РЕФЕРАТ

Расчетно-пояснительная записка 46 с., 19 рис., 10 лист., 6 ист., 2 прил.

БАЗА ДАННЫХ, WINFORMS, .NET, РЕЛЯЦИОННЫЕ, PostgreSQL

Целью данной курсовой работы является разработка базы данных для огранизации футбольных турниров.

В процессе работы былы веделены роли пользователей. Были определены все сущности, хрянящие в базе данных. Были выбраны технологии для решения поставленной задачи. Был разработан триггер, который автоматически пересчитывает рейтинги туниров при добавлении новых оценок. Была разработана функция, составляющая турнирное расписание. Для обеспечения доступа к базе данных была разработана соответствующая программа.

Проведено исследование быстродействия функции на строне базы данных при большом обьеме данных. Из результатов исследования следует, что время выполнения функции увеличивается при увеличении объема данных.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время футбол является одним из самых популярных и распространённых видов спорта. Ежегодно миллионы людей по всему миру участвуют в как в любительских, так и в профессиональных турнирах. С ростом интереса к этому виду спорта возникает необходимость информационной системы для турнира: регистрации турнира, составления расписания турнира, просмотра расписания турнира, внесения рузультата матча, просмотра статистики турнира.

Целью данной курсовой работы является разработка базы данных для организации футбольных турниров.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ предметной области в сфере организации футбольных турниров;
- выделить роли пользователей проектируемого приложения;
- спроектировать базу данных, описать ее сущности и связи, так же спроектировать триггер;
- спроектировать ролевую модель на уровне базы данных;
- создать программное обеспечение, обеспечивающее доступ к спроектированной базе данных;
- провести замеры временных характеристик српоектированной базы данных.

1 Аналитический раздел

1.1 Анализ предметной области

Футбольный турнир — это соревнование, в котором несколько команд сражаются между собой, чтобы определить лучшую из них [1].

Основная задача футбольного турнира заключается в создании среды для соревнования между командами с целью определения победителя и награждения его призом или званием чемпиона.

Турниры по футболу обычно проводятся в формате сезона, который на протяжении определенного периода времени, например, нескольких месяцев или года.

Основными правилами турнира по футболу являются:

1. Турнир может проводиться по круговой форме (каждая команда играет со всеми остальными командами один или два раза).
2. На групповом этапе команды ранжируются на основе очков, полученных в матчах (победы дают 3 очка, ничьи — 1 очко, поражения не дают очков).
3. Если две или более команды имеют одинаковое количество очков, для определения рейтинга будут использоваться второстепенные критерии, такие как разница мячей, количество забитых голов или результаты прямых матчей.
4. Оргкомитет спланирует и объявит график соревнований до начала турнира. Этот календарь включает время и место проведения матчей и может быть скорректирован при необходимости по таким причинам, как плохая погода или проблемы безопасности.

1.2 Постановка задачи

Разработка удобной платформы для футбольных сообществ, позволяя создавать и управлять тунирами с минимальными усилями. Там пользователи могут легко создавать турнир, состовлять расписание матчей, вводить результаты... Необходимо создать приложение, чтобы упростить процесс управления футбольнами турнирами.

На рисунке 1 приведены IDEF0-схемы для поставленной задачи.

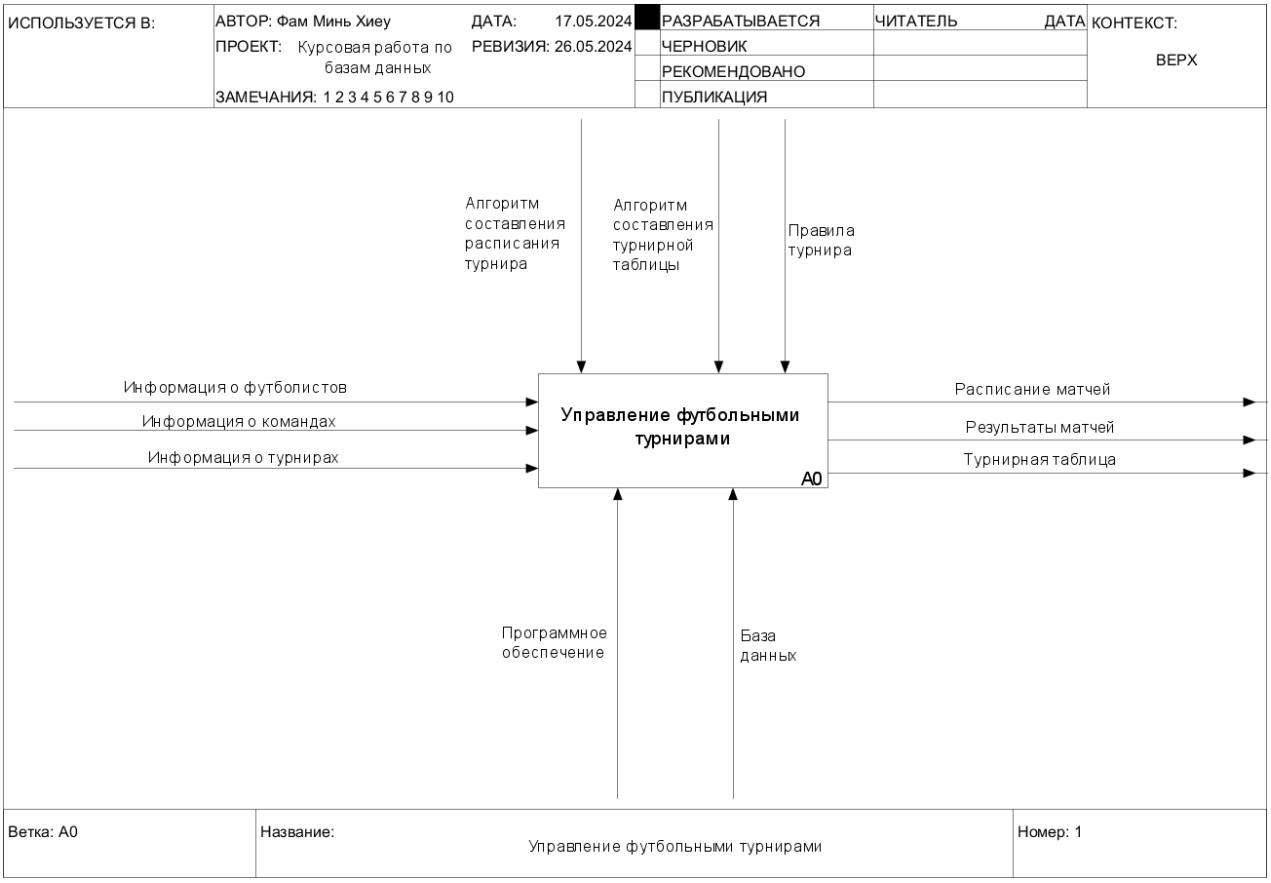

Рисунок 1 – Контекстная диаграмма (A-0)

1.3 Анализ баз данных по модели данных

Модель данных — совокупность правил порождения структур данных в базе данных, операций над ними, а также ограничений целостности, опре-

деляющих допустимые связи и значения данных, последовательность их изменений [2].

По модели баз данных рассматривается три основных типа:

1. Дореляционные.
2. Реляционные.
3. Постреляционные.

1.3.1 Дореляционные

Иерархическая и сетевая модели баз данных входят в категорию дореляционных моделей данных. Иерархическая модель организует данные в виде дерева с отношениями предок-потомок.

На рисунке 2 приведена схема иерархической модели.

