

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Автоматики и вычислительной техники

Кафедра Автоматизированных систем управления

Оценка комиссии: _____ Рейтинг: _____

Подписи членов комиссии:

| | |
|-----------|--------------------------|
| _____ | Папилина Т. М. |
| (подпись) | (фамилия, имя, отчество) |
| _____ | Волков Д. А. |
| (подпись) | (фамилия, имя, отчество) |
| _____ | |
| (дата) | |

КУРСОВАЯ РАБОТА

по дисциплине Базы данных

на тему Проектирование реляционной базы данных

«К ЗАЩИТЕ»

ВЫПОЛНИЛ:

Студент группы АС-22-05
(номер группы)

(должность, ученая степень; фамилия, и. о.)

Ильичев Роман Сергеевич
(фамилия, имя, отчество)

(подпись)

(подпись)

(дата)

(дата)

Москва, 20 24

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Автоматики и вычислительной техники

Кафедра Автоматизированных систем управления

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

по дисциплине Базы данных

на тему Проектирование реляционной базы данных

ДАНО студенту Ильичеву Роману Сергеевичу группы АС-22-05
(фамилия, имя, отчество в дательном падеже) (номер группы)

Содержание работы:

1. Концептуальное проектирование
2. Логическое проектирование
3. Физическое проектирование
4. Создание представлений для работы с БД

Исходные данные для выполнения работы:

1. —

Рекомендуемая литература:

1. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 368 с.

Графическая часть:

1. ER-диаграмма базы данных
2. —

Требования к представлению результатов:

| | |
|---|--|
| ✓ | Электронная версия |
| | Бумажный вариант и электронный образ документа |

Руководитель: _____
(уч. степень) (должность) (подпись) (фамилия, имя, отчество)

Задание принял к исполнению: студент _____
(подпись) (фамилия, имя, отчество) Ильичев Р.С.

Оглавление

| | |
|---|-----------|
| Введение..... | 4 |
| Основная часть..... | 5 |
| 1. Концептуальное проектирование | 5 |
| 2. Логическое проектирование | 9 |
| 3. Физическое проектирование | 15 |
| 4. Создание представлений | 17 |
| Заключение | 21 |
| Список литературы..... | 22 |
| Приложение..... | 23 |

Введение

База данных (БД) – это совокупность взаимосвязанных, хранящихся вместе данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для множества приложений. Различают реляционные и нереляционные БД. Реляционные БД – это БД, которые используются для хранения и предоставления доступа к взаимосвязанным элементам информации. Каждая строка, содержащаяся в таблице такой БД, представляет собой запись с уникальным идентификатором, который называют ключом. Нереляционные БД – это БД, в которых в отличие от большинства традиционных систем БД не используется табличная схема строк и столбцов. В этих БД применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных.

Целью данной работы является проектирование реляционной БД посредством прохождения 3 последовательных этапов: концептуальное проектирование, логическое проектирование, физическое проектирование. Также необходимо будет создать несколько представлений для работы с БД. В качестве темы для проектирования я выбрал «Турниры по футболу в Ярославской области». Данная БД должна хранить информацию о футбольных командах, игроках, турнирах, матчах и результативных действиях игроков.

При выполнении работы я использовал такую СУБД как PostgreSQL [2]. PostgreSQL обеспечивает очень хорошую поддержку стандарта языка SQL и также предоставляет интересные и практически полезные дополнительные возможности. Кроме того, PostgreSQL является свободно распространяемым продуктом с открытым исходным кодом, который доступен на большом числе платформ. В качестве инструмента для визуального проектирования БД я использовал программное обеспечение pgAdmin 4.

Основная часть

1. Концептуальное проектирование

Концептуальное проектирование – это проектирование БД, целью которого является создание концептуальной схемы данных на основе представлений о предметной области, каждого отдельного типа пользователя.

Концептуальная схема – это описание основных сущностей и связей между ними без учета моделей данных и синтаксиса целевой СУБД [3].

Концептуальное проектирование включает:

- выделение сущностей и их свойств (имя, идентификаторы, связи с другими сущностями);
- определение атрибутов (простые и составные, однозначные и многозначные, ключевые и неключевые, обязательные и вычисляемые, внешние и первичные ключи);
- определение связей.

На данном этапе была выбрана тема для проектируемой БД – турниры по футболу в Ярославской области. В ходе концептуального проектирования были выделены сущности и атрибут, необходимые для организации БД (*Таблица 1*). Также сразу определим первичные ключи (ПК) и внешние ключи (ВК).

Таблица 1 – Сущности и их атрибуты

| Сущность (таблица) | Описание сущности |
|--------------------|--|
| teams | Содержит все команды. Включает атрибуты: <ul style="list-style-type: none">• id (ПК) – идентификатор команды;• name – название команды;• coach – тренер команды;• city – город, в котором находится команда; |

| | |
|--------------|--|
| | <ul style="list-style-type: none"> • stadium – название стадиона, на котором играет команда. |
| players | <p>Содержит всех игроков.</p> <p>Включает атрибуты:</p> <ul style="list-style-type: none"> • id (ПК) – идентификатор игрока; • team_id (ВК) – идентификатор команды, в которой тренируется игрок; • firstname – имя игрока; • lastname – фамилия игрока; • birth_date – дата рождения игрока; • player_position – позиция игрока на футбольном поле. |
| tournaments | <p>Содержит все турниры.</p> <p>Включает атрибуты:</p> <ul style="list-style-type: none"> • id (ПК) – идентификатор турнира; • name – название турнира; • start_date – дата начала турнира; • end_date – дата окончания турнира. |
| applications | <p>Содержит заявки команд на турниры.</p> <p>Включает атрибуты:</p> <ul style="list-style-type: none"> • id (ПК) – идентификатор заявки; • tournament_id (ВК) – id турнира, на который заявляется команда; • team_id (ВК) – id команды, которая заявляется на турнир. |
| matches | <p>Содержит все матчи.</p> <p>Включает атрибуты:</p> <ul style="list-style-type: none"> • id (ПК) – идентификатор матча; |

