МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт Компьютерных наук и технологий
Высшая школа искусственного интеллекта
Направление 02.03.01 Математика и Компьютерные науки

Отчёт по дисциплине: "Теоретические основы базы данных"
Курсовая работа
"Формирование мирового рейтинга футболистов в ценовой характеристике"
3 курс, группа: 3530201/00101

Студент: _____ Боева Анастасия Владимировна

Преподаватель: _____ Попов Сергей Геннадьевич

Санкт-Петербург - 2023

Содержание

1 Аналитика

1.1 Описание предметной области

Рассматриваемая предметная область - мировой рейтинг футболистов в ценовой характеристике. Мировой рейтинг учитывает результат предыдущего сезона для всех амплуа футболистов - полузащитников, защитников и нападающих. Рейтинг обновляется один раз в год - в период с июня по август, и обусловлен перерывом мировых чемпионатов.
Цель формирования мирового рейтинга:

- на его основе футбольные клубы могут ориентироваться на стоимости футболистов, обусловленные их характеристикой;
- помощь в составлении спортивных статей расходов;
- поскольку все, наиболее важные, характеристики собраны в одном месте, пользователю удобно проводить сравнительный анализ.

Характеристика рассчитывается отдельно для каждого из амплуа футболистов: полузащитников, защитников и нападающих. При этом итоговый рейтинг является общим для всех описанных выше спортсменов.
Принцип формирования рейтинга.
Попасть в рейтинг футболисты могут, приняв участие в официальном матче, к участию в которых допускаются игроки не младше шестнадцати лет.
Причиной удаления футболиста из рейтинга является его официальное заявление об окончании карьеры.
Ни один из футболистов рейтинга не имеет нулевую стоимость. Это обусловлено тем, что:

- по достижении 16-летнего возраста игрок имеет официальное право для перехода в другие клубы;
- каждый футболист на начальном этапе своей карьеры подписывает официальный контракт.

Позиция участника рейтинга зависит от: лиги, атакующей характеристики, скорости бега, дриблинга, физической готовности и точности передач.
Данные берутся из авторитетного источника (сайт Whoscored) - уникальный всеобъемлющий статистический алгоритм, который базируется на более чем 200 статистических показателях, каждый из которых учитывается с различной степенью важности.
Рейтинг обновляется один раз в год - в период с июня по август, и обусловлен перерывом мировых чемпионатов.
После проведения последнего из чемпионатов рейтинг обновляется.
Описание футбольных качеств.

- Атакующие характеристики - вероятность того, что удар игрока завершится голом (численная характеристика);
- Скорость бега (км/ч);
- Дриблинг - количество выполненных уходов с мячом от противников (%);
- Физическая подготовка - мера, пройденная футболистом за игру (м);
- Точность передач - количество до конца доведенных передач к общему числу (%);

- Оборонительная характристика - среднее количество отобранных мячей у противника (численная характеристика).

Принцип выставления очков за конкретное футбольное качество.
Количество цены, которую участник рейтинга получает за конкретную футбольную характеристику, зависит от количественной характеристики этой футбольной характеристики, и начисляется в соответствии с Таблицей 1, где:

АХ - атакующие характеристики
СБ - скорость бега
Д - дриблинг
ФП - физическая подготовка
ТП - точность передач
ОХ - оборонительная характеристика

Количественная характеристика	Футбольная характеристика					
Цена	АХ	СБ	Д	ФП	ТП	ОХ
1	5	16	5	100	23	0,5
2	6	17	7	200	25	0,7
3	7	18	10	300	27	1,0
4	8	19	12	400	30	1,2
5	9	20	15	500	33	1,5
6	10	21	17	600	35	1,7
7	11	22	20	700	37	2,0
8	[12, 15]	23	23	800	40	2,5
9	[16,20]	24	26	900	42	2,7
10	21	25	30	1000	45	3
11	22	26	32	1100	50	3,5
12	23	27	35	1200	55	3,7
13	24	28	37	1300	60	4,0
14	25	29	40	1400	65	4,2
15	26	30	45	1500	70	4,5
16	27	31	47	1650	75	4,7
17	28	32	50	1800	80	5,0
18	29	33	55	1900	85	5,5
19	30	34	60	2000	87	5,7
20	31	35	65	2100	90	6

Таблица 1: Распределение цены в зависимости от футбольной характеристики и её числовой характеристики

Описание чемпионатов.
При расчете рейтинга очки выставляются только за участие в следующих соревнованиях:

- EPL - профессиональная футбольная лига, высший дивизион в системе футбольных лиг Англии;
- BUNDESLIGA - профессиональная футбольная лига, высший дивизион в системе футбольных лиг Германии;
- Serie A - профессиональная футбольная лига, высший дивизион системы футбольных лиг Италии;

- Ligue One Uber Eats - профессиональная футбольная лига, высший дивизион системы футбольных лиг Франции;
- LaLiga - профессиональная футбольная лига, высший дивизион в системе футбольных лиг Испании;
- Liga NOS - профессиональная футбольная лига, высший дивизион в системе футбольных лиг Португалии;
- Eredivisie - профессиональная футбольная лига, высший дивизион в системе футбольных лиг Нидерландов;
- Российская Премьер Лига - профессиональная футбольная лига, высший дивизион в системе футбольных лиг России;
- Süper Lig - профессиональная футбольная лига, высший дивизион в системе футбольных лиг Турции.

Принцип выставления очков за конкретное соревнование.
Количество цены, которую участник рейтинга получает за конкретный чемпионат, зависит от лиги и начисляется в соответствии с Таблицей 2, где:

Цена	Чемпионат
10	EPL
10	BUN
10	SeA
10	LiO
10	LaL
5	LiN
5	ERE
5	RPL
5	SuL

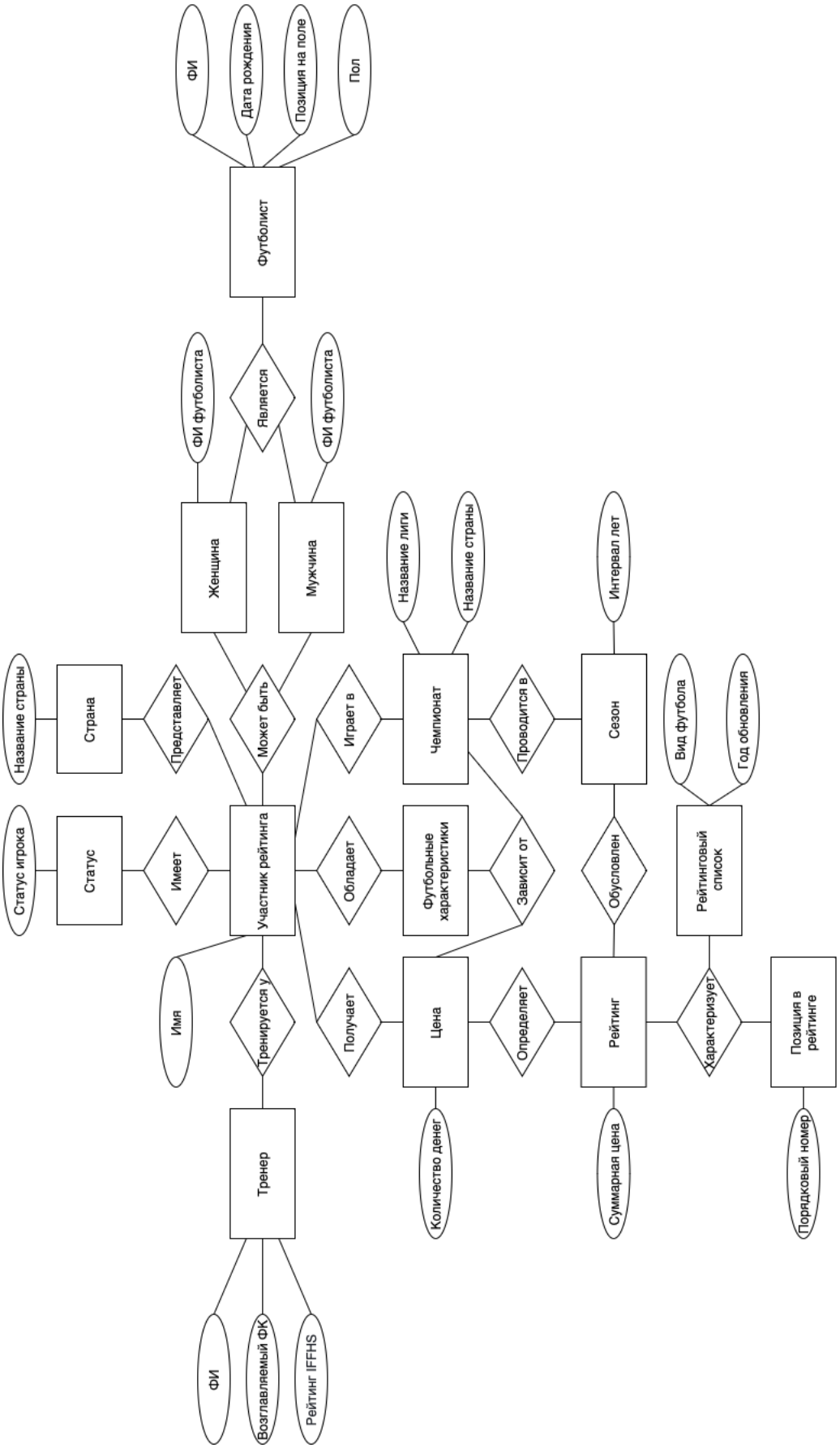
Таблица 2: Распределение цены в зависимости от чемпионата и лиги

1.2 Сущность и атрибуты

1. Участник рейтинга - футболист, который играет в одной из сборных - женскую или мужскую.
    - имя.
2. Статус - профессионал или любитель.
    - статус игрока.
3. Страна - страна, которую футболист представляет на футбольных матчах.
    - название страны.
4. Тренер - специалист в футболе, руководящий тренировкой футболистов.
    - ФИ;
    - возглавляемый ФК;
    - рейтинг IFFHS.
5. Женщина - женщина-футболист, которая выступает за вид футбола: женский (стандартный) футбол.
    - ФИ футболиста.
6. Мужчина - мужчина-футболист, который выступает за вид футбола: мужской (стандартный) футбол.
    - ФИ футболиста.
7. Футболист - человек, профессионально играющий в футбол.
    - ФИ;
    - дата рождения;
    - пол;
    - позиция на поле.
8. Чемпионат - игра между двумя футбольными командами в пределах одной лиги.
    - лига;
    - страна.
9. Футбольные характеристики - качества, характеризующие конкретнегого футболиста.
    - атакующие характеристики;
    - скорость бега;
    - дриблинг;
    - физическая подготовка;
    - точность передач;
    - оборонительная характеристика.

10. Сезон - период времени, в который проводились чемпионаты.
    - интервал лет.
11. Цена - количество денег (валюта - евро), определяющие футболиста, за каждую футбольную характеристику и чемпионат.
    - количество денег.
12. Рейтинг - итоговая цена, которую получил футболист за конкретный сезон.
    - итоговая цена.
13. Позиция в рейтинге - номер положения футболиста в рейтинговом списке.
    - порядковый номер.
14. Рейтинговый список - ранжированный список футболистов в порядке убывания их рейтинга.
    - вид футбола;
    - год обновления.

