Экзаменационный проект   
по дисциплине  
Проектирование баз данных

студента гр. **M34351**

**Юльцовой Натальи Алексеевны**

по теме

**Пляжный футбол**

# Описание проекта

В проекте рассмотрена Общая база турниров по пляжному футболу некой страны. В каждом турнире играет несколько команд по расписанию матчей. Цель матча – выявить победителя и начислить ему очки.  
Время матча - 3 тайма по 12 минут, плюс дополнительное время 3 минуты. За победу в матче в основное время плюс дополнительное команда получает 3 очка, за проигрыш 0. Матч может выиграть только одна команда. Если счет по окончании времени игры равный, то решение о победе принимается по послематчевой серии пенальти (девятиметровых ударов) - победителем матча становится команда, забившая большее количество голов при одинаковом количестве пробитых ударов. Команда, победившая в серии пенальти зарабатывает 2 очка, проигравшая - 1.

Размер поля очень маленький, поэтому позиции игроков размыты и не имеют значения - нередко вратари забивают наравне с другими участниками голы, поэтому в отдельную категорию амплуа не выносится, сейвов (удачное прерывание атаки противника в воротах) слишком много, замен не лимитированное количество, значит будем рассматривать только следующие типы событий – голы, реализованный (успешная попытка) и не реализованный пенальти, количество желтых и красных карточек. Существуют автоголы - игрок команды забивает в свои ворота, это так же будет учтено.

Игроки могут переходить из команды в команду. Игрок может играть в разных командах (например, за сборную и за региональную команду) и для каждой у него может быть разный номер на футболке, но номера футболок внутри команды не повторяются. Известно, когда игрок вступил в команду и ушел из нее. Игрок не может в один день вступить в одну и ту же команду дважды. В рамках одного турнира в одно и тоже время не может проходить одновременно несколько матчей.

Чтобы попасть на независимый турнир (турнир с одним этапом) , или низкоуровневый отборочный турнир (турнир на который нет отбора, но по его результатам проходят в следующий этап) нужно просто подать заявку на участие в данном турнире. Но есть турниры на которые проходят отборы. Тогда команда должна быть заявлена в отборочном этапе, чтобы пройти в следующий. Команда по результатам отборочного этапа может либо пройти выше, либо не пройти. При этом команда может подавать заявку на любое количество низкоуровневых отборочных турниров. Количество отборочных турниров для определенного этапа и количество этапов произвольное.

Команд с одинаковым одновременно городом и названием нет.

# Построение отношений

В результате предварительного проектирования были выделены следующие отношения:

* Tournament – список турниров.
* TournamentParticipation – информация о участвующих командах в рамках турнира.
* Match – информация о матче между 2 командами и его результатах.
* Team – список футбольных команд.
* Player – информация о футболисте.
* EventLog – информация о важных событиях матча – гол, карточки, пенальти.

## Отношение Tournament

Атрибуты:

* TournamentID - уникальный идентификатор для каждого турнира.
* TournamentName - название турнира,.
* ParentTournamentID - id турнира для которого данный турнир является отборочным.

Функциональные зависимости:

* TournamentID -> TournamentName
* TournamentName -> TournamentID
* TournamentID -> ParentTournamentID

Ключи:

* TournamentID, TournamentName

## Отношение TournamentParticipation

Атрибуты:

* TournamentID - уникальный идентификатор турнира.
* TeamID - внешний ключ, связывающий турнир с одной из участвующих команд.
* Status - статус участия команды в турнире (“Участвует”, “В рассмотрении”, “Заявка отклонена”

Функциональные зависимости:

* (TournamentID, TeamID) → Status

Ключи:

* (TournamentID, TeamID)

Отношение Match(до нормализации)

Атрибуты:

* MatchID - уникальный идентификатор матча.
* TournamentID - внешний ключ, связывающий матч с турниром.
* DateTime - дата и время проведения матча.
* Team1ID - внешний ключ, связывающий матч с первой участвующей командой.
* Team2ID - внешние ключ, связывающий матч с второй участвующей командой.
* ScoreTeam1 - количество полученных очков по результатам матча командой 1.
* ScoreTeam2 - количество полученных очков по результатам матча командой 2.