| | |
|-------------------|---|
| | <ul style="list-style-type: none"> • tournament_id (БК) – id турнира; • team1_id (БК) – id первой команды; • team2_id (БК) – id второй команды; • date – дата проведения матча; • stadium – стадион, место проведения матча; • scored_goals_team1 – количество голов, забитых первой командой; • scored_goals_team2 – количество голов, забитых второй командой. |
| effective actions | <p>Содержит все эффективные действия игроков: голы и ассисты.</p> <p>Включает атрибуты:</p> <ul style="list-style-type: none"> • id (ПК) – идентификатор эффективного действия; • match_id (БК) – id матча; • player_id (БК) – id игрока; • type – тип совершаемого действия (гол, голевой пас); • minute – минута матча, на которой было совершено эффективное действие. |

Определения связей 1:M:

- teams.id → players.team_id (в одной команде тренируется много игроков);
- teams.id → matches.team1_id (одна команда принимает участие в многих домашних матчах);
- teams.id → matches.team2_id (одна команда принимает участие в многих гостевых матчах);

- teams.id → applications.team_id (у одной команды есть несколько заявок на турниры);
- players.id → effective_actions.player_id (у одного игрока на счету много результативных действий);
- tournaments.id → applications.tournament_id (на один турнир заявлено много команд);
- tournaments.id → matches.tournament_id (для одного турнира характерно много матчей);
- matches.id → effective_actions.match_id (в одном матче совершается много результативных действий и голов).

Определение связей М:М:

- applications ↔ players (в одной заявке есть много игроков, один игрок может быть включен сразу в несколько заявок).

2. Логическое проектирование

Логическое проектирование – этап проектирования БД, целью которого является развитие концептуальной схемы с учётом выбора модели данных, но без учёта синтаксиса целевой СУБД.

Логическое проектирование включает такие этапы как:

- удаление и проверка элементов, не отвечающих принятой модели данных (удаление связей М:М, удаление связей с атрибутами, удаление сложных связей, удаление многозначных атрибутов, удаление рекурсивных связей и др.);
- нормализация отношений;
- поддержка целостности данных (домены, триггеры).

Переносим концептуальную схему в специальную программу, предоставляющую инструмент для визуального проектирования БД – pgAdmin 4, где и будем заниматься разработкой ERD [1]. Переходим к первому этапу логического проектирования.

В разработанной БД почти все элементы отвечают реляционной модели данных за исключением связи М:М. Чтобы решить данную проблему, удаляем связь М:М между сущностями players и applications с помощью добавления промежуточной таблицы players_applications (Таблица 2).

Таблица 2 – Создание промежуточной таблицы

| Сущность (таблица) | Описание сущности |
|----------------------|--|
| players_applications | Промежуточная таблица, связывает сущности applications и players, показывает, какие игроки заявлены от команд на турниры. Включает атрибуты: <ul style="list-style-type: none">• player_id (БК) – id игрока;• application_id (БК) – id заявки. |

Таким образом, у нас образовались новые связи 1:М:

- `players.id` → `players_applications.player_id` (один игрок может быть включен в несколько заявок);
- `application.id` → `players_applications.application_id` (в одной заявке есть много игроков).

Следующим этапом логического проектирования является нормализация отношений – процесс оптимизации логической схемы для построения надежной и производительной БД путем удаления функциональных зависимостей, приводящих к потенциальной противоречивости в данных.

Нормальная форма (НФ) – ограничение на схему базы данных, вводимое для устранения потенциального нарушения целостности при выполнении реляционных операций.

Существуют следующие виды НФ:

- ННФ – ненормальная форма (без требований);
- 1НФ – первая нормальная форма (обязательное требование реляционных БД);
- 2НФ, 3НФ, НФБК – устранение часто возникающих аномалий;
- 4НФ – решение проблем с многозначной зависимостью;
- 5НФ – решение проблем с зависимостями соединения.

В данной работе ставится задача довести БД до 3НФ. Рассмотрим подробнее нормальные формы до 3 включительно и проверим, соответствует ли БД критериям 3НФ.

Первая НФ: отношение находится в первой нормальной форме, если все атрибуты отношения являются простыми (требование атомарности атрибутов в реляционной модели), т.е. не имеют компонентов. Все атрибуты в проектируемой БД являются простыми, то есть БД удовлетворяет условиям 1НФ.

Вторая НФ: отношение находится во второй нормальной форме, если оно находится в 1НФ, и все неключевые атрибуты отношения функционально

полно зависят от первичного ключа отношения (в отношении атрибут В полностью зависит от атрибута А, если атрибут В функционально зависит от полного значения атрибута А и не зависит от какого-либо подмножества полного значения атрибута А). Так как в проектируемой БД нет составных ключей, то она заведомо находится во 2НФ.

Третья НФ: отношение находится в третьей нормальной форме, если оно находится во 2НФ, и нет транзитивных функциональных зависимостей неключевых атрибутов от первичного ключа (если для атрибутов А, В и С некоторого отношения существуют зависимости вида $A \rightarrow B$ и $B \rightarrow C$, то атрибут С транзитивно зависит от атрибута А через атрибут В). Проверяем БД на транзитивные функциональные зависимости неключевых атрибутов от первичного ключа и видим, что их нет. Таким образом, БД находится в 3 НФ.