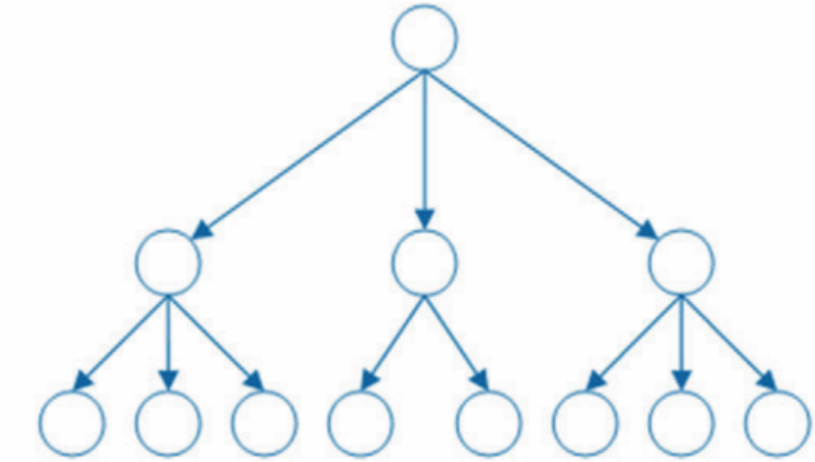

Рисунок 2 – Схема иерархической модели [2]

Эта модель подразумевает, что каждая запись имеет одного родителя и несколько детей. Связи между записями реализуются через физические указатели. Недостатком является невозможность реализации отношений многие ко многим и ситуаций, когда у записи несколько предков.

Сетевая модель организует данные в виде графа, в которой у дочерней записи может быть несколько родителей. Она отличается от иерархической

модели тем, что разрешает такие множественные связи. Основным недостатком сетевой модели данных является ее жесткость и сложность, которые проявляются в построении базы данных на основе этой модели.

Поскольку логика выбора данных привязана к физической организации этих данных, сетевая модель не обладает полной независимостью от приложения. Другими словами, любые изменения в структуре данных потребуют соответствующих изменений в самом приложении.

На рисунке 3 приведена схема сетевой модели.

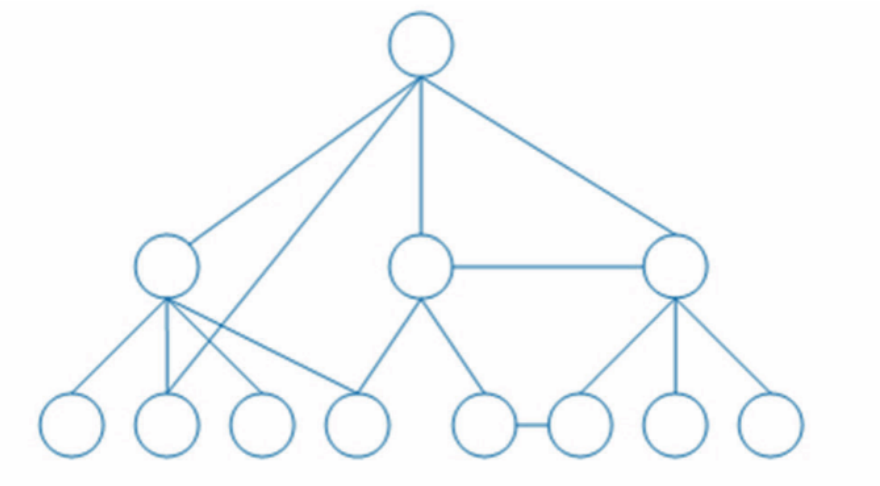

Рисунок 3 – Схема сетевой модели [2]

1.3.2 Реляционные

Реляционная модель данных включает в себя следующие компоненты:

— структурный компонент — данные организованы в базе данных в виде набора отношений, представленных в виде таблиц;

— целостностный компонент — отношения (или таблицы) в базе данных должны соответствовать определенным условиям целостности, чтобы гарантировать правильность и непротиворечивость данных.

— манипуляционный компонент — для выполнения операций над данными используются средства реляционной алгебры и реляционного исчисле-

ния.

1.3.3 Постреляционные

Это расширенная версия реляционной модели, которая позволяет хранить множественные значения в полях данных и разделять их на подзначения. Однако, сложность обеспечения целостности данных является ее главным недостатком.

На рисунке 4 приведены реляционная и постреляционная модели данных для одной и той же предметной области.

Код товара	Продавец
0011	ООО «ТриО»
0023	ООО «Бюро»
0042	ООО «Бирка»

Код товара	Покупатель
0011	Иванов Иван
0023	Петров Петр
0023	Сидоров Сидор
0042	Иванов Иван
0042	Петров Петр

Код товара	Продавец	Покупатель
0011	ООО «ТриО»	Иванов Иван
0023	ООО «Бюро»	Петров Петр
		Сидоров Сидор
0042	ООО «Бирка»	Иванов Иван
		Петров Петр

Рисунок 4 – Реляционная (слева) и постреляционная (справа) модели данных [2]

1.3.4 Выбор модели организации данных

Поскольку задача требует выполнения разнообразных запросов, которые включают в себя сложные операции выборки данных по разным критериям, наиболее важными качествами модели данных являются гибкость, простота использования, целостность данных и независимость от приложения. Именно поэтому реляционная модель организации данных была выбрана.

1.4 Формализация данных

Выделяются следующие сущности, которые в базе данных должны хранить:

— пользователь;

— турнир;

— команда

— стадион;

— страна;

— матч;

— отзыв;

— заявка.

Информация о каждой сущности проводится в таблице 1.

Таблица 1 – Сущности и их описания

Сущность	**Описание**
Пользователь	Логин, пароль, роль, фамилия, имя, возраст, команда
Турнир	Название, рейтинг, страна, создатель
Стадион	Название, количество мест, страна
Страна	Название, континент
Команда	Название, страна
Заявка	Время создания, создатель, команда, турнир
Матч	Готевая команда, домашняя команда, голы гостевой, голы домашнней, турнир, стадион, неделя, время проведения
Отзыв	Пользователь, оценка, турнир

На рисунке 5 приведена ER-диаграмма сущностей в нотации Чена.

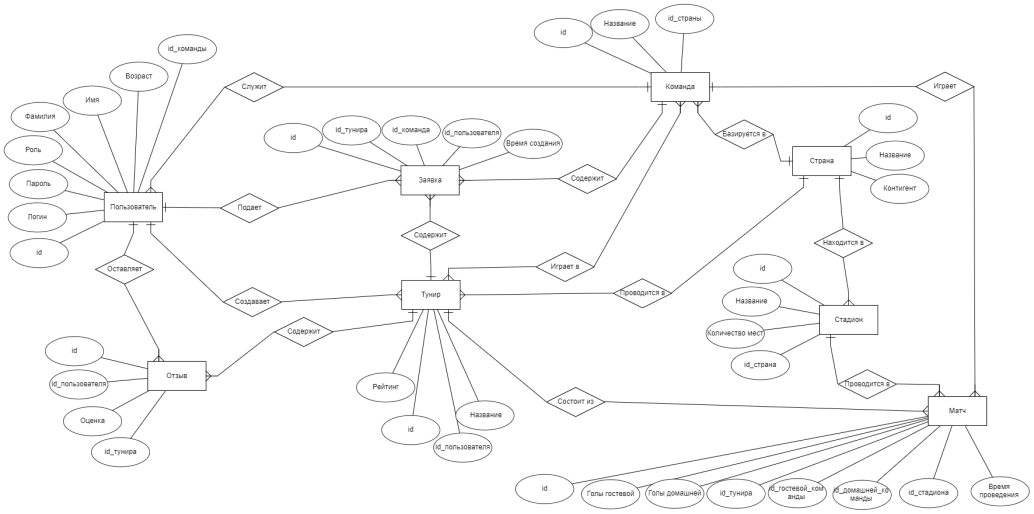

Рисунок 5 – ER-диаграмма сущностей

1.5 Системные пользователи

В приложении выделяются пять видов ролей:

1. Гость — неавторизованный пользователь, который обладающий возможностями зарегистрировать, входить в систему, просмотреть расписания турниров, посмотреть статистики турниров, оставить свои отзывы.
2. Футболист — авторизованный пользователь, который может подать заявку на поступление в клуб, просмотреть все информации о турнирах.
3. Тренер — авторизованный пользователь, который может создать свой клуб, принять/отменить заявку футболистов, подать заявку на поступление в турнир, просмотреть все информации о турнирах.
4. Судья — авторизованный пользователь, который может создать свои турниры, принять/отменить заявку тренеров, составить расписания турниров, вводить результаты матча.

5. Администатор — пользователь, обладающий возможностями всех вышеперечисленных пользователей. Кроме этого, он может изменять данных пользователей и турниров.

   На рисунках 6–7 приведена Use-Case диаграмма.