Функциональные зависимости:

* MatchID → TournamentID, DateTime, Team1ID, Team2ID
* (MatchID, Team1ID) -> ScoreTeam1
* (MatchID, Team2ID) -> ScoreTeam2

Ключи:

* MatchID

Нормализация для отношения Match.  
Атрибуты атомарны, есть ключ, есть повторяющиеся группы - Team1ID и TeamID2. Решение: уберем разделение на команды и будем делать по 2 записи на матч. Измененное отношение - (MatchID, TournamentID, DateTime, TeamID, ScoreTeam) - 1НФ

Неключевые атрибуты функционально зависят от части ключа. Решение: сделаем декомпозицию по ФЗ (MatchID, TeamID) -> ScoreTeam: уберем из Match TeamID и ScoreTeam и теперь атрибуты будут зависеть только от ключа MatchID и создадим новое отношение MatchParticipation для атрибутов, зависящих от ключа (MatchID, TeamID).

Новые отношения - (MatchID, TournamentID, DateTime), (MatchID, TeamID, ScoreTeam).

- 2 НФ

В новых отношениях неключевые атрибуты непосредственно зависят от ключей - 3НФ, нетривиальных зависимостей нет - НФБК, нет МЗ не являющихся ФЗ - 4НФ, нет ЗС не являющихся ФЗ - 5НФ.

Отношение Match  
 Атрибуты:

* MatchID - уникальный идентификатор матча.
* TournamentID - внешний ключ, связывающий матч с турниром.
* DateTime - дата и время проведения матча.

Функциональные зависимости:

* MatchID → TournamentID
* MatchID → DateTime

Ключи:

* MatchID

## Отношение MatchParticipation

Атрибуты:

* MatchID - уникальный идентификатор матча.
* TeamID - внешний ключ, связывающий матч с одной из участвующих команд.
* ScoreTeam - количество очков, полученных командой по результатам данного матча.

Функциональные зависимости:

* (MatchID, TeamID) → ScoreTeam

Ключи:

* (MatchID, TeamID)

## Отношение Team

Атрибуты:

* TeamID - уникальный идентификатор команды.
* Name - название команды.
* Hometown - город, из которого происходит команда.

Функциональные зависимости:

* TeamID -> Name
* TeamID -> Hometown
* (Name, Hometown) -> TeamID

Ключи:

* TeamID
* (Name, Hometown)

Нормализация для отношения Team  
Атрибуты атомарны, есть ключ, нет повторяющихся групп - 1НФ

Неключевые атрибуты функционально не зависят от части ключа - 2 НФ

Неключевые атрибуты непосредственно зависят от ключей - 3НФ Нетривиальных зависимостей нет - НФБК, нет МЗ не являющихся ФЗ (отбросим МЗ по контрпримерам ниже) или теореме Дейта-Фейгина 2 что в НФБК и существует простой ключ - 4НФ:

Возможные МЗ:

TeamID ->> Name|Hometown - ФЗ

Name ->> TeamId|Hometown - у одного имени команды могут быть разные id и разные города, пример (1) + (2)

Hometown ->> Name|TeamID - в одном городе могут быть команды с разными именами и id, пример (2) + (3)

Пример:

TeamID Name Hometown

1 Cпартак Москва (1)

2 Спартак Сочи (2)

3 Элмонт Сочи (3)

5НФ:  
Не все ключи простые поэтому не можем воспользоваться теоремой.

Возможная ЗС \*{(TeamID, Name), (Name, Hometown), (Hometown, TeamId)}

TeamId Name Hometown  
1 Спартак Москва  
2 Спартак Сочи

(TeamID, Name) (Name, Hometown) (Hometown, TeamId)

1 Спартак Спартак Москва Москва 1  
2 Спартак Спартак Сочи Сочи 2

Объединим:

TeamId Name Hometown  
1 Спартак Москва  
2 Спартак Сочи

И 2 некорректные комбинации

1 Спартак Сочи

2 Спартак Москва

Соответственно, не ЗС

, нет ЗС не являющихся ФЗ - 5НФ.

Отношение в 5НФ изначально, оставляем как было.

## Отношение Player (до нормализации)

Атрибуты:

* PlayerID - уникальный идентификатор игрока.
* Name - имя игрока.
* TeamID - внешний ключ, связывающий игрока с командой, за которую он играет.
* Name – имя игрока.
* PlayerNumber – номер игрока в команде.
* Nationality - страна происхождения игрока.
* Birthdate - дата рождения игрока

Функциональные зависимости:

* PlayerID → Name, Nationality, Birthdate
* (PlayerID, TeamId) -> PlayerNumber

Ключи:

* PlayerID

Нормализация для отношения Player  
Атрибуты атомарны, есть ключ, нет повторяющихся групп - 1НФ

Неключевые атрибуты функционально зависят от части ключа. Решение: сделаем декомпозицию по ФЗ (PlayerID, TeamId) -> PlayerNumber -

уберем из Player PlayerNumber и теперь атрибуты будут зависеть только от ключа PlayerID и создадим новое отношение TeamMembership для атрибутов, зависящих от ключа (PlayerID, TeamId).