Перейдём к следующему этапу логического проектирования, который заключается в поддержке целостности данных.

Для атрибута minute сущности effective_actions создадим домен, который будет накладывать численное ограничение. Так как матч идёт всего 90 минут, результативное действие может совершаться только с 1 по 90 минуту. Создаём скрипт добавления домена в БД (см. Приложение 1).

Были созданы следующие служебные таблицы (Таблица 3).

Таблица 3 – Служебные таблицы

| Служебная таблица | Описание |
|-------------------|---|
| cities | Содержит все города, в которых играют команды |
| stadiums | Содержит названия стадионов |
| player_positions | Содержит позиции игроков на футбольном поле |
| action_types | Содержит типы результативных действий |

Также необходимо обеспечить ограничение ссылочной целостности (ссылочное ограничение) – ограничение, согласно которому значения внешних ключей должны соответствовать значениям потенциальных ключей.

Есть следующие действия по поддержанию ссылочной целостности при удалениях строк:

- *cascade*: при удалении или обновлении записей родительской таблицы происходит удаление или обновление записей в дочерних таблицах;
- *no action/restrict*: не позволяют удалять или обновлять записи в родительских таблицах;
- *set null*: при удалении или обновлении записей в родительских таблицах в записях дочерних таблиц будет установлено неопределённое значение NULL;
- *set default*. при удалении или обновлении записей в родительских таблицах в записях дочерних таблиц будет установлено значение по умолчанию.

Установим действия по поддержанию ссылочной целостности для проектируемой БД (Таблица 4).

Таблица 4 – обеспечение ссылочной целостности

| Сущность | Внешний ключ | Действия |
|--------------|-----------------|---|
| teams | city | ON UPDATE CASCADE ON DELETE RESTRICT |
| | stadium | ON UPDATE CASCADE ON DELETE RESTRICT |
| players | team_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| | player_position | ON UPDATE CASCADE ON DELETE RESTRICT |
| applications | tournament_id | ON UPDATE CASCADE |

| | | |
|----------------------|----------------|---|
| | | ON DELETE RESTRICT |
| | team_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| matches | tournament_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| | team1_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| | team2_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| | stadium | ON UPDATE CASCADE ON DELETE RESTRICT |
| effective_actions | player_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| | match_id | ON UPDATE CASCADE ON DELETE RESTRICT |
| | type | ON UPDATE CASCADE ON DELETE RESTRICT |
| players_applications | player_id | ON UPDATE CASCADE ON DELETE CASCADE |
| | application_id | ON UPDATE CASCADE ON DELETE CASCADE |

В результате логического проектирования получим ER-диаграмму проектируемой БД (*Рисунок 1*).