Рисунок 6 – Use-Case диаграмма

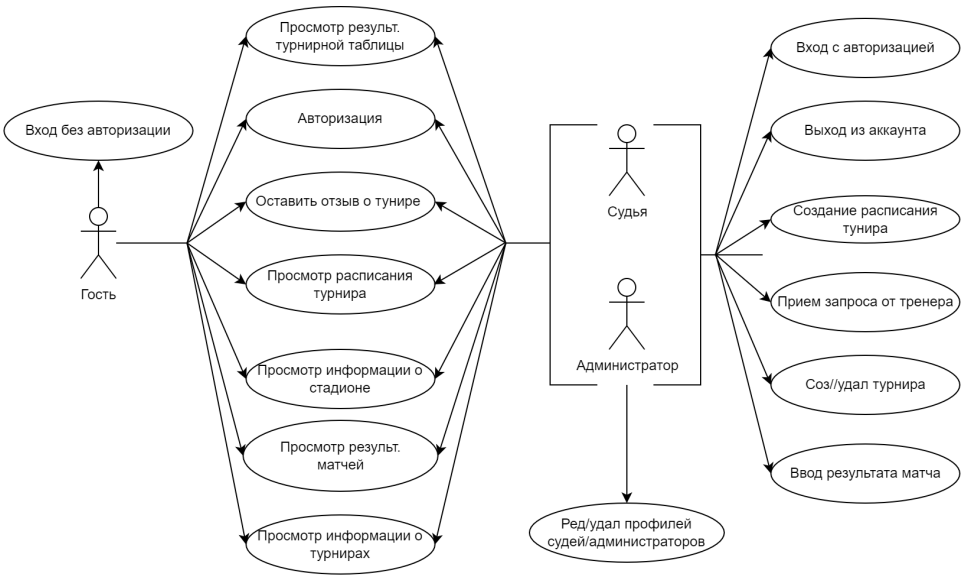

Рисунок 7 – Use-Case диаграмма (продолжение)

Вывод

В данном разделе была проанализирована предметная область. Была преведена постановка задачи, формализация данных и построена соотвествующая ER-диаграмма. Также описаны системные пользователи. Проведен анализ баз данных по модели данных и на основе этого анализа была выбрана подходящая модель для решения поставленной задачи.

2 Конструкторский раздел

2.1 Проектирование базы данных

Выделяются следующие сущности, которые должны хранить в базе данных.

1. Users — таблица пользователей;
2. Countries — таблица стран;
3. Stadiums — таблица стадионов;
4. Feedbacks — таблица отзывов;
5. Requests — таблица заявок пользователей;
6. Leagues — таблица турниров;
7. Clubs — таблица клубов;
8. Matches — таблица матчей;
9. League_club — таблица, реализующая связь многие к многим.

На основе диаграммы сущность-связей, приведенной на рисунке 5 определяются стуктуры столбцов, их типы и органичения.

Таблица 2 – Сведение о таблице users

Столбец	Тип данных	Ограничения	Значение
id	serial	PRIMARY KEY	Идентификатор
last_name	VARCHAR(32)		Фамилия
first_name	VARCHAR(32)		Имя
age	INT		Возраст
login	VARCHAR(64)	NOT NULL	Логин
password	VARCHAR(64)	NOT NULL	Пароль
role	VARCHAR(64)	NOT NULL	Права доступа
id_club	INT		ID клуба

Таблица 3 – Сведение о таблице leagues

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
name	VARCHAR(64)	NOT NULL	Название турнира
rating	DOUBLE	NOT NULL	Рейтинг турнира
id_user	INT	FOREIGN KEY	ID пользователя
id_country	INT	FOREIGN KEY	ID страны

Таблица 4 – Сведение о таблице clubs

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
name	VARCHAR(64)	NOT NULL	Название команды
id_country	INT	FOREIGN KEY	ID страны

Таблица 5 – Сведение о таблице stadiums

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
name	VARCHAR(64)	NOT NULL	Название стадиона
capacity	INT	NOT NULL	Количество мест
id_country	INT	FOREIGN KEY	ID страны

Таблица 6 – Сведение о таблице countries

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
name	VARCHAR(64)	NOT NULL	Название страны
continent	VARCHAR(64)	NOT NULL	Континент

Таблица 7 – Сведение о таблице feedbacks

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
mark	INT	NOT NULL	Оценка
id_user	INT	FOREIGN KEY	ID пользователя
id_league	INT	FOREIGN KEY	ID турнира

Таблица 8 – Сведение о таблице requests

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
created_time	DATE	NOT NULL	Время создания
id_user	INT	FOREIGN KEY	ID пользователя
id_club	INT	FOREIGN KEY	ID клуба
id_league	INT		ID турнира

Таблица 9 – Сведение о таблице matches

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
goal_home	DATE		Голы домашнего клуба
goal_guest	INT		Голы гостевого клуба
id_league	INT	FOREIGN KEY	ID турнира
id_stadium	INT	FOREIGN KEY	ID стадиона
id_home	INT	FOREIGN KEY	ID дом. клуба
id_guest	INT	FOREIGN KEY	ID гост. клуба
week	INT	NOT NULL	Неделя
time_start	VARCHAR(64)	NOT NULL	Время проведения матча

Таблица 10 – Сведение о таблице leagueclub

Столбец	Тип данных	Ограничения	Значение
id	SERIAL	PRIMARY KEY	Идентификатор
id_league	INT	FOREIGN KEY	ID турнира
id_club	INT	FOREIGN KEY	ID клуба

На рисунке 8 представлена ER-диаграмма базы данных.

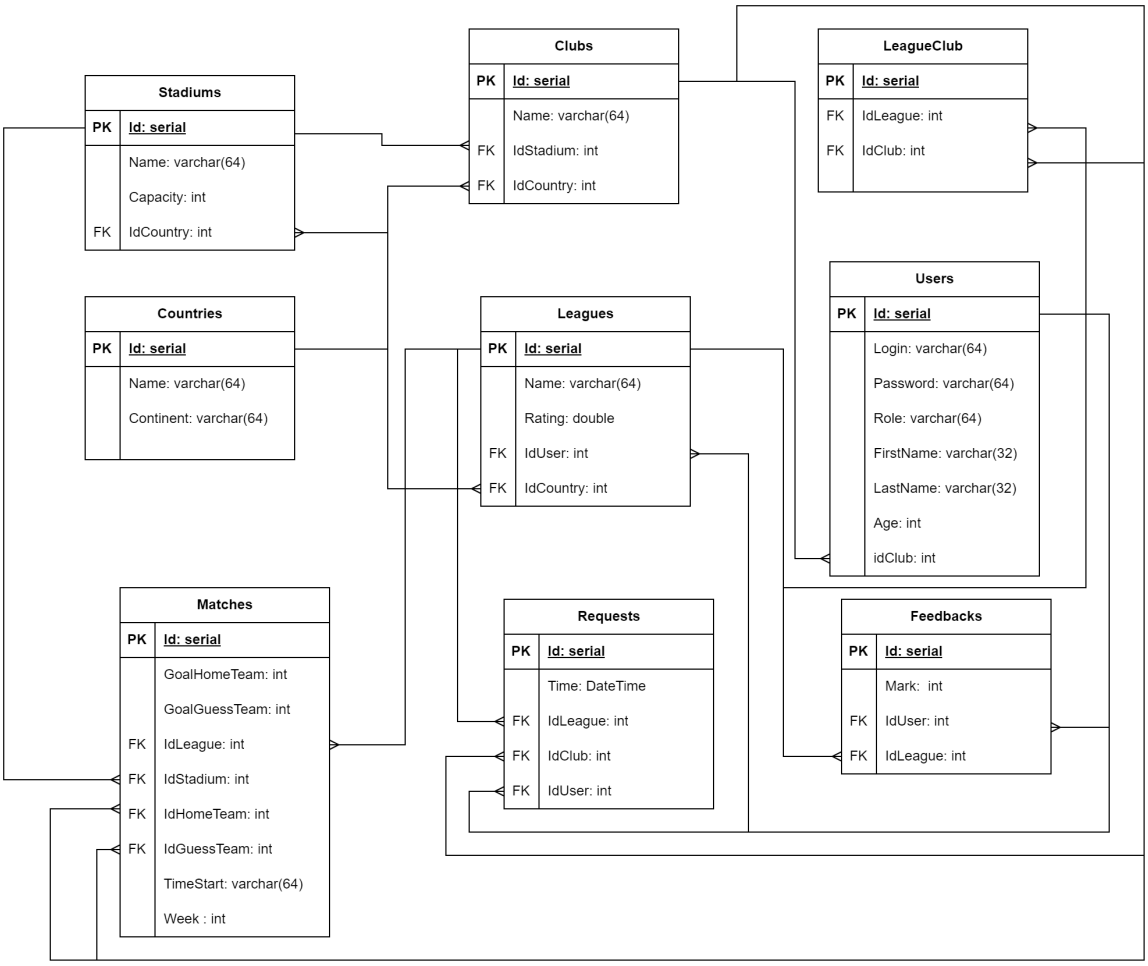

Рисунок 8 – ER-диаграмма базы данных

2.2 Проектирование приложения

Приложение будет состоять из 3 компонентов, каждый из которых имеет свою задачу.

