points = 15;

WHEN 6 THEN SET rating\_points = 15;

WHEN 7 THEN SET rating\_points = 10;

WHEN 8 THEN SET rating\_points = 10;

WHEN 9 THEN SET rating\_points = 6;

WHEN 10 THEN SET rating\_points = 6;

WHEN 11 THEN SET rating\_points = 6;

WHEN 12 THEN SET rating\_points = 6;

WHEN 13 THEN SET rating\_points = 4;

WHEN 14 THEN SET rating\_points = 4;

WHEN 15 THEN SET rating\_points = 4;

WHEN 16 THEN SET rating\_points = 4;

WHEN 17 THEN SET rating\_points = 2;

WHEN 18 THEN SET rating\_points = 2;

WHEN 19 THEN SET rating\_points = 2;

WHEN 20 THEN SET rating\_points = 2;

ELSE SET rating\_points = 0;

END CASE;

WHEN 3 THEN

CASE place

WHEN 1 THEN SET rating\_points = 50;

WHEN 2 THEN SET rating\_points = 30;

WHEN 3 THEN SET rating\_points = 20;

WHEN 4 THEN SET rating\_points = 12;

WHEN 5 THEN SET rating\_points = 8;

WHEN 6 THEN SET rating\_points = 8;

WHEN 7 THEN SET rating\_points = 4;

WHEN 8 THEN SET rating\_points = 4;

WHEN 9 THEN SET rating\_points = 2;

WHEN 10 THEN SET rating\_points = 2;

WHEN 11 THEN SET rating\_points = 2;

WHEN 12 THEN SET rating\_points = 2;

WHEN 13 THEN SET rating\_points = 1;

WHEN 14 THEN SET rating\_points = 1;

WHEN 15 THEN SET rating\_points = 1;

WHEN 16 THEN SET rating\_points = 1;

ELSE SET rating\_points = 0;

END CASE;

WHEN 4 THEN

CASE place

WHEN 1 THEN SET rating\_points = 20;

WHEN 2 THEN SET rating\_points = 12;

WHEN 3 THEN SET rating\_points = 8;

WHEN 4 THEN SET rating\_points = 4;

WHEN 5 THEN SET rating\_points = 2;

WHEN 6 THEN SET rating\_points = 2;

WHEN 7 THEN SET rating\_points = 1;

WHEN 8 THEN SET rating\_points = 1;

ELSE SET rating\_points = 0;

END CASE;

END CASE;

UPDATE `Teams`

SET `tm\_Rating` = `tm\_Rating` + rating\_points

WHERE `tm\_ID` = team\_id;

END //

CREATE PROCEDURE RatingDecrease()

BEGIN

DECLARE period1\_start DATE;

DECLARE period1\_end DATE;

DECLARE period2\_start DATE;

DECLARE period2\_end DATE;

DECLARE period3\_start DATE;

DECLARE period3\_end DATE;

DECLARE period4\_start DATE;

DECLARE period4\_end DATE;

SET period1\_start = DATE\_SUB(CURDATE(), INTERVAL 35 DAY);

SET period1\_end = DATE\_SUB(CURDATE(), INTERVAL 29 DAY);

SET period2\_start = DATE\_SUB(CURDATE(), INTERVAL 70 DAY);

SET period2\_end = DATE\_SUB(CURDATE(), INTERVAL 61 DAY);

SET period3\_start = DATE\_SUB(CURDATE(), INTERVAL 105 DAY);

SET period3\_end = DATE\_SUB(CURDATE(), INTERVAL 99 DAY);

SET period4\_start = DATE\_SUB(CURDATE(), INTERVAL 140 DAY);

SET period4\_end = DATE\_SUB(CURDATE(), INTERVAL 134 DAY);

UPDATE `Teams` t

SET `tm\_Rating` = `tm\_Rating` - (0.25 \* (

SELECT SUM(`tmr\_RatingAdd`)