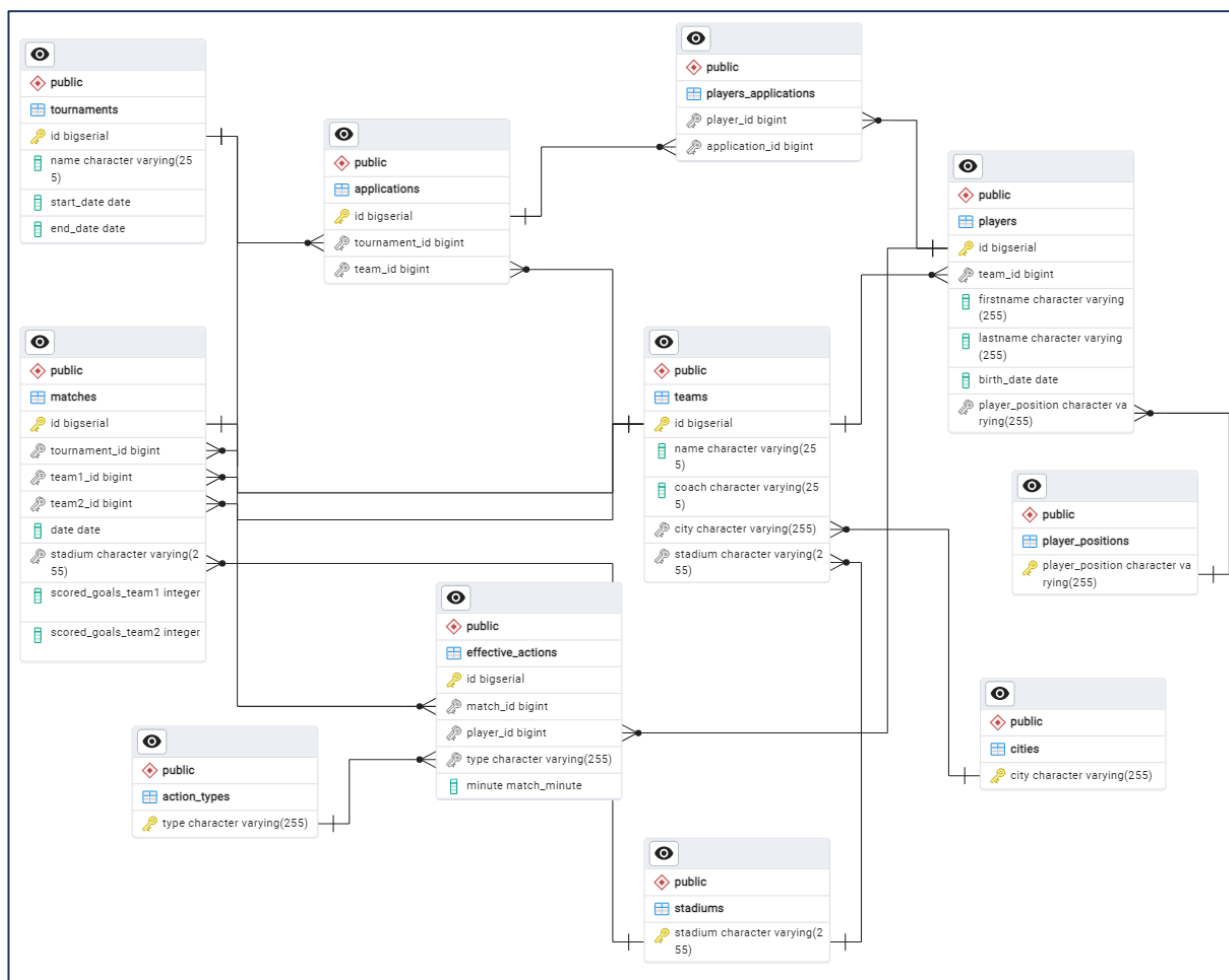


Рисунок 1 – ER-диаграмма

3. Физическое проектирование

Физическое проектирование – этап проектирования БД, целью которого является преобразование логической схемы с учетом синтаксиса и возможностей целевой СУБД.

На данном этапе идёт перенос логической схемы в среду целевой СУБД (PostgreSQL). В программе для визуального проектирования в режиме работы с ERD (ERD Tool) находим кнопку автоматического генерирования скрипта для создания БД. Создаём скрипт и запускаем его (см. Приложение 2). Таким образом, мы переносим логическую схему в среду целевой СУБД.

Запустим скрипт для добавления данных в БД (см. Приложение 3). Добавленная информация понадобится для проверки действий по обеспечению ссылочной целостности и создания представлений.

Проверим, что наше численное ограничение работает корректно. Попробуем добавить результирующее действие, совершенное на 100-й минуте матча, что невозможно (Рисунок 2).

```
INSERT INTO public.effective_actions(  
    id, match_id, player_id, type, minute)  
VALUES (10, 2, 2, 'Goal', 100);
```

Рисунок 2 – Скрипт для добавления результирующего действия

В результате получим ошибку (Рисунок 3).

| Data Output | Messages | Notifications |
|---|----------|---------------|
| ERROR: значение домена match_minute нарушает ограничение-проверку "match_minute_check" | | |
| ОШИБКА: значение домена match_minute нарушает ограничение-проверку "match_minute_check" | | |
| SQL state: 23514 | | |

Рисунок 3 – Ошибка при добавлении данных

Проверим действия по обеспечению ссылочной целостности. Попробуем удалить город (Рисунок 4).

```
DELETE FROM cities WHERE city = 'Yaroslavl'
```

Рисунок 4 – Скрипт для удаления города

В результате получим ошибку (*Рисунок 5*).

```
ERROR: На ключ (city)=(Yaroslavl) всё ещё есть ссылки в таблице "teams".UPDATE или DELETE в таблице "cities" нарушает ограничение внешнего ключа "teams_city_fkey" таблицы "teams"  
ОШИБКА: UPDATE или DELETE в таблице "cities" нарушает ограничение внешнего ключа "teams_city_fkey" таблицы "teams"  
SQL state: 23503  
Detail: На ключ (city)=(Yaroslavl) всё ещё есть ссылки в таблице "teams".
```

Рисунок 5 – Ошибка при удалении города

Попробуем обновить название города (*Рисунок 6*).

```
UPDATE cities SET city = 'Yar' WHERE city = 'Yaroslavl';
```

Рисунок 6 – Скрипт для обновления названия города

Всё успешно получилось, значит механизмы по обеспечению ссылочной целостности работают корректно (*Рисунок 7*).


| | city [PK] character varying (255)  |
|---|--|
| 1 | Pereslavl |
| 2 | Rostov Veliky |
| 3 | Rybinsk |
| 4 | Semibratovo |
| 5 | Tutaev |
| 6 | Uglich |
| 7 | Yar |

Рисунок 7. Результат после обновления названия города

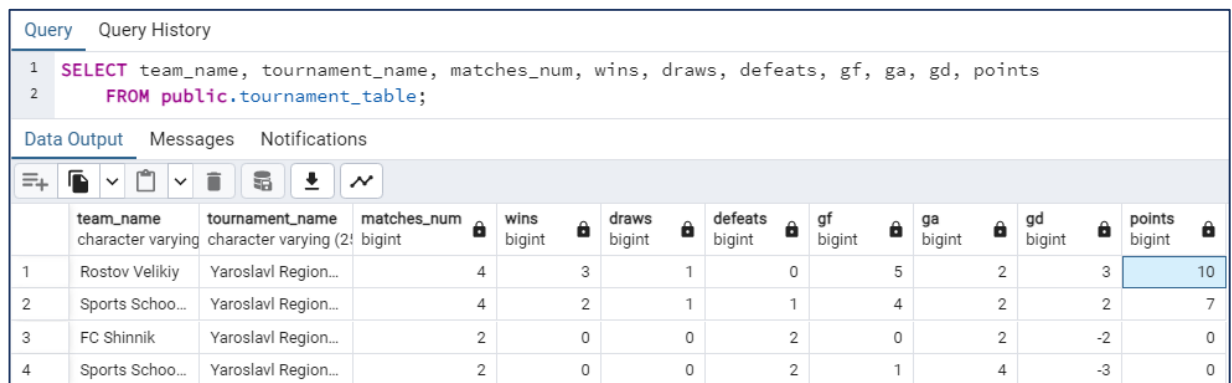
4. Создание представлений

Для упрощения доступа пользователей (клиентских приложений) к часто требуемым данным из разных таблиц или требующим обработки (агрегации) используются представления. Представление (VIEW) – виртуальная таблица, являющаяся сохраненным именованным запросом к БД.

Посредством представлений обеспечивается логическая независимость клиентских приложений от структуры БД: в случае изменений в схеме исходных таблиц достаточно скорректировать запрос представления, не меняя при этом схему самого представления.

