**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ**

**по заданию «Реализация приложения базы данных в архитектуре клиент-сервер»**

**Обучающегося Балашова Владислава Олеговича группы №20211 курса 3**

**Тема задания:** «Информационная система спортивных организаций города»

Новосибирск 2023

1. **Описание проекта**

Спортивная инфраструктура города представлена спортивными сооружениями различного типа: спортивные залы, манежи, стадионы, корты и т.д. Каждая из категорий спортивных сооружений обладает атрибутами, специфичными только для нее: стадион характеризуется вместимостью, корт - типом покрытия.

Спортсмены под руководством тренеров занимаются отдельными видами спорта, при этом один и тот же спортсмен может заниматься несколькими видами спорта, и в рамках одного и того же вида спорта может тренироваться у нескольких тренеров. Все спортсмены объединяются в спортивные клубы, при этом каждый из них может выступать только за один клуб.

Организаторы соревнований проводят состязания по отдельным видам спорта на спортивных сооружениях города. По результатам участия спортсменов в соревнованиях производится награждение.

Для выбранного проекта студент разрабатывает структуру базы данных и реализует приложение в архитектуре **клиент-сервер**, выполняющее операции внесения данных в базу данных, редактирование данных и запросы, указанные в проекте. Клиентская часть реализуется на языке программирования высокого уровня. В описании проекта дана обобщенная пользовательская спецификация приложения.

Спецификация не предполагает оптимального определения структур данных, но задает полный перечень хранимой в базе данных информации и выполняемых программой функций. Таблицы должны заполняться адекватными значениями, примерно по 7 объектов в каждой.

**Разработка проекта предполагает выполнение следующих этапов:**

1. Разработка структуры базы данных (серверная часть)

1.1 Проектирование инфологической модели задачи. Определение сущностей, атрибутов

сущностей, идентифицирующих атрибутов, связей между сущностями. При

проектировании должны учитываться требования гибкости структур для выполнения

перечисленных функций и не избыточного хранения данных.

1.2 Проектирование схемы базы данных: описание схем таблиц, типов (доменов) атрибутов,

определение ограничений целостности. Написание SQL скриптов по созданию таблиц БД.

1.3 Создание и заполнение разработанной БД на стороне сервера.

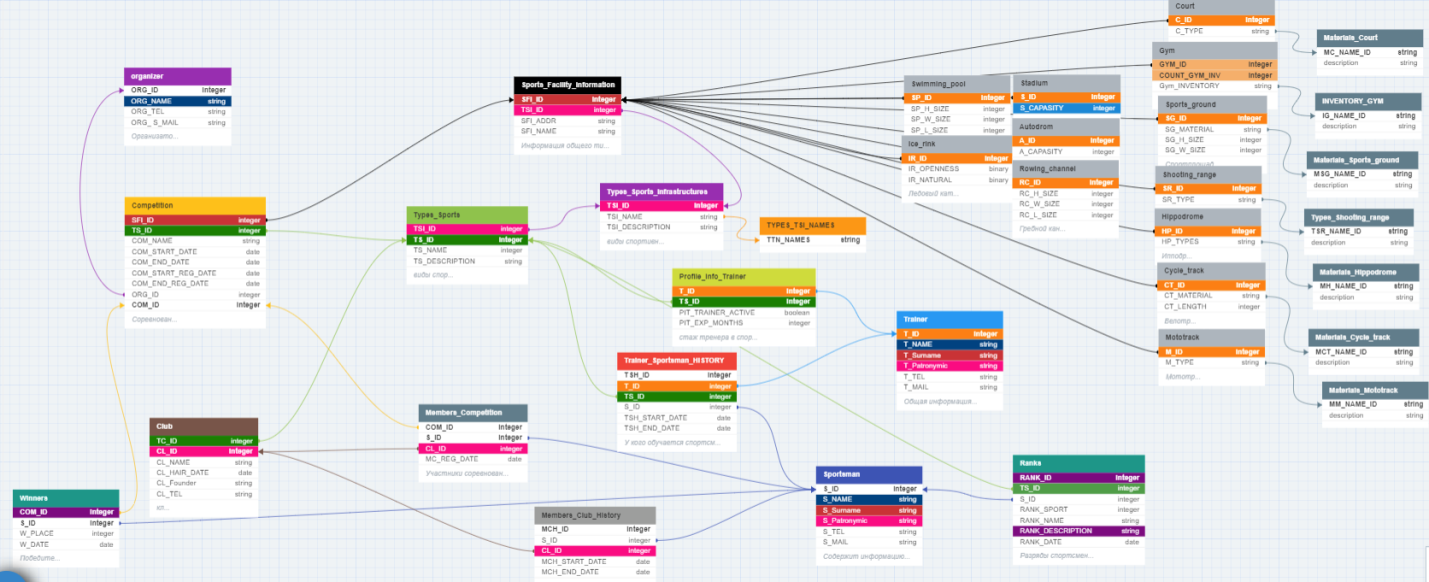
2. Написание SQL запросов к спроектированной базе данных согласно заданию.