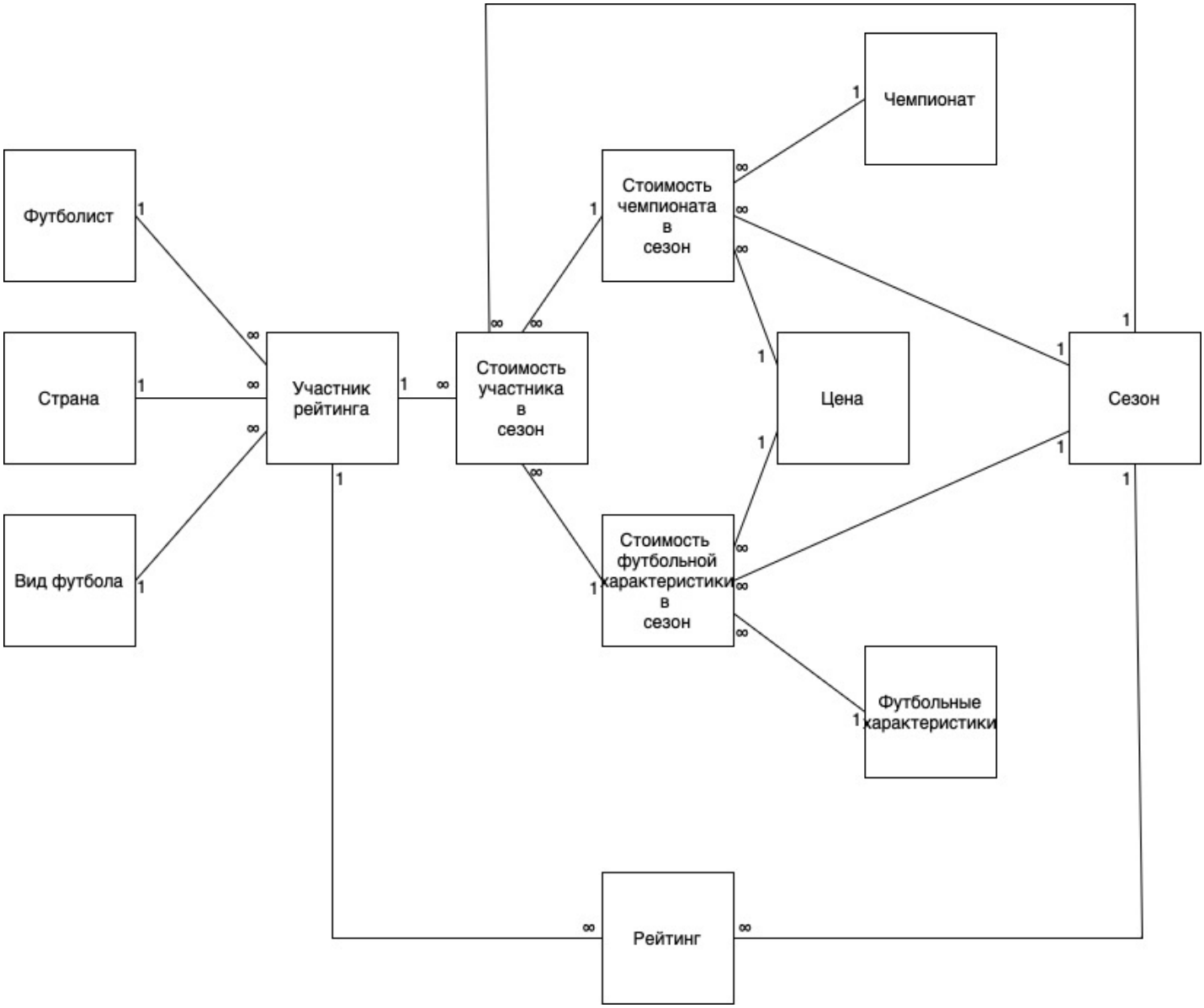
1.3 ER-диаграмма



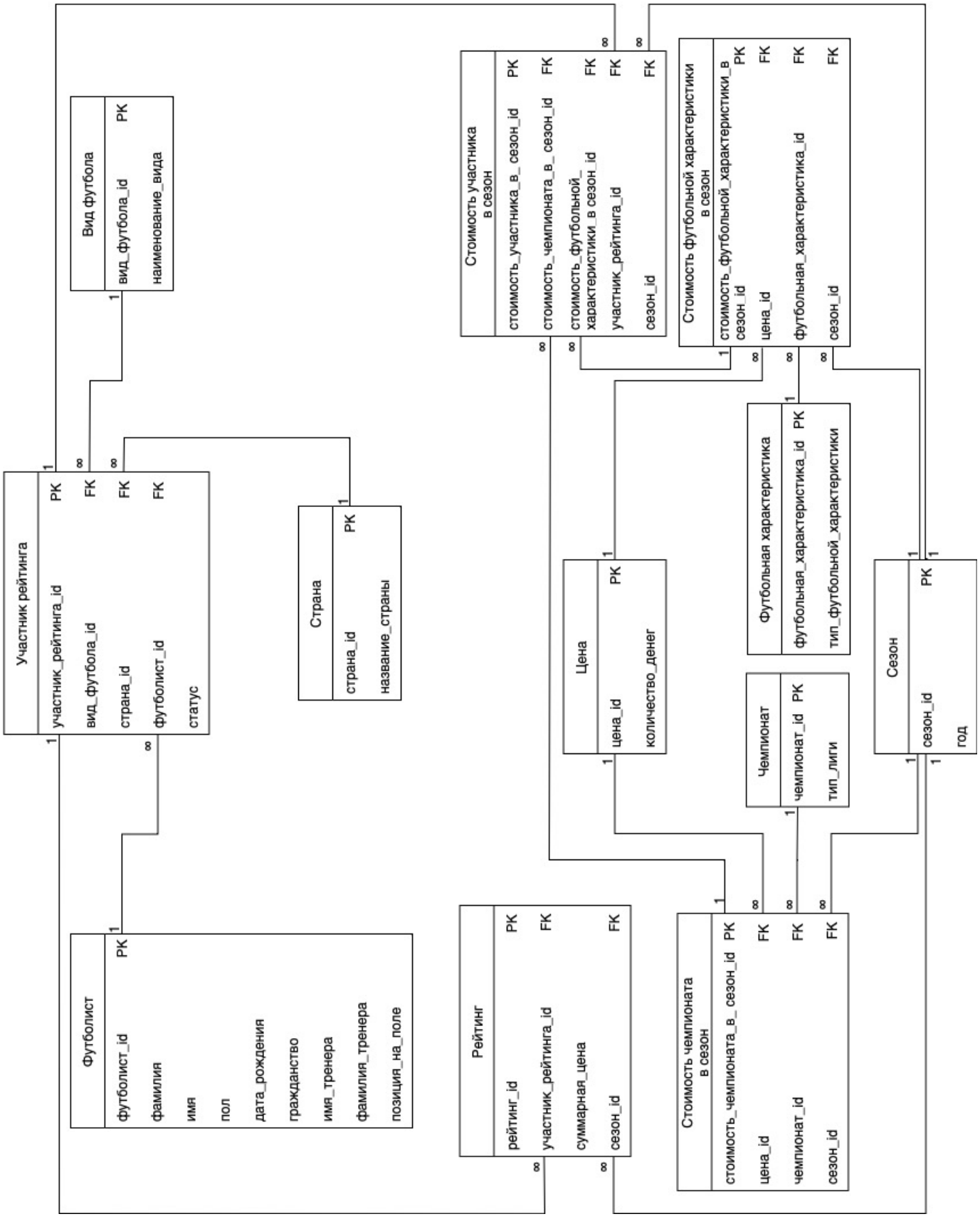
1.3.1 Чтение ER-диаграммы

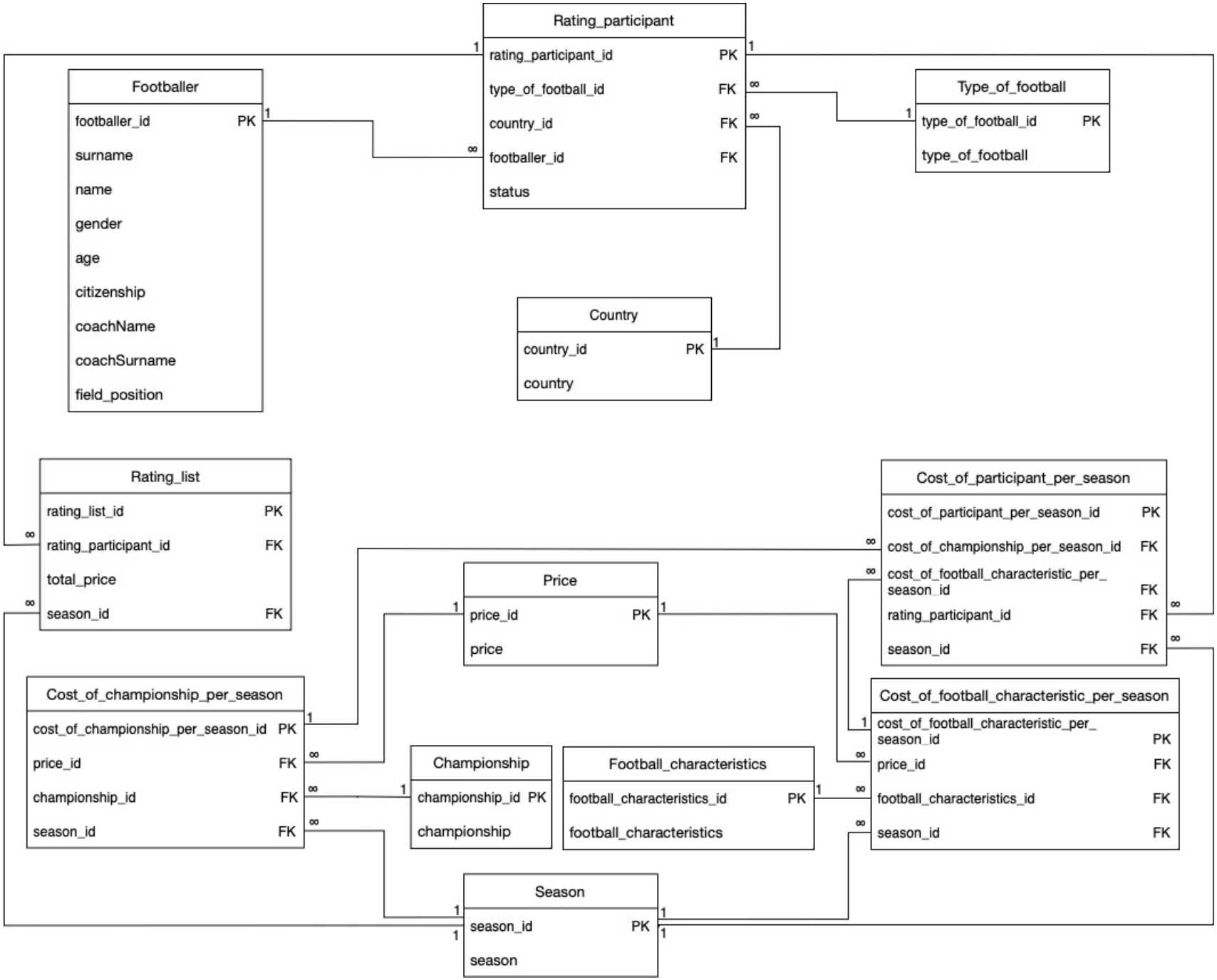
1. Участниками рейтинга могут быть либо мужчины, либо женщины.
2. Женщина является футболистом.
3. Мужчина является футболситом.
4. Участник рейтинга представляет страну.
5. Участник рейтинга имеет статус.
6. Участник рейтинга играет в чемпионатах.
7. Чемпионаты проводятся в разные сезоны.
8. Участник рейтинга обладает футбольными характеристиками.
9. Участник рейтинга получает цену.
10. Цена зависит от футбольных характеристик, которыми обладает участник рейтинга, и от чемпионата, в котором играет участник рейтинга.
11. Цена определяет рейтинг.
12. Рейтинг обусловлен сезоном, в котором проводился чемпионат.
13. Рейтинг характеризует позицию рейтинга в рейтинговом списке.