Создадим следующие представления (см. Приложение 4):

- `tournament_table` – представление для просмотра положения команд в турнирной таблице, количества сыгранных командой матчей, количества забитых голов, количества пропущенных голов, разницы мячей (Рисунок 8).



The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `1 SELECT team_name, tournament_name, matches_num, wins, draws, defeats, gf, ga, gd, points` and `2 FROM public.tournament_table;`. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with 13 columns: `team_name`, `tournament_name`, `matches_num`, `wins`, `draws`, `defeats`, `gf`, `ga`, `gd`, and `points`. The table contains 4 rows of data.

| | team_name character varying | tournament_name character varying (21) | matches_num bigint | wins bigint | draws bigint | defeats bigint | gf bigint | ga bigint | gd bigint | points bigint |
|---|--------------------------------|---|-----------------------|----------------|-----------------|-------------------|--------------|--------------|--------------|------------------|
| 1 | Rostov Velikiy | Yaroslavl Region... | 4 | 3 | 1 | 0 | 5 | 2 | 3 | 10 |
| 2 | Sports Schoo... | Yaroslavl Region... | 4 | 2 | 1 | 1 | 4 | 2 | 2 | 7 |
| 3 | FC Shinnik | Yaroslavl Region... | 2 | 0 | 0 | 2 | 0 | 2 | -2 | 0 |
| 4 | Sports Schoo... | Yaroslavl Region... | 2 | 0 | 0 | 2 | 1 | 4 | -3 | 0 |

Рисунок 8 – Представление `tournament_table`

- `team_matches` – представление для просмотра всех игр для каждой команды, а также стадиона, даты и счёта (Рисунок 9).

| Query Query History | | | | | |
|--|----------------------------------|----------------------------|--------------|------------------------------------|---------------|
| <pre> 1 SELECT team1, team2, date, stadium, score 2 FROM public.teams_matches; </pre> | | | | | |
| Data Output Messages Notifications | | | | | |
| <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🔄</div> <div>📥</div> <div>📈</div> </div> | | | | | |
| | team1 character varying (255) | team2 character varying | date date | stadium character varying (255) | score text |
| 1 | FC Shinnik | Rostov Velikiy | 2024-06-06 | Spartak | 0:1 |
| 2 | FC Shinnik | Sports School №13 | 2024-05-26 | Slavneft | 0:1 |
| 3 | Rostov Velikiy | Sports School №4 | 2024-07-13 | Spartak | 2:1 |
| 4 | Rostov Velikiy | Sports School №13 | 2024-05-27 | Spartak | 2:1 |
| 5 | Rostov Velikiy | FC Shinnik | 2024-06-06 | Spartak | 1:0 |
| 6 | Rostov Velikiy | Sports School №13 | 2024-08-07 | Slavneft | 0:0 |
| 7 | Sports School №13 | Sports School №4 | 2024-07-15 | Slavneft | 2:0 |
| 8 | Sports School №13 | Rostov Velikiy | 2024-08-07 | Slavneft | 0:0 |
| 9 | Sports School №13 | FC Shinnik | 2024-05-26 | Slavneft | 1:0 |
| 10 | Sports School №13 | Rostov Velikiy | 2024-05-27 | Spartak | 1:2 |
| 11 | Sports School №4 | Sports School №13 | 2024-07-15 | Slavneft | 0:2 |
| 12 | Sports School №4 | Rostov Velikiy | 2024-07-13 | Spartak | 1:2 |

Рисунок 9 – Представление *team_matches*

- *player_age* – представление для просмотра всех игроков и их возраста (Рисунок 10).

| Query Query History | | | |
|--|--------------------------------------|----------------|---------------|
| <pre> 1 SELECT team_name, age, num 2 FROM public.player_age; </pre> | | | |
| Data Output Messages Notifications | | | |
| <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🔄</div> <div>📥</div> <div>📈</div> </div> | | | |
| | team_name character varying (255) | age integer | num bigint |
| 1 | Shpagin | 19 | 1 |
| 2 | Shelyakina | 19 | 2 |
| 3 | Kotkov | 19 | 3 |
| 4 | Michalev | 19 | 4 |
| 5 | Galanov | 19 | 5 |
| 6 | Krylov | 19 | 6 |
| 7 | Ilichev | 20 | 1 |
| 8 | Zimin | 20 | 2 |
| 9 | Safuanov | 20 | 3 |
| 10 | Pavlov | 20 | 4 |
| 11 | Ivanov | 21 | 1 |

Рисунок 10 – Представление – *player_age*

- *assists* – представление для просмотра всех ассистентов в порядке убывания по количеству отданных ассистов (Рисунок 11).

Query Query History

1

2

SELECT

lastname, name, assists

FROM public.assists;

Data Output Messages Notifications

| | lastname character varying (255) | name character varying (255) | assists bigint |
|---|-------------------------------------|---------------------------------|-------------------|
| 1 | Kotkov | Rostov Velikiy | 2 |
| 2 | Safuanov | Sports School №13 | 1 |
| 3 | Shpagin | Rostov Velikiy | 1 |

Рисунок 11 - Представление assists

- goals – представление для просмотра бомбардиров в порядке убывания по количеству забитых голов (Рисунок 12).

Query

Query History

1

SELECT lastname, name, goals

2

FROM public.goals;

Data Output

Messages

Notifications

| | lastname character varying (255) | name character varying (255) | goals bigint |
|---|-------------------------------------|---------------------------------|-----------------|
| 1 | Ilichev | Rostov Velikiy | 3 |
| 2 | Safuanov | Sports School №13 | 3 |
| 3 | Galanov | Sports School №13 | 1 |
| 4 | Pavlov | Sports School №13 | 1 |
| 5 | Shelyakina | FC Shinnik | 1 |
| 6 | Shpagin | Rostov Velikiy | 1 |

Рисунок 12 - Представление goals

- calender – представление для просмотра календаря всех прошедших и предстоящих матчей (Рисунок 13).