1. Компонент доступа к базе данных (Data Access) — выполняет запросы к данным, обеспечивает операции CRUD (Create, Read, Update, Delete).
2. Компонент реазации интерфейса (UI) — пользовательский интерфейс, обеспечивает взаимодействие с программой.
3. Компонент бизнес логики (Business Logic) — обрабатывает бизнес правила, управляет данными и обрабатывает исключения.

2.3 Ролевая модель

В базе данных для каждой группы пользователей, выделенных выше была создана роль:

— гость имеет право просмотр всех таблиц, кроме таблиц users, requests, добавления в таблицы users, feedbacks;

— футболист имеет право просмотр всех таблиц, кроме таблицы users, добавления в таблицы feedbacks, requests;

— тренер имеет право просмотр всех таблиц, кроме таблицы users, добавления в таблицу feedbacks, requests, clubs, удаления в таблице reuquests, clubs, изменения в таблице users;

— судья имеет право просмотр всех таблиц, кроме таблицы users, добавления в таблиц leagues, feedbacks, удаления в таблице requests;

— администратор имеет право просмор всех таблиц, добавления, изменения, удаление во все таблиы.

2.4 Триггер

Триггер — это функция, которая автоматически вызывается после действия (insert, update, delete) над определенной таблицей [3].

Когда пользователь оставляет свои отзывы, необходимо пересчитать рейтинг турнира. При добавлении пользовательского отзыва нужно обновить значение поля «rating» в таблице турнира. Для процесса перерасчета рейтинга было бы удобно создать триггер, который обновляет поля рейтинга автоматически при добавлении отзыва.

На рисунке 9 приведена схема триггера для автоматически перерасчета рейтинга турнира.

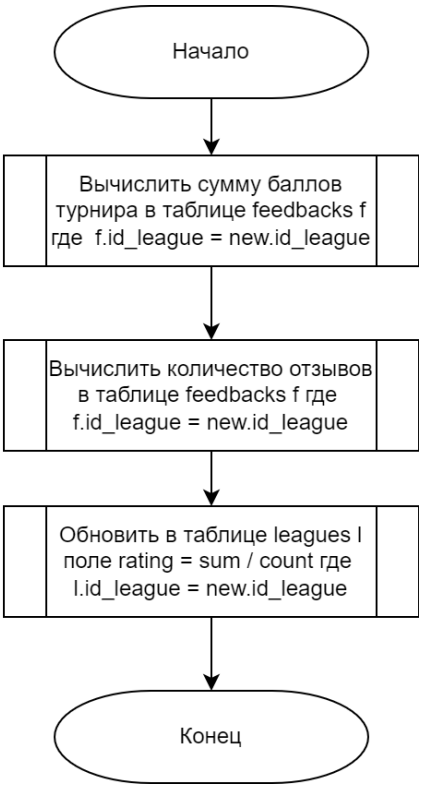

Рисунок 9 – Триггер для автоматически перерасчета рейтинга турнира

2.5 Алгоритм составления расписания турнира

Условием организации футбольного турнира является то, что количество участвующих команд должно быть четным.

Турнирное расписание матчей для набора команд , удовлетворяющее приведенному ниже набору ограничений:

— ни одна команда не должна играть 2 матча подряд в одной неделе;

— каждая команда должна была играть ровно 2 матча со всеми остальными командами. Второй матч состоится после встречи всех оставшихся команд;

— интервал между двумя матчами равен неделе.

На рисунке 10 приведена схема алгоритма составления расписания турнира.

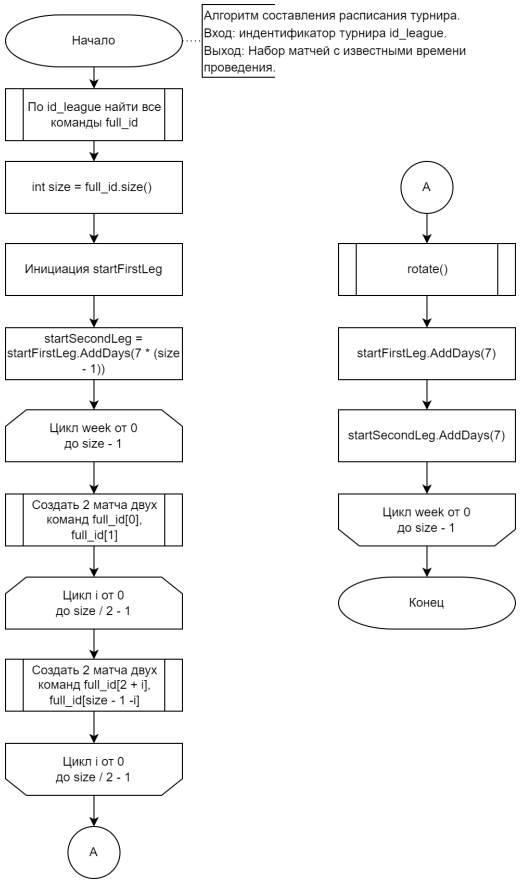

Рисунок 10 – Алгоритм составления расписания турнира

Вывод

Были приведено проектирование базы данных и проектирование приложения. Также был спроектирован триггер, осуществляющие автоматически пересчитывать рейтинга турнира при добавлении новых оценок.

3 Технологический раздел

3.1 Средства реализации

3.1.1 Выбор языка программирования

Для написания данной курсовой работы был выбран язык C# [4]. Этот выбор обусловлен следующими причинами:

— C# — объектно-ориентированный язык, что соответствует методологии, выбранной для разработки программы;

— C# предоставляется обширный набор библиотек и шаблонов, что позволяет использовать готовые конструкции и значительно экономит время разработки.

В качестве среды разработки был использован Visual Studio 2022 [5]. Данный выбор обусловлен следующими факторами:

— в Visual Studio есть возможность быстрого создания интерфейса с помощью WinForms;

— Visual Studio предоставляет шаблоны, которые облегчают процесс написания и отладки проекта.

3.1.2 Выбор СУБД

Рассматриваются некоторые полулярные реляционные СУБД [6]:

— Oracle;

— My SQL;

— Microsoft SQL Sever;

— PostgreSQL.

Выделяются следующие критерии для сравнения перечисленных СУБД:

— Лицензия;

— Возможность создания роли, выдачи права на уровне БД;

— Масштабируемость;

— Сообщество и подержка;

Таблица 11 – Сравнение реляционных СУБД [6]

	Oracle	MySQL	MSQL Server	PostgreSQL
Лицензия	Коммер.	Двойная	Коммер.	Открытая
Создания ролей	Да	Да	Да	Да
Масштабируемость	Высокая	Хорошая	Хорошая	Высокая
Сообщ. и поддержка	Крупное	Активное	Обширное	Крупное

Исходя из сравнения по этим критериям, PostgreSQL представляется привлекательным выбором ввиду своей открытой лицензии, богатых возможностей управления ролями и правами доступа, высокой масштабируемости и активного сообщества с открытым исходным кодом, что делает его привлекательным для разнообразных проектов и бизнесов.