3. Реализация триггеров и хранимых процедур (PL / SQL).

4. Разработка приложения клиента (формы ввода, редактирования и поиска данных по

запросам)

**Графическая схема**

****

Ссылка на БД схему: https://dbdesigner.page.link/ieuKtcWEJM2prTPR6

**Описание таблиц и сущностей**

Подчеркнутые атрибуты - ключевые атрибуты

**Подчёркнутые таблицы** – таблице имеющие схожее предназначение

**Sports\_Facility\_Information**(SFI\_ID,TSI\_ID,SFI\_ADDR,SFI\_NAME)

Описание: единицы спортивных сооружений, а так же информация принадлежащая им.

Содержание:

* Само спортивное сооружение, как таковое – SFI\_ID
* Вид инфраструктуры – TSI\_ID
* Адрес спортивного сооружения – TSI\_ADDR
* Название спортивного сооружения – SFI\_NAME

**Types\_Sports\_Infrastructures**(TSI\_ID,TSI\_NAME, TSI\_DESCRIPTION)

Описание: виды спортивных инфраструктур

Содержание:

* Сам вид спортивной инфраструктуры, как таковой – TSI\_ID
* Название вида спортивной инфраструктуры – TSI\_NAME
* Описание спортивной инфраструктуры – TSI\_DESCRIPTION

**Types\_Sports**(TSI\_ID, TS\_ID, TS\_NAME, TS\_DESCRIPTION)

Описание: виды спорта

Содержание:

* Вид спортивной инфраструктуры, на которой обычно проводится данный вид спорта – TSI\_ID
* Сам вид спорта, как таковой – TS\_ID
* Название вида спорта – TS\_NAME
* Описание вида спорта – TS\_DESCRIPTION

**Competition**(COM\_ID, TS\_ID, SFI\_ID, COM\_NAME,COM\_START\_DATE, COM\_END\_DATE, COM\_END\_REG\_DATE, ORG\_ID)

Описание: единицы спортивных соревнований, а также общая информациях принадлежащая им

Содержание:

* Само спортивное соревнование, как таковое – COM\_ID
* Вид спорта, к которому принадлежит данное соревнование – TS\_ID
* Спортивное сооружение, на котором проводится данное спортивное соревнование – SFI\_ID
* Название спортивного соревнования – COM\_NAME
* Дата начала соревнования – COM\_START\_DATE
* Дата окончания соревнования – COM\_END\_DATE
* Дата окончания регистрации на соревнование –COM\_END\_REG\_DATE
* Основной организатор данного соревнования – ORG\_ID

**Organizer**(ORG\_ID, ORG\_NAME, ORG\_TEL, ORG\_S\_MAIL)

Описание: организаторы и информация, относящаяся к ним

Содержание:

* Сам организатор, как таковой – ORG\_ID
* Наименование организации – ORG\_NAME
* Телефон организации – ORG\_TEL
* Почта организации – ORG\_S\_MAIL

**Members\_Competition**(COM\_ID,S\_ID,CL\_ID, MC\_REG\_DATE)

Описание: участники соревнований и информация касающаяся их

Содержание:

* Соревнование, в котором участвует участник – COM\_ID
* Спортсмен, выступающий в роли участника – S\_ID
* Клуб, за который выступает спортсмен – CL\_ID
* Дата регистрации участника – MC\_REG\_DATE

**Winners**(COM\_ID, S\_ID, W\_PLACE,W\_DATE)

Описание: победители соревнования, занявшие призовые места

Содержание:

* Соревнование, в котором выиграл участник – COM\_ID
* Участник, занявший призовое место – S\_ID
* Место, принадлежащее данному участнику – W\_PLACE
* Дата получения призового места – W\_DATE

**Ranks**(RANK\_ID, S\_ID, TS\_ID, RANK\_SPORT, RANK\_NAME,RANK\_DESCRIPTION,RANK\_DATE)

Описание: разряды, которые получил спортсмен за всё время

Содержание:

* Сам разряд, как таковой – RANK\_ID
* Спортсмен, которому присвоен данный разряд – S\_ID
* Вид спорта – TS\_ID
* Разряд по спорту – RANK\_SPORT
* Наименование данного разряда – RANK\_NAME
* Описание данного разряда – RANK\_DESCRIPTION
* Дата получения спортсменом данного разряда – RANK\_DATE

**Members\_Club\_HISTORY**(MCH\_ID,S\_ID,CL\_ID, MCH\_START\_DATE, MCH\_END\_DATE)

Описание: история вступления и ухода участников в клубе

Содержание:

* Фиксация факта вступления спортсмена в клуб – MCH\_ID
* Спортсмен, вступивший в клуб – S\_ID
* Клуб, в который вступили – CL\_ID
* Дата вступления участника в клуб – MCH\_START\_DATE
* Дата ухода участника из клуба – MCH\_END\_DATE

**Club**(TC\_ID,CL\_ID,CL\_NAME,CL\_HAIR\_DATE,CL\_FOUNDER,CL\_TEL)

Описание: клубы и прилагающаяся к ним информация

Содержание:

* Вид спорта, на котором специализируется данный клуб – TC\_ID
* Сам клуб, как таковой – CL\_ID
* Наименование данного клуба – CL\_NAME
* Дата создания данного клуба – CL\_HAIR\_DATE
* Основатель данного клуба – CL\_FOUNDER
* Телефон данного клуба – CL\_TEL

**Profile\_Info\_Trainer**(T\_ID,TS\_ID,PIT\_EXP\_MONTHS)

Описание: стаж тренера в спорте

Содержание:

* Сам тренер, которому принадлежит стаж – T\_ID
* Вид спорта, которым занимается тренер – TS\_ID
* Количество месяцев стажа – PIT\_EXP\_MONTHS

**Trainer\_Sportsman\_HISTORY**(TSH\_ID,T\_ID,TS\_ID,S\_ID,TSH\_START\_DATE,TSH\_END\_DATE)

Описание: история обучения спортсмена у тренера

Содержание:

* Фиксация факта обучения спортсмена у тренера – TSH\_ID
* Тренер, у которого занимался спортсмен – T\_ID
* Вид спорта, по которому проходили занятия – TS\_ID
* Спортсмен, который обучается – S\_ID
* Дата начала обучения – TSH\_START\_DATE
* Дата окончания обучения – TSH\_END\_DATE

**Trainer**(T\_ID,T\_NAME,T\_SURNAME,T\_PATRONYMIC,T\_TEL,T\_MAIL)

Описание: тренеры и информация, прилагающаяся к ним

Содержание:

* Сам тренер, как таковой – T\_ID
* Имя данного тренера – T\_NAME
* Фамилия данного тренера – T\_SURNAME
* Отчество(если есть) данного тренера – T\_PATRONYMIC
* Телефон данного тренера – T\_TEL
* Почта данного тренера – T\_MAIL

**Sportsman**(S\_ID, S\_NAME, S\_SURNAME, S\_PATRONYMIC, S\_TEL,S\_MAIL)

Описание: спортсмены и информация прилагающаяся к ним

Содержание:

* Сам спортсмен, как таковой – S\_ID
* Имя данного спортсмена – S\_NAME
* Фамилия данного спортсмена – S\_SURNAME
* Отчество(если есть) данного спортсмена – S\_PATRONYMIC
* Телефон данного спортсмена – S\_TEL
* Почта данного спортсмена – S\_MAIL

**Mototrack**(M\_ID, M\_TYPE)

Описание: спортивная инфраструктура мототрек

Содержание:

* Сама инфраструктура, как таковая – M\_ID
* Атрибут определяющий тип покрытия – M\_TYPE

**Sports\_ground**(SG\_ID,SG\_MATERIAL,SG\_H\_SIZE,SG\_W\_SIZE)

Описание: спортивная инфраструктура спортплощадка

Содержание:

* Сама инфраструктура, как таковая – SG\_ID
* Материал данной спортивной площадки – SG\_MATERIAL
* Длина данной спортивной площадки – SG\_H\_SIZE
* Ширина данной спортивной площадки – SG\_W\_SIZE

**Swimming\_pool**(SP\_ID,SP\_H\_SIZE,SP\_W\_SIZE,SP\_L\_SIZE)

Описание: спортивная инфраструктура плавательный бассейн

Содержание:

* Сама инфраструктура, как таковая – SP\_ID
* Глубина данного плавательного бассейна – SP\_H\_SIZE
* Ширина данного плавательного бассейна – SP\_W\_SIZE
* Протяженность данного спортивного бассейна – SP\_L\_SIZE