Новые отношения - (PlayerID, Name, Nationality, Birthdate), (PlayerID, TeamId, PlayerNumber).

- 2 НФ

В новых отношениях неключевые атрибуты непосредственно зависят от ключей - 3НФ, нетривиальных зависимостей нет - НФБК, нет МЗ не являющихся ФЗ - 4НФ, нет ЗС не являющихся ФЗ - 5НФ.

## Отношение Player

Атрибуты:

* PlayerID - уникальный идентификатор игрока.
* Name - имя игрока.
* Nationality - страна происхождения игрока.
* Birthdate - дата рождения игрока

Функциональные зависимости:

* PlayerID → Name
* PlayerID → Nationality
* PlayerID → Birthdate

Ключи:

* PlayerID

## Отношение TeamMembershipHistory

Атрибуты:

* PlayerID - уникальный идентификатор игрока.
* HistoryIMembershipID - уникальный идентификатор записи с историей членства
* TeamID - внешний ключ, связывающий игрока с командой, за которую он играет.
* PlayerNumber – номер игрока в команде (его пишут на футболке)
* JoinDate - дата начала членства в данной команде.
* LeaveDate - дата окончания членства в данной команде.

Функциональные зависимости:

* HistoryIMembershipID -> PlayerID, TeamID, JoinDate, LeaveDate, PlayerNumber
* (PlayerID, TeamID, JoinDate) -> LeaveDate, PlayerNumber, HistoryIMembershipID

Ключи:

* HistoryIMembershipID, (PlayerID, TeamID, JoinDate)

## Отношение EventLog

Атрибуты:

* EventID - уникальный идентификатор события.
* MatchID - внешний ключ, связывающий с матчем, в котором произошло событие.
* EventType - тип события (гол, желтая карточка, красная карточка и т.д.).
* EventTime – минута матча в которую произошло событие.
* PlayerID – игрок с которым связано это событие.
* ~~TeamID – команда, за которую играет данный игрок.~~

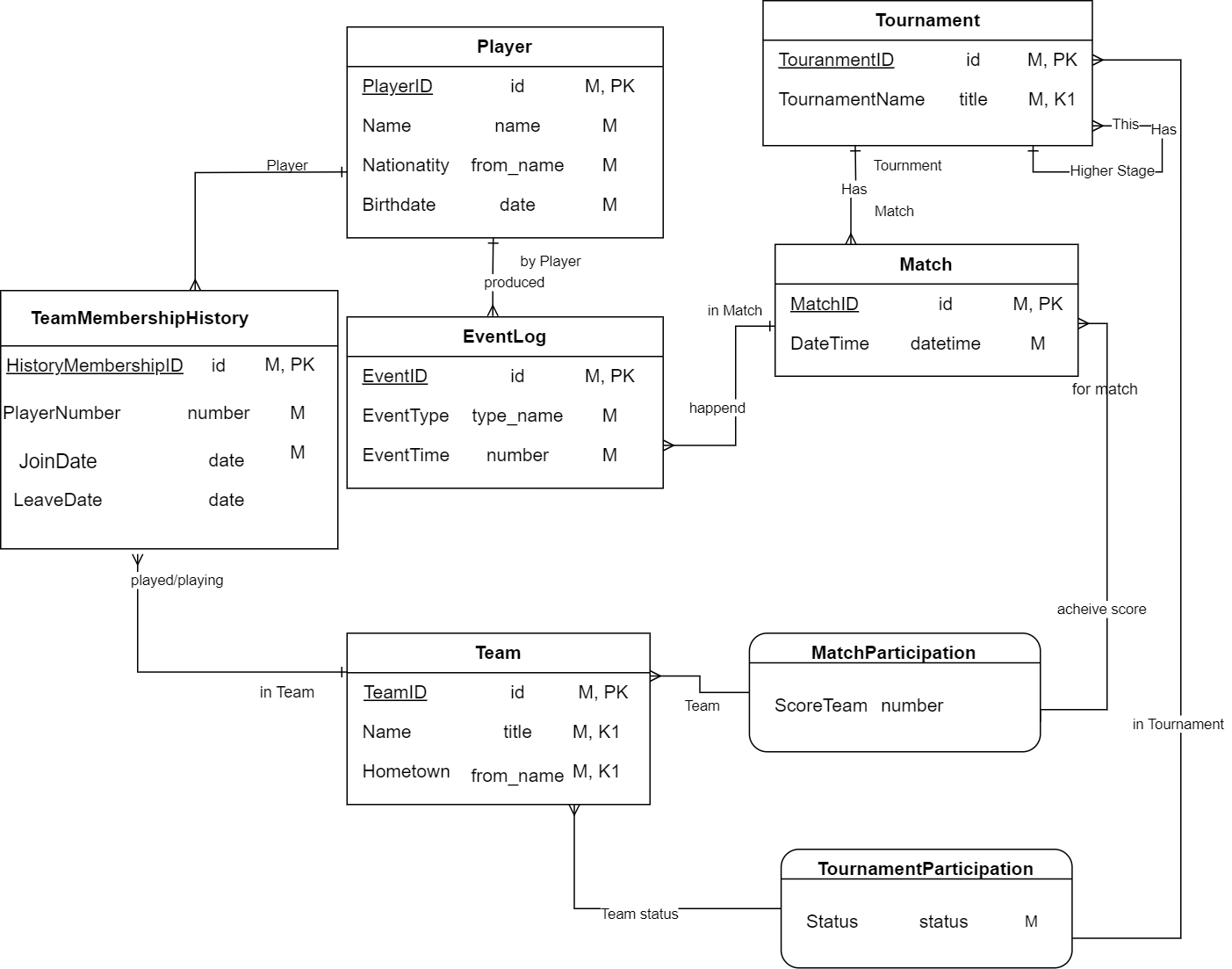
Функциональные зависимости:

* EventID → MatchID
* EventID → EventType
* EventID → EventTime
* EventID → PlayerID  
  Примечание: Даже если взять вместе все атрибуты кроме EventID, то мы не сможем вычислить EventID по ним так как в одном матче в одну и ту же минуту один и тот же игрок может сделать 2 одинаковых действия, поэтому получаем только тривиальные ФЗ.

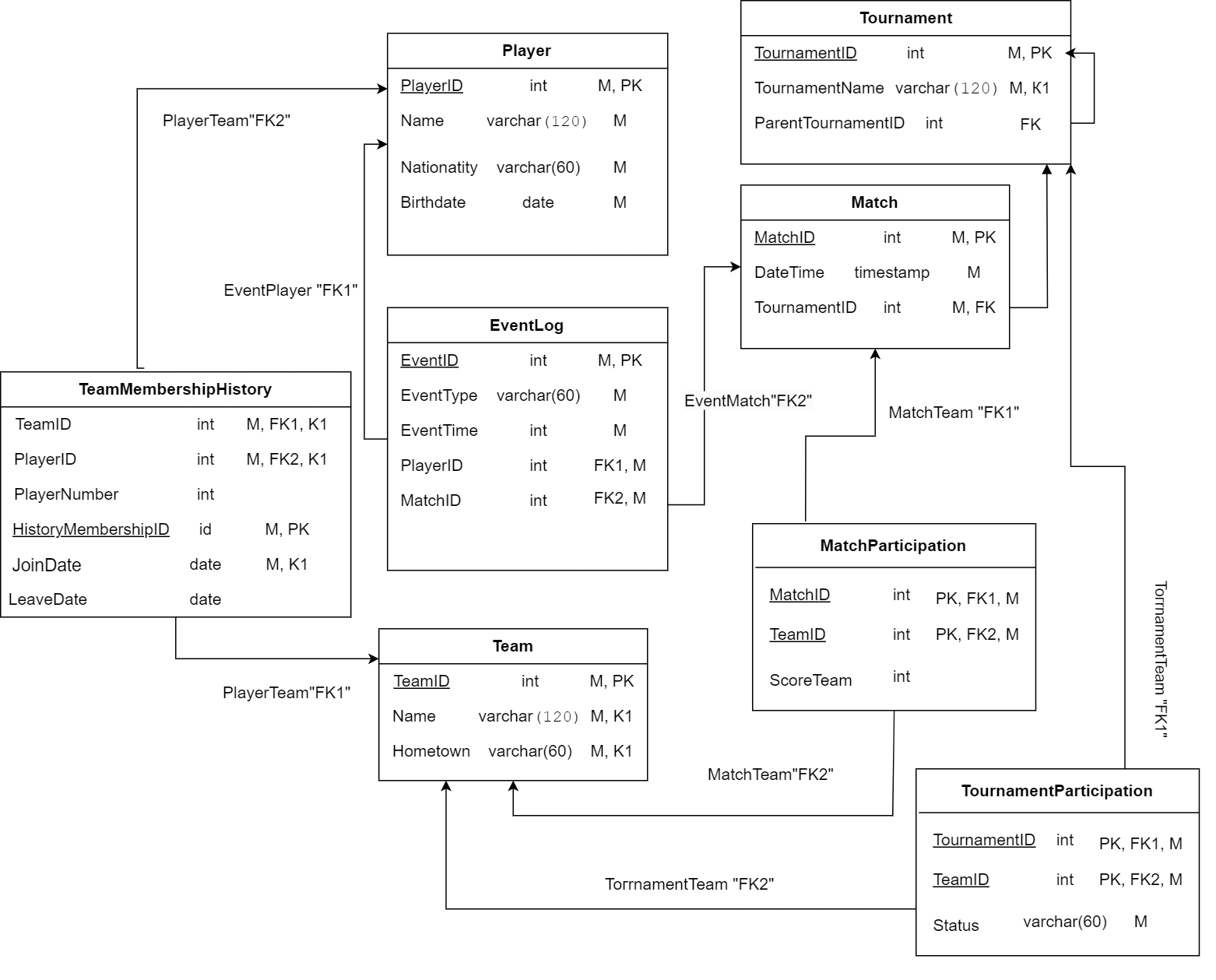
Ключи:

* EventID

# Модель сущность-связь



# Физическая модель



При построении физической модели использовалось следующее отображение доменов в типы:

| Домен | Тип |
| --- | --- |
| id | int |
| name | varchar(120) |
| number | int |
| date | date |
| datetime | timestamp |
| from\_name | varchar(60) |
| title | varchar(120) |
| type\_name | varchar(60) |
| status | varchar(60) |

# Определения схемы

Для реализации проекта использовалась СУБД PostgreSQL 15.4. Определения таблиц и их индексов приведено в файле ddl.sql (Приложение 1).

# Тестовые данные

Скрипт для добавления тестовых данных приведен в файле insert.sql (Приложение 2).

# Запросы на получение данных

В рамках проекта были реализованы следующие запросы:

* todays\_teams\_participants - текущий состав всех команд.
* tournaments\_score\_table — очки в турнирной таблице.
* team\_total\_win\_loses — подсчет побед и поражений каждой команды за ее существование. Победа/поражение по пенальти и в основное время это разные колонки.
* tournaments\_match\_cnt - количество матчей в рамках турнира
* player\_penalty\_percent - процент реализации пенальти (забитые \*100) / пробитые
* player\_cards - количества желтых и красных карточек у всех игроков
* player\_cards\_without - cписок игроков, не получавших карточек
* avg\_team\_age - средний возраст игроков команды
* tournament\_next\_stage - турнир и этап который следует за ним
* spartak\_matches - все матчи команды спартак
* russian\_leaque\_results - результаты всех матчей в Лиге России 2024
* team\_matches\_cnt - количество сыгранных матчей каждой командой
* top\_chages\_team\_membership - топ-3 по последней смене состава
* this\_year\_team\_new\_players - игроки, которые пришли в команду в текущем году
* this\_month\_matches - матчи за последний месяц с 1 числа текущего месяца
* membership\_lenght - продолжительность участия игроков в командах
* eventiest\_matches - топ-5 матчей с наибольшим количеством событий
* total\_best\_players - лучшие бомбардиры (наиболее результативные игроки по голам) за все время
* tournament\_best\_players - лучшие бомбардиры турниров
* multiple\_goals\_in\_match - игроки которые забили больше 1 гола за матч

Для реализации запросов было создано вспомогательное представление:

* PlayerGoalsPerMatch - количество голов забитых игроком за матч, помогает в поиске бомбардиров и результативных игроков матча

Запросы на получение данных и вспомогательные представления приведены в файле select.sql (Приложение 3).

# Запросы на изменение данных

В рамках проекта были реализованы следующие запросы:

* insert\_player\_in\_team - вставка нового игрока в команду
* update\_player\_number\_in\_team - изменение номера игрока на футболке
* insert\_in\_match\_event - добавление события в матч
* insert\_match\_participation - добавление команды в матч
* update\_match\_datetime - обновление времени матча
* leave\_team\_today - выход из команды игроком сегодняшней датой
* cancel\_match - отмена матча
* RejectTournamentApplications - отклонение неподходящих заявок “В рассмотрении” на участие в турнире

Запросы на изменение данных, хранимые процедуры и триггеры приведены в файле update.sql (Приложение 4).