3.2 Создание базы данных

На основе ER-диагрммы базы данных, представленной на рисунке 8 были созданы все сущности. Программный код приведен в листинге 1 в приложении А.

Была создана ролевая модель на листингах 4–8 в приложении А.

Реализация функций на листинге 2 в приложении А. Реализация триггера на листинге 3 в приложении А.

3.3 Описание интерфейсов

Проект ориентирован на создание удобного и интуитивно понятного пользовательского интерфейса для работы с базой данных через Desktop-приложение. Он предоставляет пользователям простой и интуитивно понятный способ нахождения необходимой информации и выполнения требуемых операций. На рисунках 14–19 в приложении А, отображают интерфейсы, которые были спроектированы для обеспечения этой цели. В этих изображениях могут быть показаны компоненты, предоставляющие пользователям возможность взаимодействия с данными из базы данных.

3.4 Тестирование

3.4.1 Функция

Проводится тестирования функции, которая составляет турнирную таблицу на следующих случаях:

1. Турнир состоит из 6 команд, все матчи сыграны.
2. Турнир состоит из 4 команды, часть матчей сыграна.
3. Турнир состоит из 4 команды, матчи не были сыграны.

Реализация тестирования приведена в листинге 9 в приложении А.

На рисунке 11 приведены результаты тестирования.

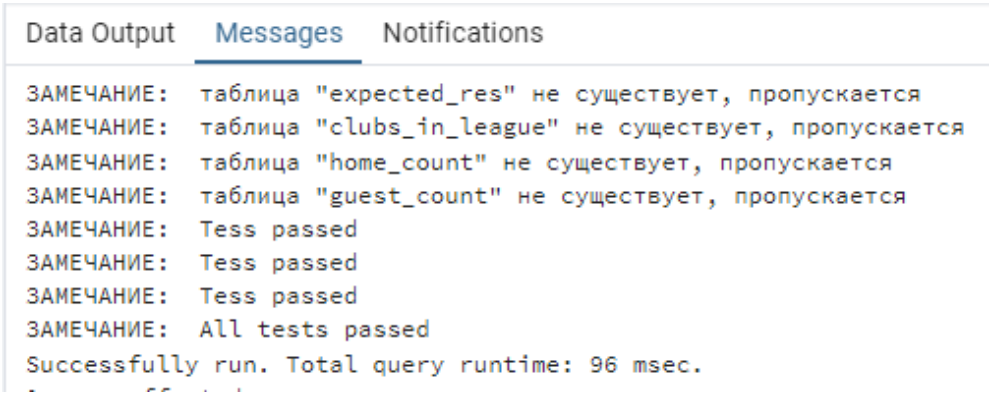

Рисунок 11 – Результаты тестов для функции get_table()

3.4.2 Триггер

Проводится тестирования триггер, который автоматически обновляет значение поля «rating» в таблице leagues при добавлении новых оценок на следующих случаях:

1. Добавление средней оценки.
2. Добавление нижней граничной оценки.
3. Добавление верхней граничной оценки.

Реализация тестирования приведена в листинге 10 в приложении А.

На рисунке 12 приведены результаты тестирования.

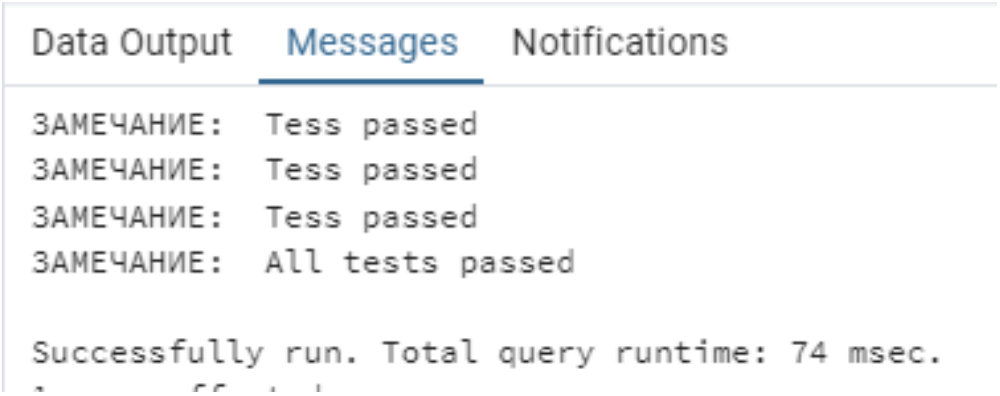

Рисунок 12 – Результаты тестов для триггера auto_cal_rating

Вывод

Был рассмотрен выбор средств реализации, описан интерфейс программного обеспечения и проведено тестирование функции на строне базы данных.

4 Исследовательский раздел

4.1 Постановка исследования

Целью исследования является проведение анализа зависимости времени работы алгоритма составления турнирного расписания от количества команд, вступающих в турнир на строне базы данных.

4.2 Технические характеристики

Исследование проводилось на устройстве со следующими техническими характеристиками:

— операционная система Window 10 Home Single Language;

— память 8 Гб;

— процессор 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 ГГц, 4 ядра.

4.3 Результаты исследования

Для исследования зависимости времени составления расписания, использовалось четное количество команд. Количество команд менялось от 10 до 100 с шагом 10. Результаты исследования представлены в таблице 12.

Таблица 12 – Результаты замеров времени

Количество команд	Время составления расписания
2	0.274000
10	1.513000
20	4.714000
30	11.359000
40	21.928000
50	45.710000
60	67.000000
70	125.247000
80	174.082000
90	200.249000
100	218.990000

На рисунке 13 приведен график зависимости времени составления от числа команд.

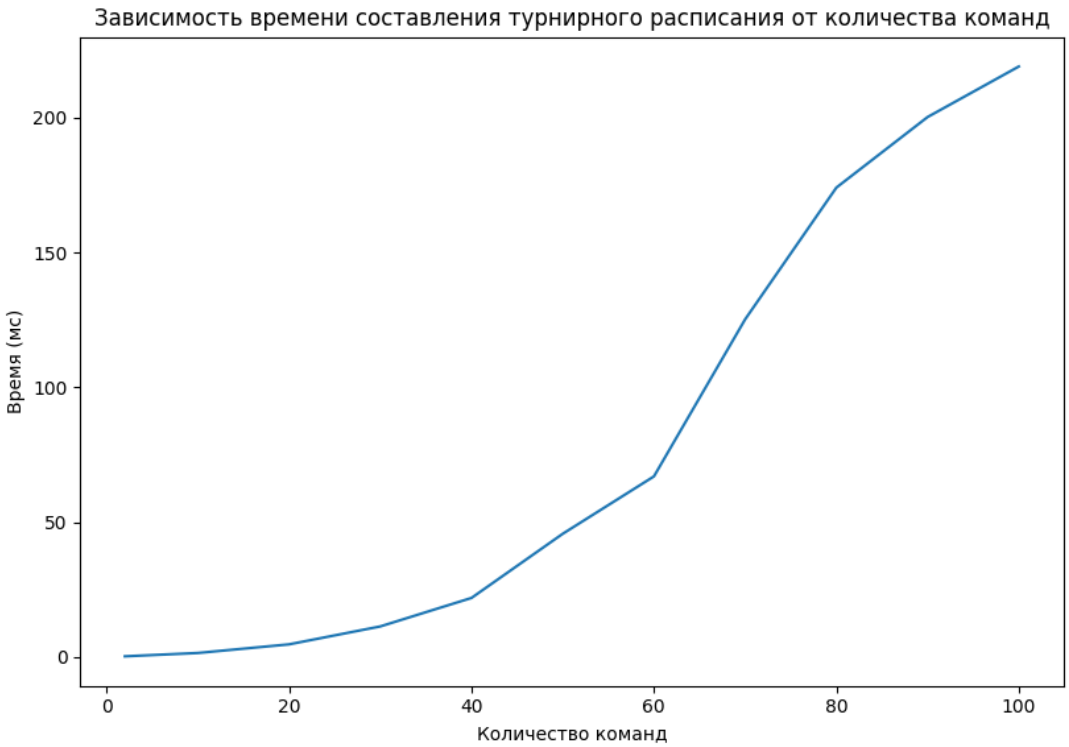

Рисунок 13 – График зависимости времени составления от числа команд

Из проведенного исследования можно сделать вывод, что время составления расписания линейно зависит от количества команд в турнире.