**Cycle\_track**(CT\_ID,CT\_MATERIAL,CT\_LENGTH)

Описание: спортивная инфраструктура велотрек

Содержание:

* Сама инфраструктура, как таковая – CT\_ID
* Материал покрытия данной трассы – CT\_MATERIAL
* Протяженность данного велотрека – CT\_LENGTH

**Court**(C\_ID,C\_TYPE)

Описание: спортивная инфраструктура корт

Содержание:

* Сама инфраструктура, как таковая – C\_ID
* Тип покрытия данного корта – C\_TYPE

**Stadium**(S\_ID,S\_CAPASITY)

Описание: спортивная инфраструктура стадион

Содержание:

* Сама инфраструктура, как таковая – S\_ID
* Вместимость данного стадиона(кол-во человек) – S\_CAPASITY

**Gym**(GYM\_ID, COUNT\_GYM\_INV, GYM\_INVENTORY)

Описание: спортивная инфраструктура спортзал

Содержание:

* Сама инфраструктура, как таковая – GYM\_ID
* Номер инвентаря - COUNT\_GYM\_INV
* Инвентарь данного спортзала – GYM\_INVENTORY

**Rowing\_channel**(RC\_ID,RC\_H\_SIZE,RC\_W\_SIZE,RC\_L\_SIZE)

Описание: спортивная инфраструктура гребной канал

Содержание:

* Сама инфраструктура, как таковая – RC\_ID
* Глубина данного гребного канала – RC\_H\_SIZE
* Ширина данного гребного канала – RC\_W\_SIZE
* Протяженность данного гребного канала – RC\_L\_SIZE

**Shooting\_range**(SR\_ID,SR\_TYPE)

Описание: спортивная инфраструктура стрелковый тир

Содержание:

* Сама инфраструктура, как таковая – SR\_ID
* Тип размещения данного тира – SR\_TYPE

**Autodrom**(A\_ID,A\_CAPASITY)

Описание: спортивная инфраструктура автодром

Содержание:

* Сама инфраструктура, как таковая – A\_ID
* Размерность данного автодрома(кол-во людей) – A\_CAPASITY

**Ice\_rink**(IR\_ID, IR\_OPENNESS, IR\_NATURAL)

Описание: спортивная инфраструктура ледовый каток

Содержание:

* Сама инфраструктура, как таковая – IR\_ID
* Открытость помещения данного ледового катка – IR\_OPENNESS
* Естественность создания данного ледового катка – IR\_NATURAL

**Hippodrome**(HP\_ID, HP\_TYPES)

Описание: спортивная инфраструктура ипподром

Содержание:

* Сама инфраструктура, как таковая – HP\_ID
* Тип покрытия данного ипподрома – HP\_TYPES

**Materials\_Court**(MC\_NAME\_ID, description)

Описание: виды материалов для court

Содержание:

* Сам материал, как таковой – MC\_NAME\_ID
* Его описание - description

**INVENTORY\_GYM**(IG\_NAME\_ID, description)

Описание: виды инвентаря для gym

Содержание:

* Сам инвентарь, как таковой – IG\_NAME\_ID
* Его описание - description

**Materials\_Sports\_ground**(MSG\_NAME\_ID, description)

Описание: виды материалов для sports\_ground

Содержание:

* Сам материал, как таковой – MSG\_NAME\_ID
* Его описание - description

**Types\_Shooting\_range**(TSR\_NAME\_ID, description)

Описание: типы выстрелов для shooting\_range

Содержание:

* Сам тип выстрелов, как таковой – TSR\_NAME\_ID
* Его описание - description

**Materials\_Hippodrome**(MH\_NAME\_ID, description)

Описание: виды материалов для hippodrome

Содержание:

* Сам материал, как таковой – MH\_NAME\_ID
* Его описание - description

**Materials\_Cycle\_track(**MCT\_NAME\_ID, description)

Описание: виды материалов для cycle\_track

Содержание:

* Сам материал, как таковой – MCT\_NAME\_ID
* Его описание - description

**Materials\_Mototrack**(MM\_NAME\_ID, description)

Описание: виды материалов для mototrack

Содержание:

* Сам материал, как таковой – MM\_NAME\_ID
* Его описание - description

TYPES\_TSI\_NAMES(TSI\_ID)

Описание: содержит названия типов структур

Содержание:

* Само название как таковое – TSI\_ID

**Определение ограничений целостности**

* У таблиц схожего предназначения, созданных как виды спортивных инфраструктур, должен быть отличимый друг от друга \*\_ID. Например у **Rowing\_channel.RC\_ID** небудет совпадений с **Autodrom.A\_ID**
* **Competition.**COM\_END\_REG\_DATE должен быть больше или равен **Members\_Competition.**MC\_REG\_DATE
* Спортсмен обязан состоять в клубе, за который выступал на тот момент, т.е. Members\_Club\_History.MCH\_START\_DATE <= Competition.COM\_START\_DATE и Members\_Club\_History.MCH\_END\_DATE >= Competition.COM\_END\_DATE или Members\_Club\_History.MCH\_END\_DATE это NULL
* Имена спортивных сооружений в таблице TYPES\_SPORTS должны совпадать с названиями таблиц схожего предназначения, созданных как виды спортивных инфраструктур

**Создание Талиц**

CREATE TABLE TYPES\_TSI\_NAMES(

TTN\_NAMES VARCHAR(120) PRIMARY KEY

)

CREATE TABLE Types\_Sports\_Infrastructures(

TSI\_ID INT PRIMARY KEY,

TSI\_NAME VARCHAR(120) NOT NULL,

TSI\_DESCRIPTION LONG

)

CREATE TABLE Sports\_Facility\_Information(

SFI\_ID INT PRIMARY KEY,

TSI\_ID INT NOT NULL,

SFI\_ADDR VARCHAR(256) NOT NULL,

SFI\_NAME VARCHAR(64) NOT NULL,

FOREIGN KEY(TSI\_ID) REFERENCES Types\_Sports\_Infrastructures(TSI\_ID)

)

CREATE TABLE Mototrack(

M\_ID INT PRIMARY KEY,

M\_TYPE VARCHAR(64) NOT NULL,

FOREIGN KEY(M\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Sports\_ground(

SG\_ID INT PRIMARY KEY,

SG\_MATERIAL VARCHAR(64) NOT NULL,

SG\_H\_SIZE INT NOT NULL,

SG\_W\_SIZE INT NOT NULL,

FOREIGN KEY(SG\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Swimming\_pool(

SP\_ID INT PRIMARY KEY,

SP\_H\_SIZE INT NOT NULL,

SP\_W\_SIZE INT NOT NULL,

SP\_L\_SIZE INT NOT NULL,

FOREIGN KEY(SP\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Cycle\_track(

CT\_ID INT PRIMARY KEY,

CT\_MATERIAL VARCHAR(64) NOT NULL,

CT\_LENGTH INT NOT NULL,

FOREIGN KEY(CT\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Court(

C\_ID INT PRIMARY KEY,

C\_TYPE VARCHAR(512) NOT NULL,

FOREIGN KEY(C\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Stadium(

S\_ID INT PRIMARY KEY,

S\_CAPASITY INT NOT NULL,

FOREIGN KEY(S\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Gym(

GYM\_ID INT PRIMARY KEY,

Gym\_INVENTORY VARCHAR(2000) NOT NULL,

FOREIGN KEY(GYM\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Rowing\_channel(

RC\_ID INT PRIMARY KEY,

RC\_H\_SIZE INT NOT NULL,

RC\_W\_SIZE INT NOT NULL,

RC\_L\_SIZE INT NOT NULL,

FOREIGN KEY(RC\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Shooting\_range(

SR\_ID INT PRIMARY KEY,

SR\_TYPE VARCHAR(512) NOT NULL,

FOREIGN KEY(SR\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Autodrom(

A\_ID INT PRIMARY KEY,

A\_CAPASITY INT NOT NULL,

FOREIGN KEY(A\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Ice\_rink(

IR\_ID INT PRIMARY KEY,

IR\_OPENNESS CHAR(1) CHECK (IR\_OPENNESS IN ('N','Y')) NOT NULL,

IR\_NATURAL CHAR(1) CHECK (IR\_NATURAL IN ('N','Y')) NOT NULL,

FOREIGN KEY(IR\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Hippodrome(

HP\_ID INT PRIMARY KEY,

HP\_TYPES VARCHAR(512) NOT NULL,

FOREIGN KEY(HP\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID)

)

CREATE TABLE Types\_Sports(

TS\_ID INT PRIMARY KEY,

TSI\_ID INT NOT NULL,

TS\_NAME VARCHAR(128) NOT NULL,

TS\_DESCRIPTION LONG,

FOREIGN KEY(TSI\_ID) REFERENCES Types\_Sports\_Infrastructures(TSI\_ID)