1.4 Схема объектов



1.5 Схема базы данных




Rating_participant
rating_participant_id PK
type_of_football_id FK
country_id FK
footballer_id FK
status
Footballer
footballer_id PK
surname
name
gender
age
citizenship
coachName
coachSurname
field_position
Type_of_football
type_of_football_id PK
type_of_football
Country
country_id PK
country
Rating_list
rating_list_id PK
rating_participant_id FK
total_price
season_id FK
Price
price_id PK
price
Cost_of_participant_per_season
cost_of_participant_per_season_id PK
cost_of_championship_per_season_id FK
cost_of_football_characteristic_per_season_id FK
rating_participant_id FK
season_id FK
Cost_of_championship_per_season
cost_of_championship_per_season_id PK
price_id FK
championship_id FK
season_id FK
Championship
championship_id PK
championship
Football_characteristics
football_characteristics_id PK
football_characteristics
Cost_of_football_characteristic_per_season
cost_of_football_characteristic_per_season_id PK
price_id FK
football_characteristics_id FK
season_id FK
Season
season_id PK
season

1.6 Атрибуты таблиц баз данных

В Таблице 3 - Таблице 14 представлены атрибуты каждой таблицы проектируемой базы данных. Перечислены тип переменной, размер переменной, ограничения атрибута и тип ключа, если поле является ключом.
PK - первичный ключ, FK - внешний ключ.

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	страна_id	country_id	INT	NOT NULL	PK	-
2	страна	country	VARCHAR(58)*	NOT NULL	-	-

Таблица 3: Страна - Country

* Ограничение количества символов для страны равно 58, потому что мировой рекорд по длине названия страны принадлежит Великобритании. Её полное официальное наименование - Соединённое Королевство Великобритании и Северной Ирландии.

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	тип_футбола_id	type_of_football_id	INT	NOT NULL	PK	-
2	тип_футбола	type_of_football	ENUM('Male', 'Female')	NOT NULL	-	-

Таблица 4: Тип футбола - Type of football

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	чемпионат_id	championship_id	INT	NOT NULL	PK	-
2	тип_лиги	championship	ENUM ('EPL','BUNDESLIGA', 'SerieA', 'LigueOneUberEats', 'LaLiga', 'LigaNOS' 'Eredivisie', 'RPL', 'SuperLig')	NOT NULL	-	-

Таблица 5: Чемпионат - Championship

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	футбольная_характеристика_id	football_characteristics_id	INT	NOT NULL	PK	-
2	футбольная_характеристик	football_characteristics	ENUM ('AttackingCharacteristics', 'RunningSpeed', 'Dribbling', 'PhysicalTraining', 'TransmissionAccuracy', 'DefensiveCharacteristic')	NOT NULL	-	-

Таблица 6: Футбольная характеристика - Football characteristics

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	сезон_id	season_id	INT	NOT NULL	PK	-
2	сезон	season	INT	NOT NULL	-	-

Таблица 7: Сезон - Season

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	цена_id	price_id	INT	NOT NULL	PK	-
2	цена	price	INT	NOT NULL	-	-

Таблица 8: Цена - Price

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	футболсит_id	footballer_id	INT	NOT NULL	PK	-
2	фамилия	surname	VARCHAR(50)*	NOT NULL	-	-
3	имя	name	VARCHAR(50)*	NOT NULL	-	-
4	пол	gender	ENUM('Man', 'Woman')	NULL	-	-
5	возраст	age	INT	NOT NULL	-	-
6	гражданство	citizenship	VARCHAR(58)*	NOT NULL	-	-
7	имя_тренера	coachName	VARCHAR(50)*	NOT NULL	-	-
8	фамилия_тренера	coachSurname	VARCHAR(50)*	NOT NULL	-	-
9	позиция_на_поле_id	field_position_id	ENUM ('Midfielder', 'Defender', 'Forward')	NOT NULL	-	-

Таблица 9: футболист - Footballer

* Ограничение количества символов для фамилии и имени равно 50, потому что максимальная длина имени и фамилии пользователя на Facebook в настоящее время составляет 50 символов.

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	участник_рейтинга_id	rating_participant_id	INT	NOT NULL	PK	-
2	тип_футбола_id	type_of_football_id	INT	NOT NULL	FK	Type_of_ football (type_of_ football_id)
3	страна_id	country_id	INT	NOT NULL	FK	Country (country_id)
4	футболист_id	footballer_id	INT	NOT NULL	FK	Footballer (footballer_id)
5	статус	status_id	INT	NOT NULL	-	-

Таблица 10: Участник рейтинга - Rating participant

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	рейтинговый_список_id	rating_list_id	INT	NOT NULL	PK	-
2	участник_рейтинга_id	rating_participant_id	INT	NOT NULL	FK	Rating_ participant (rating_ participant_id)
3	суммарная_цена	total_price	INT	NOT NULL	-	-
4	сезон_id	season_id	INT	NOT NULL	FK	Season (season_id)

Таблица 11: Рейтинговый список - Rating list

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	стоимость_футбольной_ характеристики_в_сезон_id	cost_of_football_ charscteristics_per_season_id	INT	NOT NULL	PK	-
2	цена_id	price_id	INT	NOT NULL	FK	Price (price_id)
3	футбольная_характеристика_id	football_characteristics_id	INT	NOT NULL	FK	Football_characteristics (football_characteristics_id)
4	сезон_id	season_id	INT	NOT NULL	FK	Season (season_id)

Таблица 12: Стоимость футбольной характеристики в сезон - Cost of football characteristic per season

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	стоимость_чемпионата_в_сезон_id	cost_of_championship_per_season_id	INT	NOT NULL	PK	-
2	цена_id	price_id	INT	NOT NULL	FK	Price (price_id)
3	чемпионат_id	championship_id	INT	NOT NULL	FK	Championship (championship_id)
4	сезон_id	season_id	INT	NOT NULL	FK	Season (season_id)

Таблица 13: Стоимость чемпионата в сезон - Cost of championship per season

№	Название рус.	Название англ.	Тип	Ограничения	Тип ключа	Ссылка
1	стоимость_участника_в_сезон_id	cost_of_championship_per_season_id	INT	NOT NULL	PK	-
2	стоимость_чемпионата_в_сезон_id	cost_of_championship_per_season_id	INT	-	FK	Cost_of_championship_per_season_id (cost_of_championship_per_season_id)
3	стоимость_футбольной_характеристики_в_сезон_id	cost_of_football_charscteristics_per_season_id	INT	-	FK	Cost_of_football_characteristic_per_season_id (cost_of_football_characteristic_per_season_id)
4	участник_рейтинга_id	rating_participant_id	INT	NOT NULL	FK	Rating_participant (rating_participant_id)
5	сезон_id	season_id	INT	NOT NULL	FK	Season (season_id)

Таблица 14: Стоимость участника в сезон - Cost of participant per season

2 Программирование

2.1 Создание базы данных

Для создания базы данных был написан скрипт. Код скрипта представлен в Приложении В.
Перед созданием базы данных используется инструкция DROP DATABASE IF EXISTS, которая удаляет базу данных с таким же названием, если она существовала до этого.
Для создания базы данных используется инструкция CREATE DATABASE. Затем, чтобы перейти к использованию созданной базы данных применяется инструкция USE.
С помощью инструкции CREATE TABLE создаются таблицы, перечисленные в п. 1.6.
Результатом выполнения скрипта является база данных "bd", представленная на Рис. 1

Рис. 1: Результат создания базы данных

2.2 Заполнение базы данных

Заполнение базы данных происходит автоматически сгенерированными данными. Для этого была реализована программа на языке Java 8, которая генерирует файлы с SQL-инструкциями INSERT INTO, которые используются для вставки новых строк в таблицы. Получившиеся файлы формата .sql импортируются через командную строку командой:

mysql -u root -p < *имя файла*.sql

Данные делятся на два типа:

- Словари
  - Данные в таблицах Type_of_football, Country, Championship, Football_characteristics являются настоящими данными предметной области;
  - Данные в таблице Price сгенерированы от минимального значения 10 до максимального значения 20, что соответсвует количеству денег в тысячах за ту или иную характеристику/чемпионат.

  - Данные в таблице Footballer заполнены по следующему принципу: первые 750 записей - мужские имена и фамилии, записи с 751 по 1500 - женские имена и фамилии. Поля second_name и first_name генерируются рандомно из предварительно заполненных массивов реальных имен и фамилий на английском языке.
  - В таблице Season представлены годы с 2013 по 2023.

- Таблицы с нормальным распределением данных.
  - В таблице Rating_participant первые 750 записей создаются со значением поля type_of_football_id = 1, что является id дисциплины мужского футбола, и значениями footballer_id в диапазоне [1..750], так как в этом диапазоне id находятся мужские Имена и Фамилии в таблице Footballer. Следующие 100 записей создаются со значением поля discipline_id = 2, что является id дисциплины женского футбола, и значениями footballer_id в диапазоне [751..1500], так как в этом диапазоне id находятся женские Имена и Фамилии в таблице Footballer. Поле footballer_id генерируется так, что id футболиста может появится в этой таблице от 5 до 10 раз. Поле country_id заполняется рандомными странами из диапазона [1..180]. Поле status рандомно заполняется значениями: "active" или "finished".
  - В таблицах Cost_of_football_characteristic_per_season и Cost_of_championship_per_season поля генериуется по принципу: для каждой футбольной характристики за каждый сезон своя рандомная цена из диапазона [1...20] таблицы Price.
  - В таблице Cost_of_participant_per_season поля генерируются так: для каждого участника рейтинга генерируется рандомное количество [10,...,15] чемпионатов, в которых он мог учавствовать, либо генерируется рандомное количество [1,...,5] футбольных характеристик.
  - В таблице Rating_list поля генерируются по принципу: для каждого участника рейтинга 20 лет есть своя суммарная цена. Значение рейтинга - суммарная цена, заполняется рандомными значениями из массива чисел.

Количество записей для каждой из таблиц и очередность заполнения таблиц представлено в Таблице 15.

Название таблицы	**Порядковый номер при заполнении**	**Количество записей**
Country	I	182
Type_of_football	I	2
Championship	I	9
Football_characteristics	I	6
Season	I	20
Price	I	20
Footballer	I	1500
Rating_participant	II	1500
Cost_of_football_characteristic_per_season	II	120
Cost_of_championship_per_season	II	180
Cost_of_participant_per_season	III	26412
Rating_list	IV	11274
TOTAL	-	41225

Таблица 15: Заполнение базы данных

3 Запросы к базе данных

В рамках выполнения данной курсовой работы требуется реализовать 8 запросов к базе данных:

1. Вывести всех участников, цена которых выше 990 и которые участвовали в чемпионате EPL и у них была характеристика AttackingCharacteristics.
2. Посчитать число футболистов у которых присутствует характеристика Dribbling и вид футбола Male.
3. Для каждого чемпионата посчитать число участников.
4. Найти чемпионат с:
   (a) Наибольшим числом участников.
   (b) Наименьшим числом участников.
5. Посчитать число участников с одинаковым суммарной ценой.
6. Найти футбольную характеристику, число футболистов которой больше, чем в характеристике RunningSpeed.
7. Вывести фамилии и имена всех футболистов, которые имеют позицию на поле «Midfielder» и у которых есть футбольная характеристика «PhysicalTraining», цена которой - 30.
8. Для каждого вида футбола и чемпионата посчитать число участников.