Вывод

В данном разделе приведены технические характеристики и исследования зависимости времени работы функции на строне базы данных.

ЗАКЛЮЧЕНИЕ

Цель курсовой работы была достигнута, то есть была разработана база данных для организации футбольных турниров.

Для достижение цели были решены все задачи:

- проведен анализ предметной области в сфере организации футбольных турниров;
- выделены роли пользователей проектируемого приложения;
- спроектирована база данных, описать ее сущности и связи, так же спроектировать триггер;
- спроектирована ролевая модель на уровне базы данных
- создано программное обеспечение, обеспечивающее доступ к спроектированной базе данных;
- проведены замеры временных характеристик српоектированной базы данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Турнир по футболу: значение и особенности [Электронный ресурс]. — Режим доступа: https://obzorposudy.ru/polezno/turnir-po-futbolu-znacenie-i-osobennosti (дата обращения: 14.04.2024).

2 С.Н. Ткаченко. Основы проектирования баз данных. КноРус, 2024. — С. 176.

3 Триггеры в PostgreSQL [Электронный ресурс]. — Режим доступа: https://sql-ex.ru/blogs/?/Triggery_v_PostgreSQL_chast_1.html (дата обращения: 15.04.2024).

4 Документация по языку C-Sharp [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата обращения: 27.03.2024).

5 Visual Studio Community [Электронный ресурс]. — Режим доступа: https://visualstudio.microsoft.com/vs/community/ (дата обращения: 27.03.2024).

6 Сравнение современных СУБД [Электронный ресурс]. — Режим доступа: https://drach.pro/blog/hi-tech/item/145-db-comparison (дата обращения: 14.04.2024).

ПРИЛОЖЕНИЕ А

Описание интерфейсов

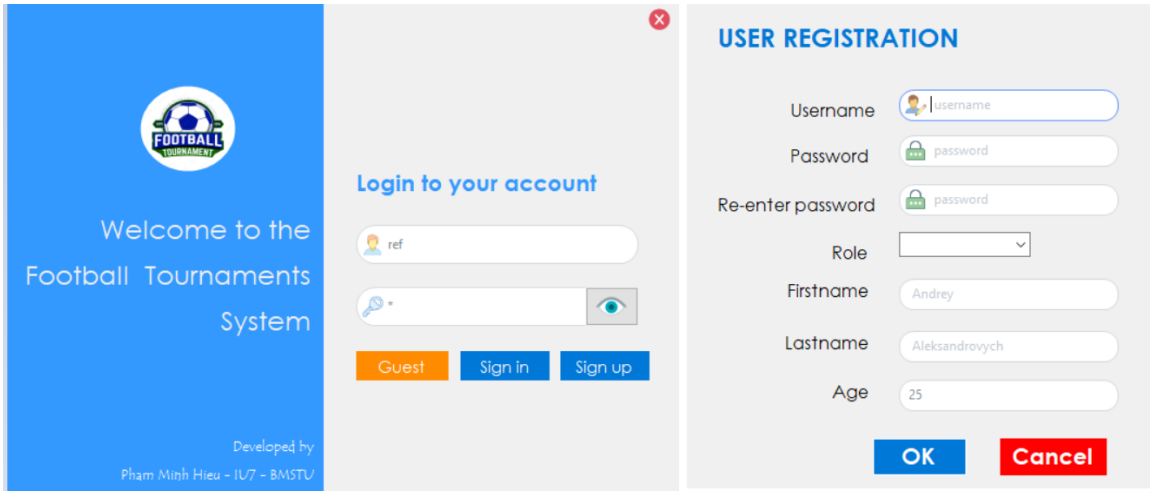

Рисунок 14 – Демонстрация работы программы при входе в систему и зарегистрации

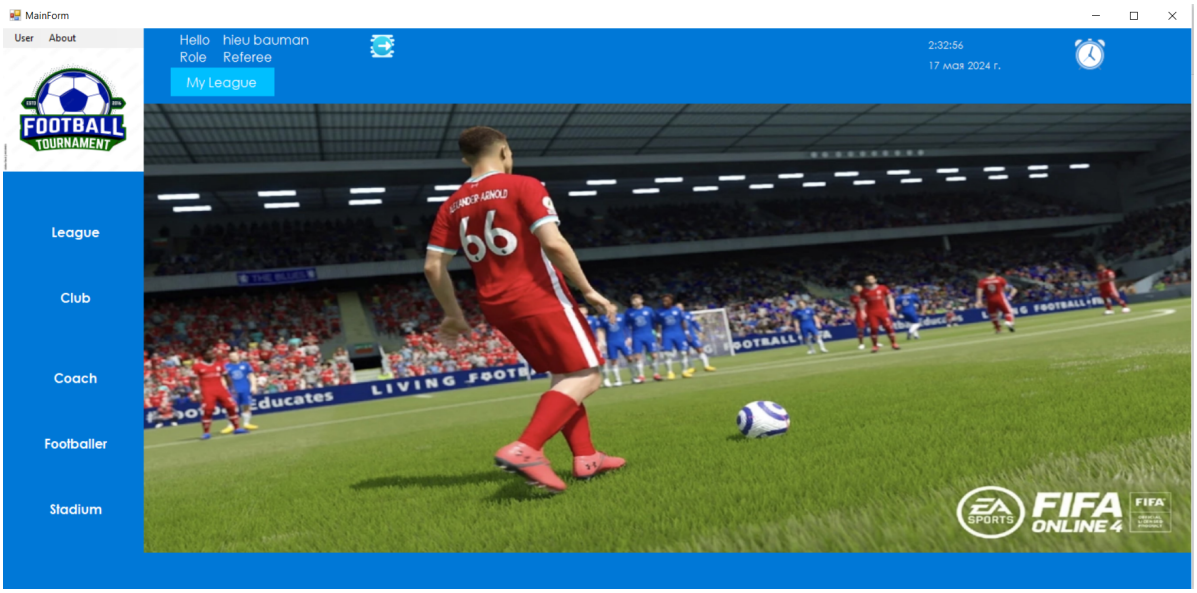

Рисунок 15 – Демонстрация главного экрана

На рисунках 16–17 приведены демонстрации работы программы.