)

CREATE TABLE organizer(

ORG\_ID INT PRIMARY KEY,

ORG\_NAME VARCHAR(64) NOT NULL,

ORG\_TEL VARCHAR(16) NOT NULL,

ORG\_S\_MAIL VARCHAR(256) NOT NULL

)

CREATE TABLE Competition(

COM\_ID INT PRIMARY KEY,

SFI\_ID INT NOT NULL,

TS\_ID INT NOT NULL,

ORG\_ID INT NOT NULL,

COM\_NAME VARCHAR(256) NOT NULL,

COM\_START\_DATE DATE NOT NULL,

COM\_END\_DATE DATE,

COM\_END\_REG\_DATE DATE NOT NULL,

FOREIGN KEY(SFI\_ID) REFERENCES Sports\_Facility\_Information(SFI\_ID),

FOREIGN KEY(TS\_ID) REFERENCES Types\_Sports(TS\_ID),

FOREIGN KEY(ORG\_ID) REFERENCES organizer(ORG\_ID),

CONSTRAINT DATE\_CHECKER CHECK(COM\_START\_DATE <= COM\_END\_DATE AND COM\_END\_REG\_DATE <= COM\_START\_DATE)

)

CREATE TABLE Trainer(

T\_ID INT PRIMARY KEY,

T\_NAME VARCHAR(64) NOT NULL,

T\_Surname VARCHAR(64) NOT NULL,

T\_Patronymic VARCHAR(64) NOT NULL,

T\_TEL VARCHAR(16) NOT NULL,

T\_MAIL VARCHAR(256) NOT NULL

)

CREATE TABLE SPORTSMAN (

S\_ID INT PRIMARY KEY,

S\_NAME VARCHAR(64) NOT NULL,

S\_Surname VARCHAR(64) NOT NULL,

S\_Patronymic VARCHAR(64) NOT NULL,

S\_TEL VARCHAR(16) NOT NULL,

S\_MAIL VARCHAR(256) NOT NULL

)

CREATE TABLE Profile\_Info\_Trainer(

T\_ID INT NOT NULL,

TS\_ID INT NOT NULL,

PIT\_EXP\_MONTHS INT DEFAULT (0) CHECK(PIT\_EXP\_MONTHS >= 0) NOT NULL,

FOREIGN KEY(T\_ID) REFERENCES Trainer(T\_ID),

FOREIGN KEY(TS\_ID) REFERENCES Types\_Sports(TS\_ID),

CONSTRAINT TRAINER\_SPORT\_ID PRIMARY KEY (T\_ID,TS\_ID)

)

CREATE TABLE Trainer\_Sportsman\_HISTORY(

TSH\_ID INT PRIMARY KEY,

T\_ID INT NOT NULL,

TS\_ID INT NOT NULL,

S\_ID INT NOT NULL,

TSH\_START\_DATE DATE NOT NULL,

TSH\_END\_DATE DATE,

FOREIGN KEY(T\_ID) REFERENCES Trainer(T\_ID),

FOREIGN KEY(TS\_ID) REFERENCES Types\_Sports(TS\_ID),

FOREIGN KEY(S\_ID) REFERENCES Sportsman(S\_ID),

CONSTRAINT CHECKER\_DATE CHECK(TSH\_START\_DATE <= TSH\_END\_DATE OR TSH\_END\_DATE IS NULL)

)

CREATE TABLE Club(

CL\_ID INT PRIMARY KEY,

TS\_ID INT NOT NULL,

CL\_NAME VARCHAR(256) NOT NULL,

CL\_HAIR\_DATE DATE,

CL\_Founder VARCHAR(256) NOT NULL,

CL\_TEL VARCHAR(16) NOT NULL,

FOREIGN KEY(TS\_ID) REFERENCES Types\_Sports(TS\_ID)

)

CREATE TABLE Winners(

COM\_ID INT NOT NULL,

S\_ID INT NOT NULL,

W\_PLACE INT NOT NULL,

W\_DATE DATE NOT NULL,

FOREIGN KEY(COM\_ID) REFERENCES Competition(COM\_ID),

FOREIGN KEY(S\_ID) REFERENCES Sportsman(S\_ID),

CONSTRAINT SPORTSMEN\_Competition\_ID PRIMARY KEY (COM\_ID,S\_ID)

)