3.1 Запрос 1

Формулировка запроса
Вывести всех участников, суммарная цена которых выше 990 и которые участвовали в чемпионате «EPL», и у них была футбольная характеристика «Attacking Characteristics».
SQL-запрос

```
SELECT Footballer.surname,
       Footballer.name,
       Rating_list.total_price,
       Championship.championship,
       Football_characteristics.football_characteristics
FROM Footballer
JOIN Rating_participant ON Footballer.footballer_id =
↪  Rating_participant.footballer_id
JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
JOIN Season ON Rating_list.season_id = Season.season_id
JOIN Cost_of_championship_per_season ON Season.season_id =
↪  Cost_of_championship_per_season.season_id
JOIN Championship ON Cost_of_championship_per_season.championship_id =
↪  Championship.championship_id
JOIN Cost_of_football_characteristic_per_season ON Season.season_id =
↪  Cost_of_football_characteristic_per_season.season_id
JOIN Football_characteristics ON
↪  Cost_of_football_characteristic_per_season.football_characteristics_id =
↪  Football_characteristics.football_characteristics_id
```

```
WHERE ((Rating_list.total_price > 990)
       AND (Championship.championship = 'EPL')
       AND (Football_characteristics.football_characteristics =
↪  'AttackingCharacteristics'));
```

Результат

На Рис. 2 представлена часть результата выполнения запроса 1. Приведено первые несколько строк, всего запрос вернул 120 строк.

surname	name	total_price	championship	football_characteristics
Barnes	Molly	1000	EPL	AttackingCharacteristics
Dean	Sara	1000	EPL	AttackingCharacteristics
Day	Daniel	1000	EPL	AttackingCharacteristics
Bawerman	Bruce	1000	EPL	AttackingCharacteristics
Eddington	Alan	1000	EPL	AttackingCharacteristics
Becker	Ethan	1000	EPL	AttackingCharacteristics
Chapman	Daniel	1000	EPL	AttackingCharacteristics
Elmers	Devin	1000	EPL	AttackingCharacteristics
Hoggarth	Alfred	1000	EPL	AttackingCharacteristics
Dean	Barbara	1000	EPL	AttackingCharacteristics
Chesterton	Victoria	1000	EPL	AttackingCharacteristics
Dickinson	Emma	1000	EPL	AttackingCharacteristics
Daniels	Cecilia	1000	EPL	AttackingCharacteristics
Galbraith	Andrew	1000	EPL	AttackingCharacteristics
Eddington	Martin	1000	EPL	AttackingCharacteristics

Рис. 2: Результат выполнения запроса 1

Время выполнения запроса – 0.01 сек.

EXPLAIN

id	select_type	table	partitions	type	possible_keys
1	SIMPLE	Football_characteristics	NULL	ALL	PRIMARY
1	SIMPLE	Championship	NULL	ALL	PRIMARY
1	SIMPLE	Rating_list	NULL	ALL	rating_participant_id,season_id
1	SIMPLE	Season	NULL	eq_ref	PRIMARY
1	SIMPLE	Rating_participant	NULL	eq_ref	PRIMARY,footballer_id
1	SIMPLE	Footballer	NULL	eq_ref	PRIMARY
1	SIMPLE	Cost_of_football_characteristic_per_season	NULL	ref	football_characteristics_id,season_id
1	SIMPLE	Cost_of_championship_per_season	NULL	ALL	championship_id,season_id

key	key_len	ref	rows	filtered	Extra
NULL	NULL	NULL	6	16.67	Using where
NULL	NULL	NULL	9	11.11	Using where; Using join buffer (hash join)
NULL	NULL	NULL	11025	33.33	Using where; Using join buffer (hash join)
PRIMARY	4	bd.Rating_list.season_id	1	100.00	Using index
PRIMARY	4	bd.Rating_list.rating_participant_id	1	100.00	NULL
PRIMARY	4	bd.Rating_participant.footballer_id	1	100.00	NULL
season_id	4	bd.Rating_list.season_id	6	16.67	Using where
NULL	NULL	NULL	180	100.00	Using where; Using join buffer (hash join)

Рис. 3: EXPLAIN запроса 1

Объяснение запроса

Для выполнения данного запроса необходимо связать таблицу Footballer с таблицами Rating_list, Championship и Football_characteristics. Сделать это возможно через таблицы Rating_participant, Season, Cost_of_championship_per_season и Cost_of_football_characteristic_per_season. При помощи WHERE в результирующую таблицу выбираются записи, в которых в поле total_price находится значение, большее 990, в поле championship значение 'EPL', а в поле football_characteristics значение 'AttackingCharacteristics'.

3.2 Запрос 2

Формулировка запроса

Посчитать число футболистов у которых присутствует характеристика «Dribbling» и вид

футбола «Male».

SQL-запрос

```
SELECT COUNT(*),
       Football_characteristics.football_characteristics,
       Type_of_football.type_of_football
FROM Footballer
JOIN Rating_participant ON Footballer.footballer_id =
↪  Rating_participant.footballer_id
JOIN Type_of_football ON Type_of_football.type_of_football_id =
↪  Rating_participant.type_of_football_id
JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
JOIN Season ON Rating_list.season_id = Season.season_id
JOIN Cost_of_championship_per_season ON Season.season_id =
↪  Cost_of_championship_per_season.season_id
JOIN Championship ON Cost_of_championship_per_season.championship_id =
↪  Championship.championship_id
JOIN Cost_of_football_characteristic_per_season ON Season.season_id =
↪  Cost_of_football_characteristic_per_season.season_id
JOIN Football_characteristics ON
↪  Cost_of_football_characteristic_per_season.football_characteristics_id =
↪  Football_characteristics.football_characteristics_id
↪  WHERE((Football_characteristics.football_characteristics = 'Dribbling')
        AND (Type_of_football.type_of_football = 'Male'));
```

Результат

На Рис. 4 представлен результат выполнения запроса 2.

COUNT(*)	football_characteristics	type_of_football
49435	Dribbling	Male

Рис. 4: Результат выполнения запроса 2

Время выполнения запроса – 0.13 сек.

EXPLAIN

id	select_type	table	partitions	type	possible_keys	key
1	SIMPLE	Type_of_football	NULL	ALL	PRIMARY	NULL
1	SIMPLE	Football_characteristics	NULL	ALL	PRIMARY	NULL
1	SIMPLE	Rating_list	NULL	ALL	rating_participant_id,season_id	NULL
1	SIMPLE	Season	NULL	eq_ref	PRIMARY	PRIMARY
1	SIMPLE	Rating_participant	NULL	eq_ref	PRIMARY,type_of_football_id,footballer_id	PRIMARY
1	SIMPLE	Cost_of_football_characteristic_per_season	NULL	ref	football_characteristics_id,season_id	season_id
1	SIMPLE	Footballer	NULL	eq_ref	PRIMARY	PRIMARY
1	SIMPLE	Cost_of_championship_per_season	NULL	ALL	championship_id,season_id	NULL
1	SIMPLE	Championship	NULL	eq_ref	PRIMARY	PRIMARY

key_len	ref	rows	filtered	Extra
NULL	NULL	2	50.00	Using where
NULL	NULL	6	16.67	Using where; Using join buffer (hash join)
NULL	NULL	11025	100.00	Using join buffer (hash join)
4	bd.Rating_list.season_id	1	100.00	Using index
4	bd.Rating_list.rating_participant_id	1	50.00	Using where
4	bd.Rating_list.season_id	6	16.67	Using where
4	bd.Rating_participant.footballer_id	1	100.00	Using index
NULL	NULL	180	100.00	Using where; Using join buffer (hash join)
4	bd.Cost_of_championship_per_season.championship_id	1	100.00	Using index

Рис. 5: EXPLAIN запроса 2

Объяснение запроса
Для выполнения данного запроса необходимо связать таблицу Footballer с таблицами Football_characteristics и Type_of_football. Сделать это возможно через таблицы Rating_participant, Rating_list, Season, Cost_of_championship_per_season, Championship, и Cost_of_football_characteristic_per_season. При помощи WHERE в результирующую таблицу выбираются записи, в которых в поле football_characteristics находится значение 'Dribbling', а в поле type_of_football значение 'Male'.

3.3 Запрос 3

Формулировка запроса
Для каждого чемпионата посчитать число участников рейтинга.
SQL-запрос

```
SELECT COUNT(Rating_participant.rating_participant_id) AS ParticipantsCount,
       Championship.championship
FROM Rating_participant
JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
JOIN Season ON Rating_list.season_id = Season.season_id
JOIN Cost_of_championship_per_season ON Season.season_id =
↪  Cost_of_championship_per_season.season_id
JOIN Championship ON Cost_of_championship_per_season.championship_id =
↪  Championship.championship_id
GROUP BY Championship.championship;
```

Результат
На Рис. 6 представлен результат выполнения запроса 3.

ParticipantsCount	championship
10770	SuperLig
9663	LigaNOS
10025	LaLiga
13081	RPL
12411	Eredivisie
9615	BUNDESLIGA
8579	SerieA
14746	LigueOneUberEats
12930	EPL

Рис. 6: Результат выполнения запроса 3

Время выполнения запроса – 0.10 сек.
На Рис.7 представлена диаграмма для запроса 3. По оси X отложены чемпионаты, по оси Y - количество участников рейтинга, которые принимали участие в чемпионатах.

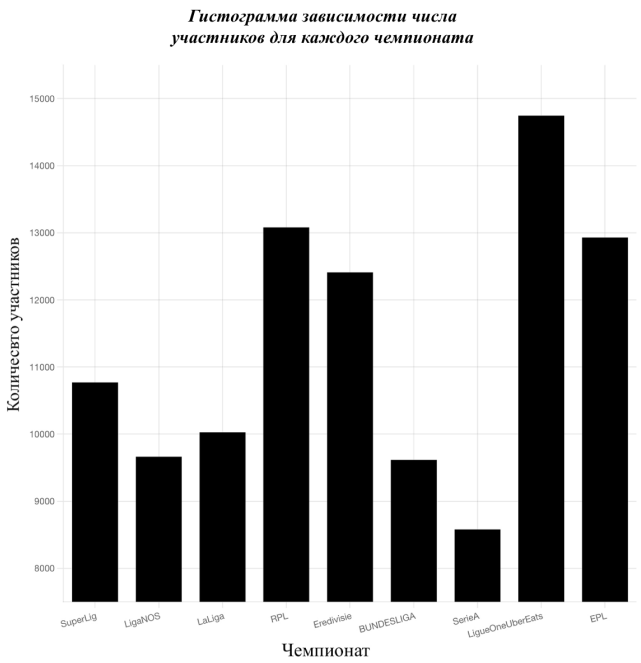


Рис. 7: Гистограмма для запроса 3

EXPLAIN

id	select_type	table	partitions	type	possible_keys	key	key_len
1	SIMPLE	Cost_of_championship_per_season	NULL	ALL	championship_id,season_id	NULL	NULL
1	SIMPLE	Championship	NULL	eq_ref	PRIMARY	PRIMARY	4
1	SIMPLE	Season	NULL	eq_ref	PRIMARY	PRIMARY	4
1	SIMPLE	Rating_list	NULL	ALL	rating_participant_id,season_id	NULL	NULL
1	SIMPLE	Rating_participant	NULL	eq_ref	PRIMARY	PRIMARY	4

ref	rows	filtered	Extra
NULL	180	100.00	Using temporary
bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL
bd.Cost_of_championship_per_season.season_id	1	100.00	Using index
NULL	11025	100.00	Using where; Using join buffer (hash join)
bd.Rating_list.rating_participant_id	1	100.00	Using index

Рис. 8: EXPLAIN запроса 3

Объяснение запроса
Для выполнения данного запроса необходимо связать таблицу Rating_participant с таблицей Championship. Сделать это возможно через таблицы Rating_list, Season, Cost_of_championship_per_season. С помощью GROUP BY записи группируются по полю championship. Функция COUNT считает все получившиеся записи, и получается число участников рейтинга, которых участвовали в конкретном чемпионате - ParticipantCount.