Query

Query History

1

SELECT team, opponent, date, stadium, score, points

2

FROM public.calender;

Data Output

Messages

Notifications

| | team character varying (255) | opponent character varying (255) | date date | stadium character varying (255) | score text | points integer |
|---|---------------------------------|-------------------------------------|--------------|------------------------------------|---------------|-------------------|
| 1 | Rostov Velikiy | Sports School №13 | 2024-05-27 | Spartak | 2:1 | 3 |
| 2 | Rostov Velikiy | FC Shinnik | 2024-06-06 | Spartak | 1:0 | 3 |
| 3 | Rostov Velikiy | Sports School №4 | 2024-07-13 | Spartak | 2:1 | 3 |
| 4 | Sports School №13 | FC Shinnik | 2024-05-26 | Slavneft | 1:0 | 3 |
| 5 | Sports School №13 | Sports School №4 | 2024-07-15 | Slavneft | 2:0 | 3 |
| 6 | Sports School №13 | Rostov Velikiy | 2024-08-07 | Slavneft | 0:0 | 1 |

Рисунок 13 – Представление calender

Заключение

В результате выполнения курсовой работы была создана реляционная БД на тему «Турниры по футболу в Ярославской области». Для этого были пройдены три основных этапа проектирования БД:

- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование.

Таким образом, в ходе работы мы ознакомились с процессом проектированием БД на языке SQL, научились создавать представления, которые упрощают работу с БД, а также изучили инструмент для визуального проектирования баз данных – pgAdmin 4.

Список литературы

[1] Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем: учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. — Москва: ИД «ФОРУМ»: ИНФРА-М, 2019. — 368 с. — Режим доступа: <https://lib.fbtuit.uz/assets/files/3.-...SQLNoSQL-..pdf> — Текст: электронный.

[2] Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с. — Режим доступа: https://edu.postgrespro.ru/sql_primer.pdf — Текст: электронный.

[3] Попова-Коварцева Д.А., Сопченко Е.В. Основы проектирования баз данных: учеб. пособие / Д.А. Попова-Коварцева, Е.В. Сопченко. — Самара: Изд-во Самарского университета, 2019. — 112 с. — Режим доступа: <http://surl.li/ujfuy> — Текст: электронный.

Приложение

Приложение 1 – Скрипт для добавления домена match_minute

```
CREATE DOMAIN public.match_minute
    AS integer;

ALTER DOMAIN public.match_minute OWNER TO postgres;

ALTER DOMAIN public.match_minute
    ADD CONSTRAINT match_minute_check CHECK (VALUE > 0
AND VALUE <= 90);
```

Приложение 2 – Скрипт для создания БД

```
BEGIN;

CREATE TABLE IF NOT EXISTS public.teams
(
    id bigserial NOT NULL,
    name character varying(255) NOT NULL,
    coach character varying(255) NOT NULL,
    city character varying(255) NOT NULL,
    stadium character varying(255) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.tournaments
(
    id bigserial NOT NULL,
    name character varying(255) NOT NULL,
    start_date date,
    end_date date,
    PRIMARY KEY (id)
);A

CREATE TABLE IF NOT EXISTS public.players
(
    id bigserial NOT NULL,
    team_id bigint NOT NULL,
    firstname character varying(255) NOT NULL,
    lastname character varying(255) NOT NULL,
    birth_date date NOT NULL,
```

```

        player_position character varying(255) NOT NULL,
        PRIMARY KEY (id)
    );

CREATE TABLE IF NOT EXISTS public.applications
(
    id bigserial NOT NULL,
    tournament_id bigint NOT NULL,
    team_id bigint NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.matches
(
    id bigserial NOT NULL,
    tournament_id bigint NOT NULL,
    team1_id bigint NOT NULL,
    team2_id bigint NOT NULL,
    date date NOT NULL,
    stadium character varying(255) NOT NULL,
    scored_goals_team1 integer NOT NULL DEFAULT 0,
    scored_goals_team2 integer NOT NULL DEFAULT 0,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.effective_actions
(
    id bigserial NOT NULL,
    match_id bigint NOT NULL,
    player_id bigint NOT NULL,
    type character varying(255) NOT NULL,
    minute match_minute NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.players_applications
(
    player_id bigint NOT NULL,
    application_id bigint NOT NULL
);

CREATE TABLE IF NOT EXISTS public.action_types
(
    type character varying(255) NOT NULL,

```



```

        PRIMARY KEY (type)
    );

CREATE TABLE IF NOT EXISTS public.player_positions
(
    player_position character varying(255) NOT NULL,
    PRIMARY KEY (player_position)
);

CREATE TABLE IF NOT EXISTS public.cities
(
    city character varying(255) NOT NULL,
    PRIMARY KEY (city)
);

CREATE TABLE IF NOT EXISTS public.stadiums
(
    stadium character varying(255) NOT NULL,
    PRIMARY KEY (stadium)
);

ALTER TABLE IF EXISTS public.teams
    ADD FOREIGN KEY (city)
    REFERENCES public.cities (city) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;

ALTER TABLE IF EXISTS public.teams
    ADD FOREIGN KEY (stadium)
    REFERENCES public.stadiums (stadium) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;

ALTER TABLE IF EXISTS public.players
    ADD FOREIGN KEY (team_id)
    REFERENCES public.teams (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;