FROM `Team\_Rating` tr

WHERE tr.`tmr\_Team` = t.`tm\_ID`

AND (

(tr.`tmr\_Date` BETWEEN period1\_start AND period1\_end)

OR (tr.`tmr\_Date` BETWEEN period2\_start AND period2\_end)

OR (tr.`tmr\_Date` BETWEEN period3\_start AND period3\_end)

OR (tr.`tmr\_Date` BETWEEN period4\_start AND period4\_end)

)

));

END //

CREATE PROCEDURE GetTeamsToUpdateRating(IN tournament\_id INT)

BEGIN

DECLARE team\_id INT;

DECLARE done INT DEFAULT 0;

DECLARE teams\_cursor CURSOR FOR

SELECT DISTINCT `trtm\_Team`

FROM `m2m\_Tournament-Team`

WHERE `trtm\_Tournament` = tournament\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN teams\_cursor;

read\_teams: LOOP

FETCH teams\_cursor INTO team\_id;

IF (team\_id IS NULL) THEN

LEAVE read\_teams;

END IF;

CALL AddRatingPoints(team\_id, tournament\_id);

END LOOP;

CLOSE teams\_cursor;

END //

CREATE PROCEDURE GetTournamentsToUpdateRating(IN last\_week\_start DATE, IN last\_week\_end DATE)

BEGIN

DECLARE tournament\_id INT;

DECLARE done INT DEFAULT 0;

DECLARE tournaments\_cursor CURSOR FOR

SELECT `tr\_ID`

FROM `Tournaments`

WHERE `tr\_EndDate` BETWEEN last\_week\_start AND last\_week\_end;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN tournaments\_cursor;

read\_tournaments: LOOP

FETCH tournaments\_cursor INTO tournament\_id;

IF done THEN

LEAVE read\_tournaments;

END IF;

CALL GetTeamsToUpdateRating(tournament\_id);

END LOOP;

CLOSE tournaments\_cursor;

END //

CREATE PROCEDURE UpdateRatingPoints()

BEGIN

DECLARE last\_week\_start DATE;

DECLARE last\_week\_end DATE;

SET last\_week\_start = DATE\_SUB(CURDATE(), INTERVAL 7 DAY);

SET last\_week\_end = DATE\_SUB(CURDATE(), INTERVAL 1 DAY);

CALL GetTournamentsToUpdateRating(last\_week\_start, last\_week\_end);

END //

CREATE PROCEDURE UpdateTournamentTierProcedure(IN tournamentID INT)

BEGIN

DECLARE participantCount INT;

DECLARE tier1Count INT;

DECLARE tier2Count INT;

DECLARE tier3Count INT;

SELECT `tr\_Participants` INTO participantCount FROM Tournaments WHERE `tr\_ID` = tournamentID;

SELECT COUNT(\*) INTO tier1Count FROM `team-rang` WHERE `tmrg\_Rang` <= participantCount;

SELECT COUNT(\*) INTO tier2Count FROM `team-rang` WHERE `tmrg\_Rang` <= (participantCount \* 2);

SELECT COUNT(\*) INTO tier3Count FROM `team-rang` WHERE `tmrg\_Rang` <= (participantCount \* 4);

IF (tier1Count >= participantCount / 2) THEN

UPDATE Tournaments SET `tr\_Tier` = 1 WHERE `tr\_ID` = tournamentID;

ELSEIF (tier2Count >= participantCount / 2) THEN

UPDATE Tournaments SET `tr\_Tier` = 2 WHERE `tr\_ID` = tournamentID;

ELSEIF (tier3Count >= participantCount / 2) THEN

UPDATE Tournaments SET `tr\_Tier` = 3 WHERE `tr\_ID` = tournamentID;

ELSE

UPDATE Tournaments SET `tr\_Tier` = 4 WHERE `tr\_ID` = tournamentID;

END IF;

END //

DELIMITER ;