3.4 Запрос 4а

Формулировка запроса
Найти чемпионат с наибольшим числом участников рейтинга.
SQL-запрос

```
SELECT Championship.championship,
       COUNT(Rating_participant.rating_participant_id) AS ParticipantsCount
FROM Rating_participant
```

```
JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
JOIN Season ON Rating_list.season_id = Season.season_id
JOIN Cost_of_championship_per_season ON Season.season_id =
↪  Cost_of_championship_per_season.season_id
JOIN Championship ON Cost_of_championship_per_season.championship_id =
↪  Championship.championship_id
GROUP BY Championship.championship
HAVING ParticipantsCount =
  (SELECT COUNT(Rating_participant.rating_participant_id) AS cnt
   FROM Rating_participant
   JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
   JOIN Season ON Rating_list.season_id = Season.season_id
   JOIN Cost_of_championship_per_season ON Season.season_id =
↪  Cost_of_championship_per_season.season_id
   JOIN Championship ON Cost_of_championship_per_season.championship_id =
↪  Championship.championship_id
   GROUP BY Championship.championship
   ORDER BY cnt DESC
   LIMIT 1);
```

Результат

На Рис. 9 представлен результат выполнения запроса 4а.

championship	ParticipantsCount
LigueOneUberEats	14746

Рис. 9: Результат выполнения запроса 4а

Время выполнения запроса – 0.22 сек.

EXPLAIN

id	select_type	table	partitions	type	possible_keys	key
1	PRIMARY	Season	NULL	index	PRIMARY	PRIMARY
1	PRIMARY	Cost_of_championship_per_season	NULL	ref	championship_id,season_id	season_id
1	PRIMARY	Championship	NULL	eq_ref	PRIMARY	PRIMARY
1	PRIMARY	Rating_list	NULL	ref	rating_participant_id,season_id	season_id
1	PRIMARY	Rating_participant	NULL	eq_ref	PRIMARY	PRIMARY
2	SUBQUERY	Season	NULL	index	PRIMARY	PRIMARY
2	SUBQUERY	Cost_of_championship_per_season	NULL	ref	championship_id,season_id	season_id
2	SUBQUERY	Championship	NULL	eq_ref	PRIMARY	PRIMARY
2	SUBQUERY	Rating_list	NULL	ref	rating_participant_id,season_id	season_id
2	SUBQUERY	Rating_participant	NULL	eq_ref	PRIMARY	PRIMARY

key_len	ref	rows	filtered	Extra
4	NULL	20	100.00	Using index; Using temporary
4	bd.Season.season_id	9	100.00	NULL
4	bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL
4	bd.Season.season_id	551	100.00	NULL
4	bd.Rating_list.rating_participant_id	1	100.00	Using index
4	NULL	20	100.00	Using index; Using temporary; Using filesort
4	bd.Season.season_id	9	100.00	NULL
4	bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL
4	bd.Season.season_id	551	100.00	NULL
4	bd.Rating_list.rating_participant_id	1	100.00	Using index

Рис. 10: EXPLAIN запроса 4а

Объяснение запроса

Для выполнения данного запроса необходимо связать таблицу Championship с таблицей

Rating_participant. Сделать это возможно через таблицы Rating_list, Season, Cost_of_championship_per_season. С помощью GROUP BY записи группируются по полю championship. Это позволяет подсчитать участников для каждого чемпионата отдельно. Благодаря HAVING происходит фильтрация результатов сгруппированной выборки. Функция выбирает только те записи, для которых количество участников (ParticipantsCount) равно максимальному количеству участников в каком-либо чемпионате. Внутренний подзапрос сначала считает количество участников для каждого чемпионата, затем сортирует результаты по убыванию (ORDER BY cnt DESC) и выбирает только первую запись (LIMIT 1).

3.5 Запрос 4б

Формулировка запроса
Найти чемпионат с наименьшим числом участников рейтинга.
SQL-запрос

```
SELECT Championship.championship,
       COUNT(Rating_participant.rating_participant_id) AS ParticipantsCount
FROM Championship
JOIN Cost_of_championship_per_season ON Championship.championship_id =
↪  Cost_of_championship_per_season.championship_id
JOIN Season ON Cost_of_championship_per_season.season_id = Season.season_id
JOIN Rating_list ON Season.season_id = Rating_list.season_id
LEFT JOIN Rating_participant ON Rating_list.rating_participant_id =
↪  Rating_participant.rating_participant_id
GROUP BY Championship.championship
HAVING ParticipantsCount =
  (SELECT COUNT(Rating_participant.rating_participant_id) AS cnt
   FROM Championship
   JOIN Cost_of_championship_per_season ON Championship.championship_id =
↪   Cost_of_championship_per_season.championship_id
   JOIN Season ON Cost_of_championship_per_season.season_id = Season.season_id
   JOIN Rating_list ON Season.season_id = Rating_list.season_id
   LEFT JOIN Rating_participant ON Rating_list.rating_participant_id =
↪   Rating_participant.rating_participant_id
   GROUP BY Championship.championship
   ORDER BY cnt ASC
   LIMIT 1);
```

Результат
На Рис. 11 представлен результат выполнения запроса 4б.

championship	ParticipantsCount
SerieA	8579

Рис. 11: Результат выполнения запроса 4б

Время выполнения запроса – 0.21 сек.
EXPLAIN

id	select_type	table	partitions	type	possible_keys	key	key_len
1	PRIMARY	Season	NULL	index	PRIMARY	PRIMARY	4
1	PRIMARY	Cost_of_championship_per_season	NULL	ref	championship_id,season_id	season_id	4
1	PRIMARY	Championship	NULL	eq_ref	PRIMARY	PRIMARY	4
1	PRIMARY	Rating_list	NULL	ref	season_id	season_id	4
1	PRIMARY	Rating_participant	NULL	eq_ref	PRIMARY	PRIMARY	4
2	SUBQUERY	Season	NULL	index	PRIMARY	PRIMARY	4
2	SUBQUERY	Cost_of_championship_per_season	NULL	ref	championship_id,season_id	season_id	4
2	SUBQUERY	Championship	NULL	eq_ref	PRIMARY	PRIMARY	4
2	SUBQUERY	Rating_list	NULL	ref	season_id	season_id	4
2	SUBQUERY	Rating_participant	NULL	eq_ref	PRIMARY	PRIMARY	4

ref	rows	filtered	Extra
NULL	20	100.00	Using index; Using temporary
bd.Season.season_id	9	100.00	NULL
bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL
bd.Season.season_id	551	100.00	NULL
bd.Rating_list.rating_participant_id	1	100.00	Using index
NULL	20	100.00	Using index; Using temporary; Using filesort
bd.Season.season_id	9	100.00	NULL
bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL
bd.Season.season_id	551	100.00	NULL
bd.Rating_list.rating_participant_id	1	100.00	Using index

Рис. 12: EXPLAIN запроса 4б

Объяснение запроса

Для выполнения данного запроса необходимо связать таблицу Championship с таблицей Rating_participant. Сделать это возможно через таблицы Rating_list, Season, Cost_of_championship_per_season. С помощью GROUP BY записи группируются по полю championship. Это позволяет подсчитать участников для каждого чемпионата отдельно. Благодаря HAVING происходит фильтрация результатов сгруппированной выборки. Функция выбирает только те записи, для которых количество участников (ParticipantsCount) равно максимальному количеству участников в каком-либо чемпионате. Внутренний подзапрос сначала считает количество участников для каждого чемпионата, затем сортирует результаты по возрастанию (ORDER BY cnt ASC) и выбирает только первую запись (LIMIT 1).

3.6 Запрос 5

Формулировка запроса

Посчитать число участников с одинаковым суммарной ценой в рейтинге.

SQL-запрос

```
SELECT Rating_list.total_price,
       COUNT(*) AS ParticipantsCount
FROM Rating_participant
JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
GROUP BY Rating_list.total_price
HAVING COUNT(*) > 1
ORDER BY Rating_list.total_price ASC;
```

Результат

На Рис. 13 представлены 30 из 85 строк результата выполнения запроса 5.

total_price	ParticipantsCount
160	127
170	122
180	123
190	126
200	132
210	142
220	120
230	134
240	108
250	135
260	144
270	155
280	130
290	110
300	137
310	131
320	126
330	134
340	126
350	143
360	138
370	116
380	146
390	151
400	151
410	124
420	167
430	117
440	131
450	133

Рис. 13: Результат выполнения запроса 5

Время выполнения запроса – 0.03 сек.
На Рис.14 представлена диаграмма для запроса 5. По оси X отложены суммарные цены участников, по оси Y - количество участников рейтинга, которые имеют суммырную цену.

Рис. 14: Гистограмма для запроса 5

EXPLAIN

id	select_type	table	partitions	type	possible_keys	key	key_len
1	SIMPLE	Cost_of_championship_per_season	NULL	ALL	championship_id,season_id	NULL	NULL
1	SIMPLE	Championship	NULL	eq_ref	PRIMARY	PRIMARY	4
1	SIMPLE	Season	NULL	eq_ref	PRIMARY	PRIMARY	4
1	SIMPLE	Rating_list	NULL	ALL	rating_participant_id,season_id	NULL	NULL
1	SIMPLE	Rating_participant	NULL	eq_ref	PRIMARY	PRIMARY	4

ref	rows	filtered	Extra
NULL	180	100.00	Using temporary
bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL
bd.Cost_of_championship_per_season.season_id	1	100.00	Using index
NULL	11025	100.00	Using where; Using join buffer (hash join)
bd.Rating_list.rating_participant_id	1	100.00	Using index

Рис. 15: EXPLAIN запроса 3

Объяснение запроса

Для выполнения данного запроса необходимо связать таблицу Rating_list с таблицей Rating_participant. С помощью GROUP BY записи группируются по полю total_price. Функция HAVING создаёт условие фильтрации, которое выбирает только те группы, в которых количество записей больше 1. То есть, это только те поля total_price, которые встречаются более одного раза. С помощью ORDER BY результаты сортируются в порядке возрастания (ASC) по полю total_price.

3.7 Запрос 6

Формулировка запроса

Найти футбольную характеристику, число футболистов которой больше, чем в характеристике RunningSpeed.

SQL-запрос

```
SELECT fc.football_characteristics AS Characteristic, COUNT(*) AS
↪  Number_of_Footballers
FROM Football_characteristics fc
JOIN Cost_of_football_characteristic_per_season cost
ON fc.football_characteristics_id = cost.football_characteristics_id
WHERE fc.football_characteristics <> 'RunningSpeed'
GROUP BY fc.football_characteristics
HAVING Number_of_Footballers > (
    SELECT COUNT(*)
    FROM Football_characteristics fc_running_speed
    JOIN Cost_of_football_characteristic_per_season cost_running_speed
    ON fc_running_speed.football_characteristics_id =
↪  cost_running_speed.football_characteristics_id
    WHERE fc_running_speed.football_characteristics = 'RunningSpeed'
);
```

Результат

На Рис. 16 представлен результат выполнения запроса 6.

Characteristic	Number_of_Footballers
AttackingCharacteristics	18
Dribbling	20
PhysicalTraining	26
TransmissionAccuracy	26

Рис. 16: Результат выполнения запроса 6

Время выполнения запроса – 0.01 сек.

EXPLAIN

id	select_type	table	partitions	type	possible_keys	key
1	PRIMARY	fc	NULL	ALL	PRIMARY	NULL
1	PRIMARY	cost	NULL	ref	football_characteristics_id	football_characteristics_id
2	SUBQUERY	fc_running_speed	NULL	ALL	PRIMARY	NULL
2	SUBQUERY	cost_running_speed	NULL	ref	football_characteristics_id	football_characteristics_id

key_len	ref	rows	filtered	Extra
NULL	NULL	6	83.33	Using where; Using temporary
4	bd.fc.football_characteristics_id	20	100.00	Using index
NULL	NULL	6	16.67	Using where
4	bd.fc_running_speed.football_characteristics_id	20	100.00	Using index

Рис. 17: EXPLAIN запроса 6

Объяснение запроса

Для выполнения данного запроса необходимо связать таблицу Football_characteristics с таблицей Cost_of_football_charactristics_per_season. С помощью WHERE реализовано условие фильтрации, которое исключает из результата записи, где характеристика футболиста равна «RunningSpeed». Результаты выборки группируются при помощи GROUP BY по значениям в поле football_characteristics Функция HAVING создаёт подзапрос, в котором реализовано ещё одно условие фильтрации, которое выбирает только те группы характеристик, у которых количество футболистов больше, чем количество футболистов с характеристикой «RunningSpeed».