CREATE TABLE Members\_Competition(

COM\_ID INT NOT NULL,

S\_ID INT NOT NULL,

CL\_ID INT NOT NULL,

MC\_REG\_DATE DATE NOT NULL,

FOREIGN KEY(COM\_ID) REFERENCES Competition(COM\_ID),

FOREIGN KEY(S\_ID) REFERENCES Sportsman(S\_ID),

FOREIGN KEY(CL\_ID) REFERENCES Club(CL\_ID),

CONSTRAINT SPORTSMEN\_Competition\_ID\_1 PRIMARY KEY (COM\_ID,S\_ID)

)

CREATE TABLE Members\_Club\_History(

MCH\_ID INT PRIMARY KEY,

S\_ID INT NOT NULL,

CL\_ID INT NOT NULL,

MCH\_START\_DATE DATE NOT NULL,

MCH\_END\_DATE DATE,

FOREIGN KEY(S\_ID) REFERENCES Sportsman(S\_ID),

FOREIGN KEY(CL\_ID) REFERENCES Club(CL\_ID),

CONSTRAINT CHECKER\_DATE\_1 CHECK(MCH\_START\_DATE <= MCH\_END\_DATE OR MCH\_END\_DATE IS NULL)

)

CREATE TABLE Materials\_Mototrack(

MM\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE Materials\_Cycle\_track(

MCT\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE Materials\_Hippodrome(

MH\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE Types\_Shooting\_range(

TSR\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE Materials\_Sports\_ground(

MSG\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE INVENTORY\_GYM(

IG\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE Materials\_Court(

MC\_NAME\_ID VARCHAR(256) PRIMARY KEY,

description LONG

)

CREATE TABLE Ranks(

RANK\_ID INT PRIMARY KEY,

S\_ID INT NOT NULL,

TS\_ID INT NOT NULL,

RANK\_SPORT INT CHECK (RANK\_SPORT >= 0) NOT NULL,

RANK\_NAME VARCHAR(256) NOT NULL,

RANK\_DESCRIPTION LONG,

RANK\_DATE DATE NOT NULL,

FOREIGN KEY(S\_ID) REFERENCES Sportsman(S\_ID),

FOREIGN KEY(TS\_ID) REFERENCES Types\_Sports(TS\_ID)

)

**Update:**

COMPETITION:

ALTER TABLE COMPETITION

ADD COM\_START\_REG\_DATE DATE NOT NULL

ALTER TABLE COMPETITION

ADD CONSTRAINT REG\_CHECK CHECK(COM\_START\_REG\_DATE <= COM\_END\_REG\_DATE )

GYM:

ALTER TABLE GYM

DROP PRIMARY KEY

ALTER TABLE GYM

ADD CONSTRAINT KEY\_INV\_AND\_OBJ PRIMARY KEY (GYM\_ID, COUNT\_GYM\_INV)

Profile\_Info\_Trainer:

ALTER TABLE PROFILE\_INFO\_TRAINER

ADD PIT\_TRAINER\_ACTIVE CHAR(1) CHECK (PIT\_TRAINER\_ACTIVE IN ('N','Y')) NOT NULL

ALTER TABLE Types\_Sports\_Infrastructures

ADD CONSTRAINT type\_Infrastructures\_1 FOREIGN KEY(TSI\_NAME) REFERENCES TYPES\_TSI\_NAMES(TTN\_NAMES)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Update foreing keys:**

ALTER TABLE Mototrack

ADD CONSTRAINT type\_specifier\_7 FOREIGN KEY(M\_TYPE) REFERENCES Materials\_Mototrack(MM\_NAME\_ID)

ALTER TABLE Cycle\_track

ADD CONSTRAINT type\_specifier\_6 FOREIGN KEY(CT\_MATERIAL) REFERENCES Materials\_Cycle\_track(MCT\_NAME\_ID)

ALTER TABLE Hippodrome

ADD CONSTRAINT type\_specifier\_5 FOREIGN KEY(HP\_TYPES) REFERENCES Materials\_Hippodrome(MH\_NAME\_ID)

ALTER TABLE Shooting\_range

ADD CONSTRAINT type\_specifier\_4 FOREIGN KEY(SR\_TYPE) REFERENCES Types\_Shooting\_range(TSR\_NAME\_ID)

ALTER TABLE Sports\_ground

ADD CONSTRAINT type\_specifier\_3 FOREIGN KEY(SG\_MATERIAL) REFERENCES Materials\_Sports\_ground(MSG\_NAME\_ID)

ALTER TABLE Gym

ADD CONSTRAINT type\_specifier\_2 FOREIGN KEY(Gym\_INVENTORY) REFERENCES INVENTORY\_GYM(IG\_NAME\_ID)