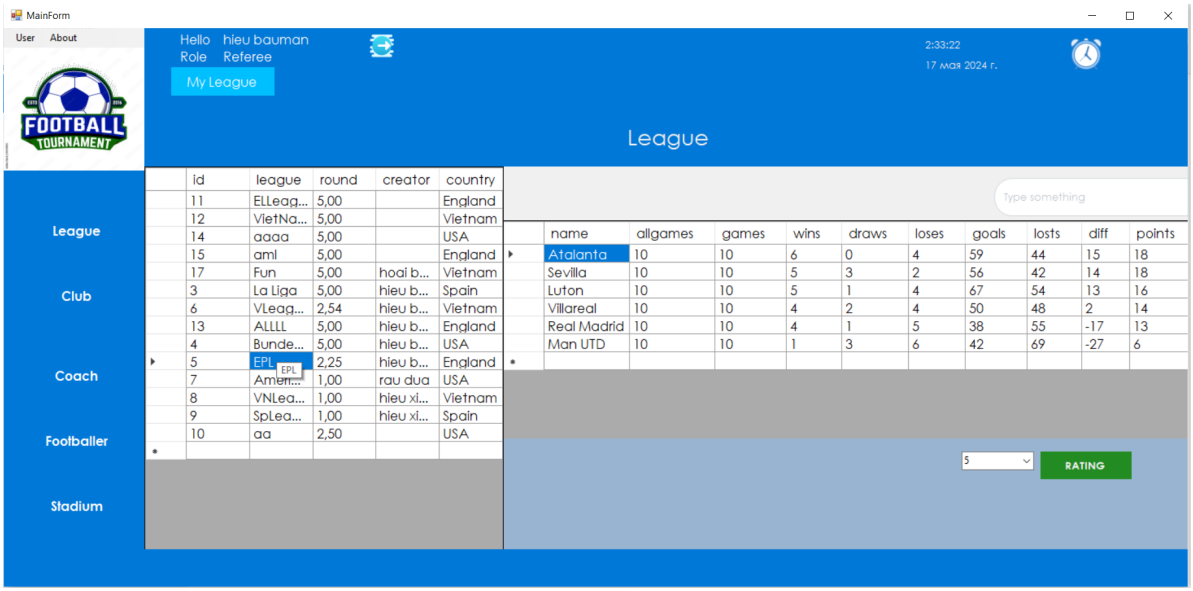

Рисунок 16 – Демонстрация работы программы при просмотре статистики турнира

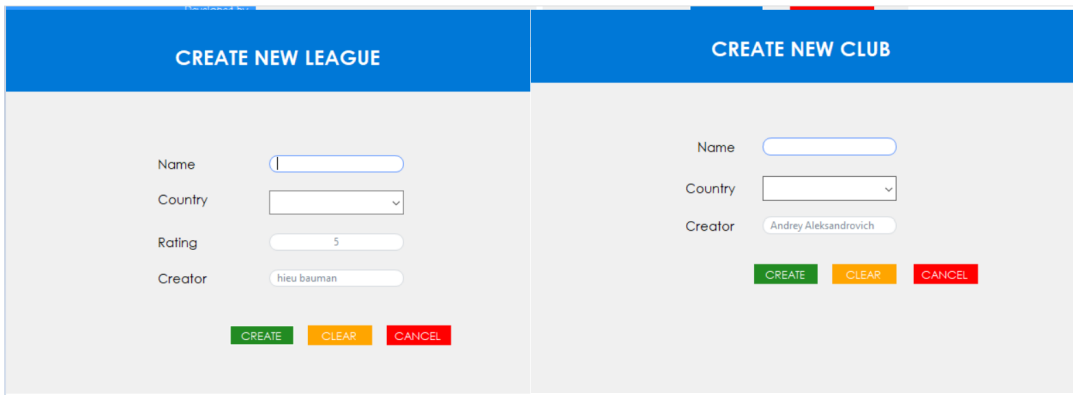

Рисунок 17 – Демонстрация работы программы при создании нового турнира и нового клуба

На рисунках 18–19 приведены демонстрации работы программы.

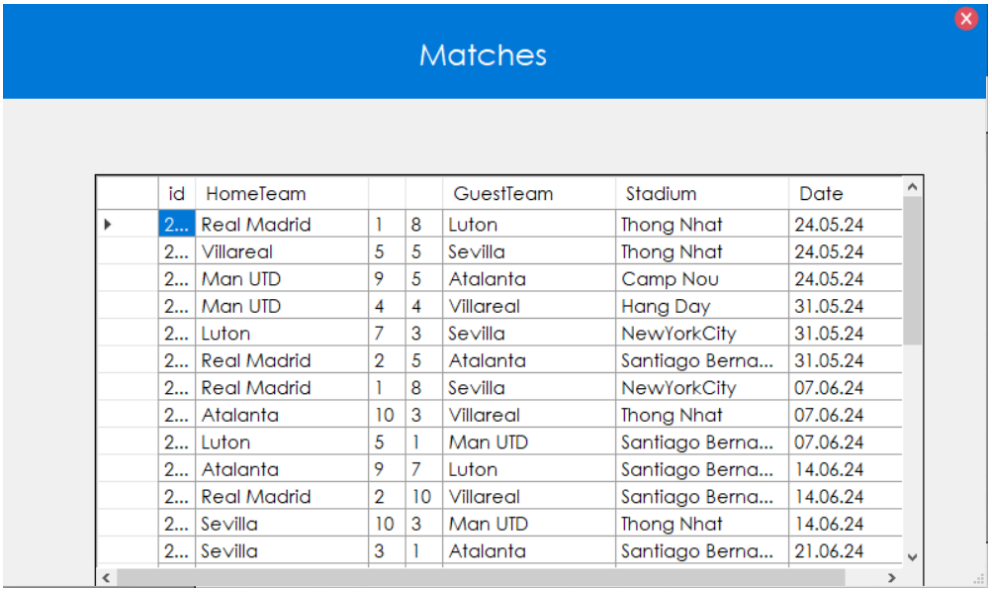

Рисунок 18 – Демонстрация работы программы при просмотре результата матчей

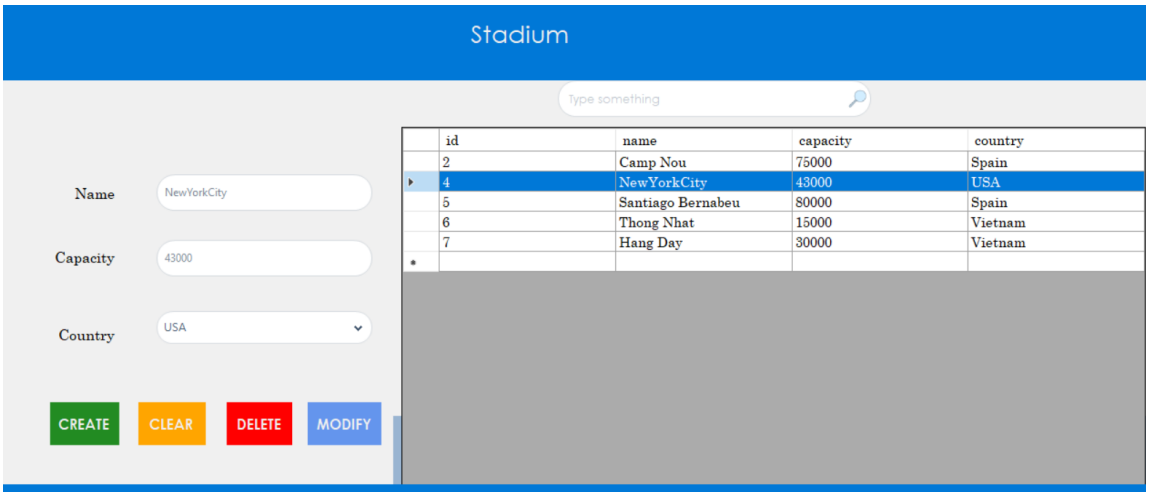

Рисунок 19 – Демонстрация работы программы при просмотре список стадионов

Создание базы данных

Листинг 1 – Создание всех таблиц

```
1  create table if not exists Users
2  (
3      id serial primary key,
4      login varchar(30) not null,
5      password varchar(30) not null,
6      role varchar(20) not null,
7      firstname varchar(30) default 'hieu',
8      lastname varchar(30) default 'bich',
9      age int default 25,
10     id_club int default -1,
11 );
12
13 create table if not exists Countries
14 (
15     id serial not null primary key,
16     name varchar(40) not null,
17     continent varchar(40) not null
18 );
19
20 create table if not exists Clubs
21 (
22     id serial not null primary key,
23     name varchar(50) not null,
24     id_country int not null,
25
26     foreign key (id_country) references Countries(id)
27 );
28
29 create table if not exists Stadiums
30 (
31     id serial not null primary key,
32     name varchar(30) not null,
33     capacity int not null,
34     id_country int not null,
35
36     foreign key (id_country) references Countries(id)
```

```
37 );
38
39 create table if not exists Leagues
40 (
41     id serial not null primary key,
42     name varchar(30) not null,
43     rating float,
44     id_user int,
45
46     foreign key (id_user) references Users(id)
47 );
48
49 create table if not exists Matches
50 (
51     id serial not null primary key,
52     goal_home_club int not null,
53     goal_guess_club int not null,
54     id_league int not null,
55     id_stadium int not null,
56     id_home_club int not null,
57     id_guess_club int not null,
58
59     foreign key (id_league) references Leagues(id),
60     foreign key (id_stadium) references Stadiums(id),
61     foreign key (id_home_club) references Clubs(id),
62     foreign key (id_guess_club) references Clubs(id)
63 );
64
65 create table if not exists Requests
66 (
67     id serial not null primary key,
68     created_time timestamp default now(),
69     id_league int default -1,
70     id_club int default -1,
71     id_user int not null,
72
73     foreign key (id_league) references Leagues(id),
74     foreign key (id_club) references Clubs(id),
75     foreign key (id_user) references Users(id)
76 );
```

```
77
78 create table if not exists Feedbacks
79 (
80     id serial not null primary key,
81     comment varchar(200),
82     grade int not null,
83     id_user int not null,
84     id_league int not null,
85
86     foreign key (id_user) references Users(id),
87     foreign key (id_league) references Leagues(id)
88 );
89
90 create table if not exists LeagueClub
91 (
92     id serial not null primary key,
93     id_league int not null,
94     id_club int not null,
95
96     foreign key (id_league) references Leagues(id),
97     foreign key (id_club) references Clubs(id)
98 );
```