3.8 Запрос 7

Формулировка запроса

Вывести фамилии и имена всех футболистов, которые имеют позицию на поле «Midfielder» и у которых есть футбольная характеристика «PhysicalTraining», цена которой - 30.

SQL-запрос

```
SELECT Footballer.surname,
       Footballer.name,
       Footballer.field_position,
       Football_characteristics.football_characteristics,
       Price.price
FROM Footballer
JOIN Rating_participant ON Footballer.footballer_id =
↪  Rating_participant.footballer_id
JOIN Rating_list ON Rating_participant.rating_participant_id =
↪  Rating_list.rating_participant_id
JOIN Season ON Rating_list.season_id = Season.season_id
JOIN Cost_of_football_characteristic_per_season ON Season.season_id =
↪  Cost_of_football_characteristic_per_season.season_id
JOIN Football_characteristics ON
↪  Cost_of_football_characteristic_per_season.football_characteristics_id =
↪  Football_characteristics.football_characteristics_id
JOIN Price ON Cost_of_football_characteristic_per_season.price_id = Price.price_id
WHERE (Footballer.field_position = 'Midfielder'
       AND Football_characteristics.football_characteristics = 'PhysicalTraining'
       AND Price.price = 50);
```

Результат

На Рис. 18 представлены 23 из 185 строк результата выполнения запроса 7.

surname	name	field_position	football_characteristics	price
Carey	Timothy	Midfielder	PhysicalTraining	50
Allford	Christian	Midfielder	PhysicalTraining	50
Birch	Brandon	Midfielder	PhysicalTraining	50
Birch	Brandon	Midfielder	PhysicalTraining	50
Dean	Martin	Midfielder	PhysicalTraining	50
Derrick	Diego	Midfielder	PhysicalTraining	50
Backer	Bruce	Midfielder	PhysicalTraining	50
Bawerman	Henry	Midfielder	PhysicalTraining	50
Anderson	Dennis	Midfielder	PhysicalTraining	50
Anderson	Dennis	Midfielder	PhysicalTraining	50
Alsopp	Cameron	Midfielder	PhysicalTraining	50
Elmers	Edward	Midfielder	PhysicalTraining	50
Elmers	Hugh	Midfielder	PhysicalTraining	50
Elmers	Carlos	Midfielder	PhysicalTraining	50
Gerald	Francis	Midfielder	PhysicalTraining	50
Hoggarth	Alan	Midfielder	PhysicalTraining	50
Becker	Ethan	Midfielder	PhysicalTraining	50
Carter	Bernard	Midfielder	PhysicalTraining	50
Carter	Bernard	Midfielder	PhysicalTraining	50
Addington	Ethan	Midfielder	PhysicalTraining	50
Holmes	Christopher	Midfielder	PhysicalTraining	50
Daniels	Ethan	Midfielder	PhysicalTraining	50
Calhoun	Anthony	Midfielder	PhysicalTraining	50
Gilbert	Adam	Midfielder	PhysicalTraining	50
Leman	Benjamin	Midfielder	PhysicalTraining	50

Рис. 18: Результат выполнения запроса 7

Время выполнения запроса – 0.01 сек.
EXPLAIN

id	select_type	table	partitions	type	possible_keys
1	SIMPLE	Football_characteristics	NULL	ALL	PRIMARY
1	SIMPLE	Price	NULL	ALL	PRIMARY
1	SIMPLE	Season	NULL	index	PRIMARY
1	SIMPLE	Cost_of_football_characteristic_per_season	NULL	ref	price_id,football_characteristics_id,season_id
1	SIMPLE	Rating_list	NULL	ref	rating_participant_id,season_id
1	SIMPLE	Rating_participant	NULL	eq_ref	PRIMARY,footballer_id
1	SIMPLE	Footballer	NULL	eq_ref	PRIMARY

key	key_len	ref	rows	filtered	Extra
NULL	NULL	NULL	6	16.67	Using where
NULL	NULL	NULL	20	10.00	Using where; Using join buffer (hash join)
PRIMARY	4	NULL	20	100.00	Using index; Using join buffer (hash join)
price_id	4	bd.Price.price_id	6	0.83	Using where
season_id	4	bd.Season.season_id	551	100.00	NULL
PRIMARY	4	bd.Rating_list.rating_participant_id	1	100.00	NULL
PRIMARY	4	bd.Rating_participant.footballer_id	1	33.33	Using where

Рис. 19: EXPLAIN запроса 7

Объяснение запроса
Для выполнения данного запроса необходимо связать таблицу Footballer с таблицами Football_characteristics и Price. Сделать это возможно через таблицы Rating_participant, Rating_list, Season, Cost_of_football_characteristics_per_season. При помощи WHERE в результирующую таблицу выбираются записи, в которых в поле field_position значение 'Midfielder', в поле championship значение 'EPL', в поле football_characteristics значение 'PhysicalTraining' и в поле prcie значение 50.

3.9 Запрос 8

Формулировка запроса
Для каждого вида футбола и чемпионата посчитать число участников.
SQL-запрос

```
SELECT Type_of_football.type_of_football_id,
       Championship.championship_id,

  (SELECT COUNT(*)
   FROM Rating_list rl
   JOIN Rating_participant rp ON rp.rating_participant_id = rl.rating_participant_id
   AND Type_of_football.type_of_football_id=rp.type_of_football_id
   JOIN Season s ON s.season_id = rl.season_id
   JOIN Cost_of_championship_per_season co ON co.season_id = s.season_id
   AND co.championship_id=Championship.championship_id) AS amount
FROM Championship
JOIN Type_of_football;
```

Результат

На Рис. 20 представлены результат выполнения запроса 8.

type_of_football_id	championship_id	amount
2	1	6601
1	1	6719
2	2	5837
1	2	5951
2	3	4454
1	3	4696
2	4	6711
1	4	6769
2	5	5053
1	5	5077
2	6	4128
1	6	4325
2	7	6089
1	7	6323
2	8	5299
1	8	5503
2	9	5869
1	9	5875
2	10	0
1	10	0

Рис. 20: Результат выполнения запроса 8

Время выполнения запроса – 0.19 сек.

На Рис.21 представлена диаграмма для запроса 8. По оси X отложены чемпионаты, по оси Y - вид футбола, по оси z - количество участников.

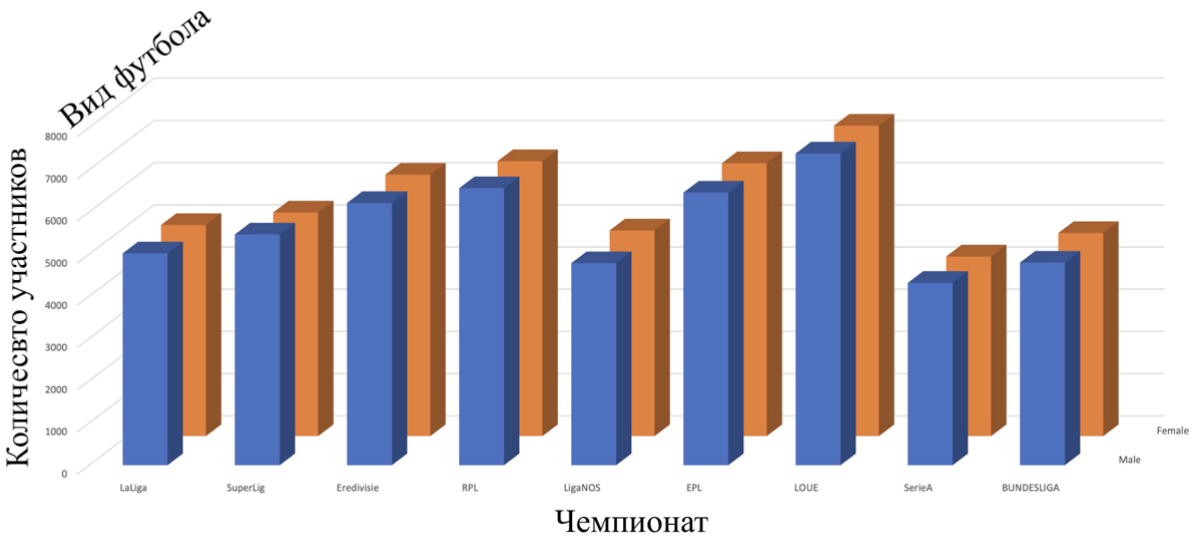


Рис. 21: 3D-диаграмма для запроса 8

EXPLAIN

id	select_type	table	partitions	type	possible_keys
1	SIMPLE	Season	NULL	index	PRIMARY
1	SIMPLE	Rating_list	NULL	ref	rating_participant_id,season_id
1	SIMPLE	Rating_participant	NULL	eq_ref	PRIMARY,type_of_football_id
1	SIMPLE	Type_of_football	NULL	ALL	PRIMARY
1	SIMPLE	Cost_of_championship_per_season	NULL	ref	championship_id,season_id
1	SIMPLE	Championship	NULL	eq_ref	PRIMARY

key	key_len	ref	rows	filtered	Extra
PRIMARY	4	NULL	20	100.00	Using index; Using temporary
season_id	4	bd.Season.season_id	551	100.00	NULL
PRIMARY	4	bd.Rating_list.rating_participant_id	1	100.00	NULL
NULL	NULL	NULL	2	50.00	Using where; Using join buffer (hash join)
season_id	4	bd.Season.season_id	9	100.00	NULL
PRIMARY	4	bd.Cost_of_championship_per_season.championship_id	1	100.00	NULL

Рис. 22: EXPLAIN запроса 8

Объяснение запроса

Для выполнения данного запроса необходимо связать таблицу Rating_participant с таблицами Type_of_football и Championship. Сделать это возможно через таблицы Rating_list, Season, Cost_of_championship_per_season. Записи группируются благодаря GROUP BY по полям type_of_football и championship.

Вывод

В ходе выполнения курсовой работы была выбрана и проанализирована предметная область для создания базы данных - «Формирование мирового рейтинга футболистов в ценовой характеристике». Было выделено 14 сущностей и 26 атрибута. На их основе построена ER-диаграмма, которая содержит три типа связей: тернарная, бинарная и одинарная. На основе выделенных сущностей была построена схема объектов, которая содержит 12 объектов, связей 14 «один-ко-многим». Далее была реализована схема базы данных, которая состоит из 12 таблиц, 7 из которых словари. Связи между таблицами были реализованы с помощью внешних ключей.

Был изучен язык SQL и написан скрипт для создания базы данных. На языке Java написана программа, которая генерирует sql-команды для заполнения базы данных случайно сгенерированными данными. В результате база данных содержит записей. Была написана программа на языке Java, содержащая 326 строк, заполняющая базу случайно сгенерированными записями. База данных была заполнена 41225 записями.

Было составлено и реализовано 8 запросов к базе данных на языке SQL. Все запросы проанализированы с помощью команды EXPLAIN.