```

```
ALTER TABLE IF EXISTS public.players
    ADD FOREIGN KEY (player_position)
    REFERENCES public.player_positions
    (player_position) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public.applications
    ADD FOREIGN KEY (team_id)
    REFERENCES public.teams (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public.applications
    ADD FOREIGN KEY (tournament_id)
    REFERENCES public.tournaments (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public.matches
    ADD FOREIGN KEY (tournament_id)
    REFERENCES public.tournaments (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public.matches
    ADD FOREIGN KEY (team1_id)
    REFERENCES public.teams (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID;
```

```
ALTER TABLE IF EXISTS public.matches
    ADD FOREIGN KEY (team2_id)
```

```

REFERENCES public.teams (id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
NOT VALID;

ALTER TABLE IF EXISTS public.matches
ADD FOREIGN KEY (stadium)
REFERENCES public.stadiums (stadium) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
NOT VALID;

ALTER TABLE IF EXISTS public.effective_actions
ADD FOREIGN KEY (match_id)
REFERENCES public.matches (id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
NOT VALID;

ALTER TABLE IF EXISTS public.effective_actions
ADD FOREIGN KEY (player_id)
REFERENCES public.players (id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
NOT VALID;

ALTER TABLE IF EXISTS public.effective_actions
ADD FOREIGN KEY (type)
REFERENCES public.action_types (type) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
NOT VALID;

ALTER TABLE IF EXISTS public.players_applications
ADD FOREIGN KEY (player_id)
REFERENCES public.players (id) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE CASCADE
NOT VALID;

```

```

ALTER TABLE IF EXISTS public.players_applications
    ADD FOREIGN KEY (application_id)
    REFERENCES public.applications (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID;

END;

```

Приложение 3 – Скрипт для добавления данных

```

INSERT INTO cities(city)
VALUES ('Rybinsk'),
('Rostov Veliky'),
('Pereslavl'),
('Tutaev'),
('Uglich'),
('Semibratovo'),
('Yaroslavl');

INSERT INTO public.player_positions(player_position)
VALUES ('Goalkeeper'),
('Right Back'),
('Left Back'),
('Centre Back'),
('Centre Midfielders'),
('Right Midfielder'),
('Left Midfielder'),
('Right Forward'),
('Left Forward'),
('Centre Forward');

INSERT INTO public.action_types(type)
VALUES ('Goal'),
('Assist');

INSERT INTO public.stadiums(stadium)
VALUES ('Spartak'),
('Slavneft'),
('Shinnik'),
('Metalist');

```

```

INSERT INTO public.teams(name, coach, city, stadium)
VALUES ('Rostov Velikiy', 'Sovetlyanov', 'Rostov
Velikiy', 'Spartak'),
('Sports School №13', 'Grigoriev', 'Yaroslavl',
'Slavneft'),
('FC Shinnik', 'Ivanov', 'Yaroslavl', 'Shinnik'),
('Sports School №4', 'Troitsky', 'Semibratovo',
'Metalist');

INSERT INTO public.players(team_id, firstname,
lastname, birth_date, player_position)
VALUES (1, 'Alexei', 'Shpagin', '2005-03-03',
'Goalkeeper'),
(1, 'Roman', 'Illichev', '2004-02-14', 'Left
Midfielder'),
(4, 'Dmitry', 'Zimin', '2004-05-14', 'Centre Back'),
(3, 'Nastya', 'Shelyakina', '2004-09-09', 'Centre
Forward'),
(1, 'Roman', 'Kotkov', '2004-07-24', 'Centre
Forward'),
(2, 'Pavel', 'Pavlov', '2004-05-07', 'Right Forward'),
(4, 'Ivan', 'Ivanov', '2003-01-13', 'Goalkeeper'),
(2, 'Artur', 'Safuanov', '2004-05-18', 'Left Back'),
(3, 'Kirill', 'Michalev', '2004-09-06', 'Right
Forward'),
(2, 'Pavel', 'Galanov', '2004-11-14', 'Centre
Forward'),
(3, 'Alexei', 'Krylov', '2004-08-11', 'Goalkeeper');

INSERT INTO public.tournaments(name, start_date,
end_date)
VALUES ('Yaroslavl Region Championship', '2024-05-17',
'2024-09-21'),
VALUES ('The Golden Ring Championship', '2024-06-29',
'2024-07-08');

INSERT INTO public.applications(tournament_id,
team_id)
VALUES (1, 1),
(1, 2),
(1, 3),
(1, 4),
(2, 1),

```

```

(2, 2),
(2, 3),
(2, 4);

INSERT INTO public.players_applications(player_id,
application_id)
VALUES (1, 1),
(2, 1),
(5, 1),
(6, 6),
(8, 6),
(10, 6),
(11, 6);

INSERT INTO public.matches(tournament_id, team1_id,
team2_id, date, city, stadium, scored_goals_team1,
scored_goals_team2)
VALUES (1, 1, 2, '2024-05-27', 'Rostov Velikiy',
'Spartak', 2, 1),
(1, 1, 3, '2024-06-06', 'Rostov Velikiy', 'Spartak',
1, 0),
(1, 1, 4, '2024-07-13', 'Rostov Velikiy', 'Spartak',
2, 1),
(1, 2, 1, '2024-08-07', 'Yaroslavl', 'Slavneft', 0,
0),
(1, 2, 3, '2024-05-26', 'Yaroslavl', 'Slavneft', 1,
0),
(1, 2, 4, '2024-07-15', 'Yaroslavl', 'Slavneft', 2,
0);

INSERT INTO public.effective_actions(match_id,
player_id, type, minute)
VALUES (1, 2, 'Goal', 34),
(1, 2, 'Goal', 68),
(1, 1, 'Assist', 68),
(1, 5, 'Assist', 34),
(1, 6, 'Goal', 23),
(1, 8, 'Goal', 23),
(1, 8, 'Goal', 23),
(6, 10, 'Goal', 4),
(6, 8, 'Goal', 17),
(6, 8, 'Assist', 4),
(3, 2, 'Goal', 15),
(3, 1, 'Goal', 44),