Листинг 2 – Функция составляет турнирное расписание

```
1  create or replace function create_schedule(id_l int) returns void as $$
2  declare
3      size int;
4      ss int;
5      start_first_leg date := '2024-05-24';
6      start_second_leg date;
7      full_id int[];
8      id_stadiums int[];
9      week int;
10     random_index int;
11 begin
12     select array_agg(id_club) into full_id
13     from leagueclub where id_league = id_l;
14     size := array_length(full_id, 1);
15     select array_agg(id) into id_stadiums from stadiums;
16     ss := array_length(id_stadiums, 1);
```

```
17
18      start_second_leg := start_first_leg + INTERVAL '7' DAY * (size - 1);
19
20      for week in 1..(size - 1) loop
21          call createMatch(id_l, id_stadiums[getRandomNum(ss)], full_id[1],
                full_id[2], week, start_first_leg);
22          call createMatch(id_l, id_stadiums[getRandomNum(ss)], full_id[2],
                full_id[1], week + size - 1, start_second_leg);
23
24          for i in 0..(size / 2 - 2) loop
25              call createMatch(id_l, id_stadiums[getRandomNum(ss)],
                    full_id[3 + i],
26              full_id[size - i], week, start_first_leg);
27              call createMatch(id_l,
28              id_stadiums[getRandomNum(ss)],
29               full_id[size - i], full_id[3 + i],
30               week + size - 1, start_second_leg);
31          end loop;
32
33          full_id = rotate_teams(full_id);
34          start_first_leg := start_first_leg +
35           INTERVAL '7' day;
36          start_second_leg := start_second_leg
37           + INTERVAL '7' day;
38      end loop;
39  end;
40  $$
41  language plpgsql;
42
43  create or replace function rotate_teams(full_id int[])
44  returns int[]
45  as
46  $$
47  declare
48      last_id int;
49      size int;
50  begin
51      size := array_length(full_id, 1);
52      last_id := full_id[size];
53      for i in reverse size..3 loop
```

```
54      full_id[i] := full_id[i - 1];
55      end loop;
56      full_id[2] := last_id;
57      return full_id;
58  end;
59  $$
60  language plpgsql;
```

Листинг 3 – Реализация триггера

```
1   create or replace function calculate_rating()
2   returns trigger
3   as
4   $$
5   begin
6       update leagues set rating = (select sum(f.grade) / count(*)::double
            precision as rating
7       from feedbacks f
8       where id_league = new.id_league)
9       where id = new.id_league;
10      return new;
11  end;
12  $$
13  language plpgsql;
14
15  create trigger auto_cal_rating
16  after insert on feedbacks
17  for each row execute function calculate_rating();
```

Листинг 4 – Создание роли администратора и выдыча права

```
1   create role league_admin with
2   connection limit -1
3   login
4   password '#admin123';
5
6   grant all privileges
7   on all tables in schema public
8   to league_admin;
```

Листинг 5 – Создание роли судьи и выдыча права

```
create role league_referee with
connection limit - 1
login
password '#referee123';

grant select on all tables in schema public
to league_referee;

grant insert on
public."leagues",
public."leagueclub",
public."feedbacks",
public."matches"
to league_referee;

grant update on
public."leagues",
public."users",
public."matches"
to league_referee;

grant delete on
public."requests"
to league_referee;
```

Листинг 6 – Создание роли тренера и выдыча права

```
create role league_coach with
connection limit - 1
login
password '#coach123';

grant select on all tables in schema public
to league_coach;

grant insert on
public."clubs",
public."feedbacks",
public."requests"
to league_coach;
```

```
14
15 grant update on
16 public."clubs",
17 public."users"
18 to league_coach;
19
20 grant delete on
21 public."requests"
22 to league_coach;
```

Листинг 7 – Создание роли футболиста и выдыча права

```
1  create role league_footballer with
2  connection limit -1
3  login
4  password '#footballer123';
5
6  grant select on all tables in schema public
7  to league_footballer;
8
9  grant insert on
10 public."feedbacks",
11 public."requests"
12 to league_footballer;
13
14 grant update on
15 public."users"
16 to league_footballer;
17
18 grant delete on
19 public."requests",
20 public."feedbacks"
21 to league_footballer;
```

Листинг 8 – Создание роли гости и выдыча права

```
1  create role league_guest with
2  connection limit -1
3  login
4  password '#guest123';
5
6  grant select on all tables in schema public
```

```
to league_referee;

grant insert on
public."users",
public."feedbacks"
to league_guest
```

Листинг 9 – Реализация тестирования для функции get_table_league

```
create or replace function test_get_table()
returns void as
$$
begin
    drop table if exists expected_res;
    create temp table if not exists expected_res
    (
    name varchar(64),
    allgames int,
    games int,
    wins int,
    draws int,
    loses int,
    goals int,
    losts int,
    diff int,
    points int
    );

    insert into expected_res values
    ('Atalanta',    10, 10, 6,  0,  4,  59, 44, 15, 18),
    ('Sevilla', 10, 10, 5,  3,  2,  56, 42, 14, 18),
    ('Luton',   10, 10, 5,  1,  4,  67, 54, 13, 16),
    ('Villareal',   10, 10, 4,  2,  4,  50, 48, 2,  14),
    ('Real Madrid', 10, 10, 4,  1,  5,  38, 55, -17,    13),
    ('Man UTD', 10, 10, 1,  3,  6,  42, 69, -27,    6);

    call auto_compare(5);

    truncate expected_res;
    insert into expected_res values
    ('Atalanta',    6,  4,  2,  1,  1,  22, 21, 1,  7),
```

```
33      ('Villareal',   6,  3,  2,  0,  1,  25, 20, 5,  6),
34      ('Sevilla', 6,  3,  1,  1,  1,  18, 16, 2,  4),
35      ('Aston Villa', 6,  2,  0,  0,  2,  11, 19, -8, 0);
36
37      call auto_compare(4);
38
39      truncate expected_res;
40      insert into expected_res values
41      ('Arsenal', 6,  0,  0,  0,  0,  0,  0,  0,  0),
42      ('Aston Villa', 6,  0,  0,  0,  0,  0,  0,  0,  0),
43      ('Luton',   6,  0,  0,  0,  0,  0,  0,  0,  0),
44      ('Real Madrid', 6,  0,  0,  0,  0,  0,  0,  0,  0);
45
46      call auto_compare(8);
47
48      raise notice 'All tests passed';
49  end;
50  $$
51  language plpgsql;
```

Листинг 10 – Реализация тестирования для триггера auto_cal_rating

```
1  CREATE OR REPLACE FUNCTION test_trigger()
2  RETURNS void AS $$
3  DECLARE
4  rec RECORD;
5  BEGIN
6  select rating from leagues where id in (5, 8, 9);
7  INSERT INTO feedbacks(grade, id_user, id_league)
8  VALUES (3, 100, 5);
9  INSERT INTO feedbacks(grade, id_user, id_league)
10 VALUES (5, 100, 8);
11 INSERT INTO feedbacks(grade, id_user, id_league)
12 VALUES (1, 100, 9);
13
14
15 FOR rec IN (SELECT id, rating FROM leagues WHERE (id, rating) NOT IN ((5,
       2.24), (8, 3), (9, 1))) LOOP
16 RAISE EXCEPTION 'Test failed: Unexpected result %', rec;
17 END LOOP;
18
```

```
19 RAISE NOTICE 'All tests passed';
20 END;
21 $$ LANGUAGE plpgsql;
```

ПРИЛОЖЕНИЕ Б

Презентация к курсовой работе

Презентация содержит 14 слайдов.