Приложение А. Создание базы данных

Создание базы данных

```
mysql -uroot -p

DROP DATABASE IF EXISTS BD;
CREATE DATABASE BD;
USE BD;

CREATE TABLE Country (
  country_id INT NOT NULL AUTO_INCREMENT,
  country VARCHAR(58) NOT NULL,

  PRIMARY KEY(country_id)
);

CREATE TABLE Type_of_football (
  type_of_football_id INT NOT NULL AUTO_INCREMENT,
  type_of_football ENUM('Male', 'Female') NOT NULL,

  PRIMARY KEY(type_of_football_id)
);

CREATE TABLE Championship (
  championship_id INT NOT NULL AUTO_INCREMENT,
  championship ENUM('EPL', 'BUNDESLIGA', 'SerieA', 'LigueOneUberEats', 'LaLiga',
↪  'LigaNOS', 'Eredivisie', 'RPL', 'SuperLig') NOT NULL,

  PRIMARY KEY(championship_id)
);

CREATE TABLE Football_characteristics (
  football_characteristics_id INT NOT NULL AUTO_INCREMENT,
  football_characteristics ENUM('AttackingCharacteristics' , 'RunningSpeed',
↪  'Dribbling', 'PhysicalTraining',  'TransmissionAccuracy',
↪  'DefensiveCharacteristic'),

  PRIMARY KEY (football_characteristics_id)
);

CREATE TABLE Season (
season_id INT NOT NULL AUTO_INCREMENT,
season INT NOT NULL,

PRIMARY KEY(season_id)
);

CREATE TABLE Price (
price_id INT NOT NULL AUTO_INCREMENT,
price INT NOT NULL,

PRIMARY KEY(price_id)
```

```
);

CREATE TABLE Footballer (
  footballer_id INT NOT NULL AUTO_INCREMENT,
  surname VARCHAR(50) NOT NULL,
  name VARCHAR(50) NOT NULL,
  gender ENUM('Man', 'Woman' ) NOT NULL,
  age INT NOT NULL,
  citizenship VARCHAR(58) NOT NULL,
  coachName VARCHAR(50) NOT NULL,
  coachSurname VARCHAR(50) NOT NULL,
  field_position ENUM('Midfielder', 'Defender', 'Forward') NOT NULL,

  PRIMARY KEY(footballer_id)
);

CREATE TABLE Rating_participant (
rating_participant_id INT NOT NULL AUTO_INCREMENT,
type_of_football_id INT NOT NULL,
country_id INT NOT NULL,
footballer_id INT NOT NULL,
status ENUM('active','finished') NOT NULL,

PRIMARY KEY(rating_participant_id),
INDEX (type_of_football_id),
INDEX (country_id),
INDEX (footballer_id),

FOREIGN KEY (type_of_football_id)
    REFERENCES Type_of_football(type_of_football_id)
    ON UPDATE CASCADE,

 FOREIGN KEY (country_id)
    REFERENCES Country(country_id)
    ON UPDATE CASCADE,

  FOREIGN KEY (footballer_id)
    REFERENCES Footballer(footballer_id)
    ON UPDATE CASCADE
);

CREATE TABLE Cost_of_football_characteristic_per_season (
cost_of_football_characteristic_per_season_id INT NOT NULL AUTO_INCREMENT,
price_id INT NOT NULL,
football_characteristics_id INT NOT NULL,
season_id INT NOT NULL,

PRIMARY KEY(cost_of_football_characteristic_per_season_id),
INDEX (price_id),
INDEX (football_characteristics_id),
INDEX (season_id),

FOREIGN KEY (price_id)
```

```
    REFERENCES Price(price_id)
    ON UPDATE CASCADE,

 FOREIGN KEY (football_characteristics_id)
    REFERENCES Football_characteristics(football_characteristics_id)
    ON UPDATE CASCADE,

  FOREIGN KEY (season_id)
    REFERENCES Season(season_id)
    ON UPDATE CASCADE
);

CREATE TABLE Cost_of_championship_per_season (
cost_of_championship_per_season_id INT NOT NULL AUTO_INCREMENT,
price_id INT NOT NULL,
championship_id INT NOT NULL,
season_id INT NOT NULL,

PRIMARY KEY(cost_of_championship_per_season_id),
INDEX (price_id),
INDEX (championship_id),
INDEX (season_id),

FOREIGN KEY (price_id)
    REFERENCES Price(price_id)
    ON UPDATE CASCADE,

 FOREIGN KEY (championship_id)
    REFERENCES Championship(championship_id)
    ON UPDATE CASCADE,

  FOREIGN KEY (season_id)
    REFERENCES Season(season_id)
    ON UPDATE CASCADE
);

CREATE TABLE Cost_of_participant_per_season (
cost_of_participant_per_season_id INT NOT NULL AUTO_INCREMENT,
cost_of_championship_per_season_id INT,
cost_of_football_characteristic_per_season_id INT,
rating_participant_id INT NOT NULL,
season_id INT NOT NULL,

PRIMARY KEY(cost_of_participant_per_season_id),
INDEX (cost_of_championship_per_season_id),
INDEX (cost_of_football_characteristic_per_season_id),
INDEX (rating_participant_id),
INDEX (season_id),

FOREIGN KEY (cost_of_championship_per_season_id)
    REFERENCES Cost_of_championship_per_season(cost_of_championship_per_season_id)
    ON UPDATE CASCADE,
```

```
 FOREIGN KEY (cost_of_football_characteristic_per_season_id)
    REFERENCES Cost_of_football_characteristic_per_season(cost_of_football_character
↪   istic_per_season_id)
    ON UPDATE CASCADE,

FOREIGN KEY (rating_participant_id)
    REFERENCES Rating_participant(rating_participant_id)
    ON UPDATE CASCADE,

  FOREIGN KEY (season_id)
    REFERENCES Season(season_id)
    ON UPDATE CASCADE
);

CREATE TABLE Rating_list (
  rating_list_id INT NOT NULL AUTO_INCREMENT,
  rating_participant_id INT NOT NULL,
  total_price INT NOT NULL,
  season_id INT NOT NULL,

  PRIMARY KEY(rating_list_id),
  INDEX (Rating_participant_id),
  INDEX (season_id),

  FOREIGN KEY (Rating_participant_id)
    REFERENCES Rating_participant(Rating_participant_id)
    ON UPDATE CASCADE,

  FOREIGN KEY (season_id)
    REFERENCES Season(season_id)
    ON UPDATE CASCADE
);-
```