```

```
(3, 5, 'Assist', 15),  
(3, 4, 'Goal', 85);
```

Приложение 4 – Скрипт для создания представлений

```
CREATE OR REPLACE VIEW public.tournament_table  
AS  
SELECT  
t.name AS team_name,  
tour.name AS tournament_name,  
count(*) AS matches_num,  
sum(  
    CASE  
        WHEN t.id = m.team1_id AND  
m.scored_goals_team1 > m.scored_goals_team2 THEN 1  
        WHEN t.id = m.team2_id AND  
m.scored_goals_team1 < m.scored_goals_team2 THEN 1  
        ELSE 0  
    END) AS wins,  
sum(  
    CASE  
        WHEN m.scored_goals_team1 =  
m.scored_goals_team2 THEN 1  
        ELSE 0  
    END) AS draws,  
sum(  
    CASE  
        WHEN t.id = m.team1_id AND  
m.scored_goals_team1 < m.scored_goals_team2 THEN 1  
        WHEN t.id = m.team2_id AND  
m.scored_goals_team1 > m.scored_goals_team2 THEN 1  
        ELSE 0  
    END) AS defeats,  
sum(  
    CASE  
        WHEN t.id = m.team1_id THEN  
m.scored_goals_team1  
        WHEN t.id = m.team2_id THEN  
m.scored_goals_team2  
        ELSE 0  
    END) AS gf,  
sum(  
    CASE
```

```

        WHEN t.id = m.team1_id THEN
m.scored_goals_team2
        WHEN t.id = m.team2_id THEN
m.scored_goals_team1
        ELSE 0
    END) AS ga,
sum(
    CASE
        WHEN t.id = m.team1_id THEN
m.scored_goals_team1 - m.scored_goals_team2
        WHEN t.id = m.team2_id THEN
m.scored_goals_team2 - m.scored_goals_team1
        ELSE 0
    END) AS gd,
sum(
    CASE
        WHEN m.scored_goals_team1 >
m.scored_goals_team2 AND t.id = m.team1_id THEN 3
        WHEN m.scored_goals_team2 >
m.scored_goals_team1 AND t.id = m.team2_id THEN 3
        WHEN m.scored_goals_team2 =
m.scored_goals_team1 AND (t.id = m.team1_id OR t.id =
m.team2_id) THEN 1
        ELSE 0
    END) AS points
FROM teams t
JOIN matches m ON m.team1_id = t.id OR m.team2_id =
t.id
JOIN tournaments tour ON tour.id = m.tournament_id
GROUP BY t.name, tour.name
ORDER BY tour.name, points DESC, t.name;

CREATE OR REPLACE VIEW public.teams_matches
AS
SELECT t.name AS team1,
    CASE
        WHEN t.id = m.team1_id THEN t2.name
        WHEN t.id = m.team2_id THEN t3.name
        ELSE NULL::character varying
    END AS team2,
m.date,
m.stadium,
    CASE

```



```

        WHEN t.id = m.team1_id THEN
concat(m.scored_goals_team1::text, ': ',
m.scored_goals_team2::text)
        WHEN t.id = m.team2_id THEN
concat(m.scored_goals_team2::text, ': ',
m.scored_goals_team1::text)
        ELSE NULL::text
    END AS score
FROM teams t
    JOIN matches m ON m.team1_id = t.id OR m.team2_id
= t.id
    JOIN teams t2 ON m.team2_id = t2.id
    JOIN teams t3 ON m.team1_id = t3.id
ORDER BY t.name;

CREATE OR REPLACE FUNCTION public.find_years(
    birth_date date)
    RETURNS integer
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
SELECT date_part('year', age(birth_date));
$BODY$;

CREATE OR REPLACE VIEW public.player_age
AS
SELECT
lastname AS team_name,
find_years(birth_date) AS age,
row_number() OVER (PARTITION BY
(find_years(birth_date))) AS num
FROM players p
ORDER BY (find_years(birth_date));

CREATE OR REPLACE VIEW public.goals
AS
SELECT
p.lastname,
t.name,
count(a.type) AS goals
FROM players p
JOIN effective_actions a ON p.id = a.player_id
JOIN teams t ON p.team_id = t.id

```

```

WHERE a.type::text = 'Goal'::text
GROUP BY p.lastname, t.name
ORDER BY goals DESC, p.lastname;

CREATE OR REPLACE VIEW public.assists
AS
SELECT
p.lastname,
t.name,
count(a.type) AS assists
FROM players p
JOIN effective_actions a ON p.id = a.player_id
JOIN teams t ON p.team_id = t.id
WHERE a.type::text = 'Assist'::text
GROUP BY p.lastname, t.name
ORDER BY (count(a.type)) DESC, p.lastname;

CREATE OR REPLACE VIEW public.calender
AS
SELECT
t1.name AS team,
t2.name AS opponent,
m.date,
m.stadium,
concat(m.scored_goals_team1::text, ':',
m.scored_goals_team2::text) AS score,
CASE
    WHEN m.scored_goals_team1 > m.scored_goals_team2
THEN 3
    WHEN m.scored_goals_team1 = m.scored_goals_team2
THEN 1
    ELSE 0
END AS points
FROM matches m
JOIN teams t1 ON t1.id = m.team1_id
JOIN teams t2 ON t2.id = m.team2_id
ORDER BY t1.name, m.date;

```