Приложение В. Код генерации данных

Генерация данных
Файл dataGenerator.java

```
static int sumId = 1;
    public static String[] firstNameFemale = {"Abigail", "Adelina", "Agatha",
    ↪  "Alexa", "Alexis", "Alise", "Amanda",
            "Amber", "Amelia", "Angelina", "Barbara", "Bridjet", "Caroline",
            ↪  "Catherine", "Cecilia", "Daisy",
            "Danielle", "Eleanor", "Elizabeth", "Ella", "Emily", "Emma",
            "Fiona", "Florence", "Freda", "Hailey", "Haley", "Hannah", "Helen",
            ↪  "Katelyn", "Katherine", "Kathryn",
            "Kayla", "Leonora", "Leslie", "Lillian", "Lily", "Linda", "Lorna",
            ↪  "Luccile", "Rachel", "Rebecca", "Riley",
            "Rita", "Rose", "Samantha", "Sandra", "Sara", "Savannah", "Sheila",
            ↪  "Sierra", "Sofia", "Taylor", "Vanessa",
            "Victoria", "Violet", "Zoe", "Mia", "Michelle", "Molly"};
    public static String[] firstNameMale = {"Aaron", "Adam", "Adrian", "Aidan",
    ↪  "Alan", "Albert", "Alejandro",
            "Alex", "Alfred", "Andrew", "Anthony", "Antonio", "Austin",
            ↪  "Benjamin", "Bernard", "Blake", "Brandon",
            "Bruce", "Bryan", "Cameron", "Carl", "Carlos", "Charles",
            ↪  "Christopher", "Connor", "Caleb", "Christian",
            "Clifford", "Colin", "Daniel", "David", "Dennis", "Devin", "Diego",
            ↪  "Dominic", "Donald", "Douglas",
            "Dylan", "Edward", "Eric", "Ethan", "Evan", "Francis", "Fred",
            ↪  "Harold", "Harry", "Hayden", "Henry",
            "Herbert", "Horace", "Howard", "Hugh", "Hunter", "Martin", "Mason",
            ↪  "Matthew", "Michael", "Thomas",
            "Timothy", "Tyler"};
    public static String[] secondName = {"Abramson", "Adamson", "Adderiy",
    ↪  "Addington", "Albertson", "Aldridge",
            "Allford", "Alsopp", "Anderson", "Backer", "Baldwin", "Bargeman",
            ↪  "Barnes", "Barrington", "Bawerman",
            "Becker", "Benson", "Berrington", "Birch", "Bishop", "Calhoun",
            ↪  "Campbell", "Carey", "Carrington",
            "Carroll", "Carter", "Chandter", "Chapman", "Charlson", "Chesterton",
            ↪  "Daniels", "Davidson", "Day",
            "Dean", "Derrick", "Dickinson", "Dodson", "Donaldson", "Eddington",
            ↪  "Edwards", "Ellington", "Elmers",
            "Galbraith", "Gardner", "Garrison", "Gate", "Gerald", "Gibbs",
            ↪  "Gilbert", "Higgins", "Hodges",
            "Hoggarth", "Holiday", "Holmes", "Howard", "Leman", "Lewin", "Little",
            ↪  "Livingston", "Longman"};
    public static String[] countiesEng = {"Abkhazia", "Australia", "Austria",
    ↪  "Azerbaijan", "Albania", "Algeria",
            "Angola", "Andorra", "Argentina", "Argentina", "Armenia",
            ↪  "Afghanistan", "Afghanistan", "Bangladesh",
            "Barbados", "Bahrain", "Belize", "Belorussia", "Belgium", "Benin",
            ↪  "Bolivia", "Botswana", "Brazil",
            "Bulgaria", "Burundi", "Bhutan", "Vanuatu", "Vatican", "City",
            ↪  "United", "Kingdom", "Hungary",
```

```
        "Venezuela", "Vietnam", "Gabon", "Haiti", "Guyana", "Gambia",
        ↪  "Cambodia", "Cameroon", "Canada", "Qatar",
        "Kenya", "Cyprus", "Kyrgyzstan", "Kiribati", "China", "DPRK",
        ↪  "Colombia", "Costa", "Rica", "Egypt",
        "Ethiopia", "Ghana", "Guatemala", "Guinea", "Germany", "Grenada",
        ↪  "Greece", "Georgia", "Denmark",
        "Djibouti", "Dominica", "Congo", "Egypt", "Zambia", "Zimbabwe",
        ↪  "Israel", "India", "Indonesia", "Iran",
        "Iraq", "Ireland", "Spain", "Iceland", "Italy", "Yemen", "Kazakhstan",
        ↪  "Cambodia", "Cameroon", "Rica",
        "Cuba", "Kuwait", "Laos", "Latvia", "Lesotho", "Liberia", "Lebanon",
        ↪  "Libya", "Lithuania", "Liechtenstein",
        "Luxembourg", "Madagascar", "Malawi", "Malaysia", "Mali", "Malta",
        ↪  "Mauritania", "Mauritius", "Mexico",
        "Morocco", "Mozambique", "Moldova", "Monaco", "Mongolia", "Myanmar",
        ↪  "Namibia", "Nauru", "Nepal",
        "Netherlands", "New", "Zealand", "Nicaragua", "Niger", "Nigeria",
        ↪  "Norway", "UAE", "Oman", "Pakistan",
        "Palau", "Panama", "Papua", "Paraguay", "Peru", "Poland", "Portugal",
        ↪  "Russia", "Rwanda", "Romania",
        "Salvador", "Samoa", "Senegal", "Serbia", "Singapore", "Syria",
        ↪  "Slovakia", "Slovenia", "Solomon", "Islands",
        "Somalia", "Sudan", "Suriname", "USA", "Sierra", "Leone",
        ↪  "Tajikistan", "Thailand", "Tanzania", "Togo",
        "Tonga", "Tuvalu", "Tunisia", "Turkey", "Turkmenistan", "Uganda",
        ↪  "Uzbekistan", "Ukraine", "Uruguay", "Fiji",
        "Philippines", "Finland", "France", "Croatia", "CAR", "Chad",
        ↪  "Montenegro", "Czech", "Republic", "Chile",
        "Sweden", "Switzerland", "Sri", "Lanka", "Ecuador", "Eritrea",
        ↪  "Eswatini", "Estonia", "Ethiopia", "South",
        "Africa", "Jamaica", "Japan"};
public static String[] fieldPosition = {"Midfielder", "Defender", "Forward"};

public static int[] totalPrice = {160, 170, 180, 190, 200, 210, 220, 230, 240,
        250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370,
        ↪  380,390, 400,
410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560,
↪  570, 580, 590, 600,
610, 620, 630, 640, 650, 660, 670, 680, 690, 700, 710, 720, 730, 740, 750, 760,
↪  770, 780, 790, 800,
810, 820, 830, 840, 850, 860, 870, 880, 890, 900, 910, 920, 930, 940, 950, 960,
↪  970, 980, 990, 1000};

public static void fillCountry(BufferedWriter writer) throws IOException {//ок
    int id = 1;
    writer.write("DELETE FROM Country;");
    writer.newLine();

    for (int i = 0; i < 182; i++) {
        writer.write("INSERT INTO Country VALUES (" + id + "," + "'" +
        ↪  countiesEng[i] + "'" + ");");
        id++;
        writer.newLine();
```

```
    }
}

public static void fillType_of_football(BufferedWriter writer) throws
↪  IOException {//Ок
    String[] type_of_football = {"Male", "Female"};
    writer.write("DELETE FROM Type_of_football;");
    writer.newLine();
    for (int i = 1; i <= 2; i++) {
        writer.write("INSERT INTO Type_of_football VALUES (" + i + "," + "'" +
        ↪  type_of_football[i - 1] + "'" + ");");
        writer.newLine();
    }
}

public static void fillChampionship(BufferedWriter writer) throws IOException
↪  {//Ок
    String[] ligaType = {"EPL", "BUNDESLIGA", "SerieA", "LigueOneUberEats",
    ↪  "LaLiga", "LigaNOS", "Eredivisie", "RPL", "SuperLig"};
    writer.write("DELETE FROM Championship;");
    writer.newLine();
    for (int i = 1; i <= 9; i++) {
        writer.write("INSERT INTO Championship VALUES (" + i + "," + "'" +
        ↪  ligaType[i - 1] + "'" + ");");
        writer.newLine();
    }
}
public static void fillFootball_characteristics(BufferedWriter writer) throws
↪  IOException {//Ок
    String[] fkType = {"AttackingCharacteristics", "RunningSpeed", "Dribbling",
    ↪  "PhysicalTraining", "TransmissionAccuracy", "DefensiveCharacteristic"};
    writer.write("DELETE FROM Football_characteristics;");
    writer.newLine();
    for (int i = 1; i <= 6; i++) {
        writer.write("INSERT INTO Football_characteristics VALUES (" + i + ","
        ↪  + "'" + fkType[i - 1] + "'" + ");");
        writer.newLine();
    }
}

public static void fillSeason(BufferedWriter writer) throws IOException {//Ок
    LocalDate currentDate = LocalDate.now();
    int currentYear = currentDate.getYear();
    writer.write("DELETE FROM Season;");
    writer.newLine();
    for (int i = 1; i <= 20; i++) {
        currentYear--;
        writer.write("INSERT INTO Season VALUES (" + i + "," + currentYear +
        ↪  ");");
        writer.newLine();
    }
}
```

```
public static void fillPrice(BufferedWriter writer) throws IOException {//Ок
    writer.write("DELETE FROM Price;");
    writer.newLine();
    int id = 10;
    for (int i = 1; i <= 20; i++) {
        writer.write("INSERT INTO Price VALUES (" + i + "," + id + ");");
        writer.newLine();
        id+=10;
    }
}

public static void fillFootballer(BufferedWriter writer) throws IOException {//Ок
    writer.write("DELETE FROM Footballer;");
    writer.newLine();
    Random rnd;
    Date date;
    long ms;
    Calendar cal = Calendar.getInstance();

    for (int i = 1; i <= 750; i++) {
        rnd = new Random();
        ms = -946771200000L + (Math.abs(rnd.nextLong()) % (70L * 365 * 24 * 60 *
        ↪  60 * 1000));
        date = new Date(ms);
        cal.setTime(date);
        writer.write("INSERT INTO Footballer VALUES (" + i +  ",'" +
                secondName[getRandomInt(secondName.length)] + "','"
                + firstNameMale[getRandomInt(firstNameMale.length)]
                + "'," + "'Man'" + "," + date + ",'"
                + countiesEng[getRandomInt(countiesEng.length)] + "','" +
                ↪  secondName[getRandomInt(secondName.length)] + "','"
                + firstNameMale[getRandomInt(firstNameMale.length)]
                + "','" + fieldPosition[getRandomInt(fieldPosition.length)] +
                ↪  "');");
        writer.newLine();
    }
    for (int i = 751; i <= 1500; i++) {
        rnd = new Random();
        ms = -946771200000L + (Math.abs(rnd.nextLong()) % (70L * 365 * 24 * 60 *
        ↪  60 * 1000));
        date = new Date(ms);
        cal.setTime(date);
        writer.write("INSERT INTO Footballer VALUES (" + i  + ",'" +
                secondName[getRandomInt(secondName.length)] + "','"
                + firstNameFemale[getRandomInt(firstNameFemale.length)]
                + "'," + "'Woman'" + "," + date + ",'"
                + countiesEng[getRandomInt(countiesEng.length)] + "','" +
                ↪  secondName[getRandomInt(secondName.length)] + "','"
                + firstNameFemale[getRandomInt(firstNameFemale.length)]
                + "','" + fieldPosition[getRandomInt(fieldPosition.length)] +
                ↪  "');");
        writer.newLine();
    }
```

```
}

public static int getRandomInt(int number) {
    return new Random().nextInt(number);
}

public static void firstItr() {
    Charset charset = StandardCharsets.UTF_8;
    try (BufferedWriter writer =
    ↪ Files.newBufferedWriter(Path.of("/Users/nnnn/6д/fill_1.sql"), charset)) {
        writer.write("USE BD;");
        writer.newLine();
        fillCountry(writer);
        fillType_of_football(writer);
        fillChampionship(writer);
        fillFootball_characteristics(writer);
        fillSeason(writer);
        fillPrice(writer);
        fillFootballer(writer);
    } catch (IOException x) {
        System.err.format("IOException: %s%n", x);
    }
}
public static void fillRatingParticipant(BufferedWriter writer) throws
↪ IOException {//Ок
    writer.write("DELETE FROM Rating_participant;");
    writer.newLine();
    String[] status = {"'professional'", "'amateur'"};
    for (int i = 1; i <= 750; i++) {
            int countryId = getRandomInt(182) + 1;
            writer.write("INSERT INTO Rating_participant VALUES(" + sumId + ","
            ↪  + 1 + "," + countryId + "," + i + ","
                    + status[getRandomInt(2)] + ");");
            writer.newLine();
            sumId++;
    }
    for (int i = 751; i <= 1500; i++) {
            int countryId = getRandomInt(182) + 1;
            writer.write("INSERT INTO Rating_participant VALUES(" + sumId + ","
            ↪  + 2 + "," + countryId + "," + i + ","
                    + status[getRandomInt(2)] + ");");
            writer.newLine();
            sumId++;
    }
}
public static int randInt(int min, int max) {
 Random rand = new Random();
    return rand.nextInt((max - min) + 1) + min;
}
public static void fillCostOfFootballCharacteristicPerSeason(BufferedWriter
↪ writer) throws IOException {
    writer.write("DELETE FROM Cost_of_football_characteristic_per_season;");
    writer.newLine();
```

```
    int id = 1;
    for (int j = 1; j <= 6; j++) {
    for (int i = 1; i <= 20; i++) {
        int rndCost = getRandomInt(20) + 1;
        int rndSeason = getRandomInt(20) + 1;
        int ranFc = getRandomInt(6)+1;
                writer.write("INSERT INTO
                ↪  Cost_of_football_characteristic_per_season VALUES (" + id +
                ↪  "," + rndCost + "," + ranFc  + "," + rndSeason + ");");
                writer.newLine();
                id++;
        }
    }
}

public static void CostOfChampionshipPerSeason(BufferedWriter writer) throws
↪  IOException {
    writer.write("DELETE FROM Cost_of_championship_per_season;");
    writer.newLine();
    int id = 1;
    for (int j = 1; j <= 9; j++) {
         for (int i = 1; i <= 20; i++) {
             int rndCost = getRandomInt(20) + 1;
             int rndSeason = getRandomInt(20) + 1;
             int ranChamp = getRandomInt(9)+1;

                writer.write("INSERT INTO Cost_of_championship_per_season VALUES
                ↪  (" + id + "," + rndCost + "," + ranChamp + "," + rndSeason
                ↪  +  ");");
                writer.newLine();
                id++;
        }
    }
}

public static void secondItr() {
    Charset charset = StandardCharsets.UTF_8;
    try (BufferedWriter writer =
    ↪  Files.newBufferedWriter(Path.of("/Users/nnnn/6∂/fill_2.sql"), charset)) {
        writer.write("USE BD;");
        writer.newLine();
        fillRatingParticipant(writer);
        fillCostOfFootballCharacteristicPerSeason(writer);
        CostOfChampionshipPerSeason(writer);
    } catch (IOException x) {
        System.err.format("IOException: %s%n", x);
    }
}

public static int getRandom(int[] array) {
    int rnd = new Random().nextInt(array.length);
    return array[rnd];
}
```

```
public static void fillCostOfParticipantPerSeason(BufferedWriter writer) throws
↪  IOException {
    writer.write("DELETE FROM Cost_of_participant_per_season;");
    writer.newLine();
    int id = 1;
    for(int i = 1; i <= 1500; i++) {//каждый участник
            int rndCompetitor = getRandomInt(1500) + 1;
            int randomNum1 = randInt(10, 15);
            for (int rc = 0; rc <= randomNum1; rc++) {
                int rndSeason = getRandomInt(20) + 1;
                writer.write("INSERT INTO Cost_of_participant_per_season VALUES
                ↪  (" + id + "," + randInt(1, 180) + "," + "null" + "," + i +
                ↪  "," + rndSeason + ");");
                writer.newLine();
                id++;
            }
            randomNum1 = randInt(1, 5);
            for (int k = 0; k <= randomNum1; k++) {
                int rndSeason = getRandomInt(20) + 1;
                writer.write("INSERT INTO Cost_of_participant_per_season VALUES
                ↪  (" + id + "," + "null" + "," + randInt(1, 120) + "," + i +
                ↪  "," + rndSeason + ");");
                writer.newLine();
                id++;
            }
    }
}
public static void thirdItr() {
    Charset charset = StandardCharsets.UTF_8;
    try (BufferedWriter writer =
    ↪  Files.newBufferedWriter(Path.of("/Users/nnnn/6д/fill_3.sql"), charset)) {
        writer.write("USE BD;");
        writer.newLine();
        fillCostOfParticipantPerSeason(writer);
    } catch (IOException x) {
        System.err.format("IOException: %s%n", x);
    }
}


public static void fillRatingList(BufferedWriter writer) throws IOException {
    writer.write("DELETE FROM Rating_list;");
    writer.newLine();
    int id = 1;
    for (int i = 1; i <= 1500; i++) {
        int rndUchastie = randInt(5,10);
        for (int j = 1; j <= rndUchastie; j++) {
            int rndTotalPoints = getRandom(totalPrice);
            int rndSeason = randInt(1, 20);
            writer.write("INSERT INTO Rating_list VALUES (" + id + "," + i +
            ↪  "," + rndTotalPoints + "," + rndSeason +");");
            writer.newLine();
            id++;
```

```
        }
    }
}

public static void fourthItr() {
    Charset charset = StandardCharsets.UTF_8;
    try (BufferedWriter writer =
    ↪ Files.newBufferedWriter(Path.of("/Users/nnnn/6∂/fill_4.sql"), charset)) {
        writer.write("USE BD;");
        writer.newLine();
        fillRatingList(writer);
    } catch (IOException x) {
        System.err.format("IOException: %s%n", x);
    }
}


public static void main(String[] args) {
     firstItr();
    secondItr();
    thirdItr();
    fourthItr();
}
```