ALTER TABLE Court

ADD CONSTRAINT type\_specifier\_1 FOREIGN KEY(C\_TYPE) REFERENCES Materials\_Court(MC\_NAME\_ID)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Update varchar to varchar(256):**

ALTER TABLE Mototrack

MODIFY M\_TYPE VARCHAR(256);

ALTER TABLE Cycle\_track

MODIFY CT\_MATERIAL VARCHAR(256);

ALTER TABLE Hippodrome

MODIFY HP\_TYPES VARCHAR(256);

ALTER TABLE Shooting\_range

MODIFY SR\_TYPE VARCHAR(256);

ALTER TABLE Sports\_ground

MODIFY SG\_MATERIAL VARCHAR(256);

ALTER TABLE Gym

MODIFY Gym\_INVENTORY VARCHAR(256);

ALTER TABLE Court

MODIFY C\_TYPE VARCHAR(256);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Update constraint modify >=0:**

ALTER TABLE Sports\_ground

MODIFY SG\_H\_SIZE CHECK(SG\_H\_SIZE >= 0);

ALTER TABLE Sports\_ground

MODIFY SG\_W\_SIZE CHECK(SG\_W\_SIZE >= 0);

ALTER TABLE Cycle\_track

MODIFY CT\_LENGTH CHECK(CT\_LENGTH >= 0);

ALTER TABLE Stadium

MODIFY S\_CAPASITY CHECK(S\_CAPASITY >= 0 AND S\_CAPASITY <= 200000);

ALTER TABLE Autodrom

MODIFY A\_CAPASITY CHECK(A\_CAPASITY >= 0);

ALTER TABLE Rowing\_channel

MODIFY RC\_H\_SIZE CHECK(RC\_H\_SIZE >= 0);

ALTER TABLE Rowing\_channel

MODIFY RC\_W\_SIZE CHECK(RC\_W\_SIZE >= 0);

ALTER TABLE Rowing\_channel

MODIFY RC\_L\_SIZE CHECK(RC\_L\_SIZE >= 0);

ALTER TABLE Swimming\_pool

MODIFY SP\_H\_SIZE CHECK(SP\_H\_SIZE >= 0);

ALTER TABLE Swimming\_pool

MODIFY SP\_W\_SIZE CHECK(SP\_W\_SIZE >= 0);

ALTER TABLE Swimming\_pool

MODIFY SP\_L\_SIZE CHECK(SP\_L\_SIZE >= 0);

ALTER TABLE Winners

MODIFY W\_PLACE CHECK(W\_PLACE >= 0);

ALTER TABLE Profile\_Info\_Trainer

MODIFY PIT\_EXP\_MONTHS CHECK(PIT\_EXP\_MONTHS >= 0);

1. **SQL-запросы**
2. **Получить перечень спортивных сооружений указанного типа в целом или удовлетворяющих заданным характеристикам (например, стадионы, вмещающие не менее указанного числа зрителей).**

SELECT \*

FROM Sports\_Facility\_Information sfi

RIGHT JOIN Stadium s on sfi.SFI\_ID = s.S\_ID

WHERE s.S\_CAPASITY >= 50000

1. **Получить перечень спортивных сооружений указанного типа в целом или удовлетворяющих заданным характеристикам (например, стадионы, вмещающие не менее указанного числа зрителей).**

SELECT s.S\_NAME as ИМЯ, s.S\_SURNAME as фамилия, s.S\_PATRONYMIC as отчество, r.RANK\_SPORT AS разряд,r.RANK\_NAME as Наименование , ts.TS\_NAME as Вид\_спорта

FROM Sportsman s

RIGHT JOIN Ranks r on s.S\_ID = r.S\_ID

RIGHT JOIN Types\_Sports ts on ts.TS\_ID = r.TS\_ID

WHERE r.RANK\_SPORT >= 2 AND ts.TS\_NAME = 'футбол'

1. **Получить список спортсменов, тренирующихся у некого тренера в целом либо не ниже определенного разряда.**

SELECT s.S\_NAME as Имя\_СПОРТСМ, s.S\_SURNAME as Фам\_СПОРТСМ, s.S\_PATRONYMIC as ОТЧ\_СПОРТСМ, tr.T\_NAME as Имя\_ТРЕН, tr.T\_SURNAME as Фам\_ТРЕН, tr.T\_PATRONYMIC as ОТЧ\_ТРЕН, TS\_NAME as СПОРТ

FROM Sportsman s

RIGHT JOIN Trainer\_Sportsman\_HISTORY tsh on s.S\_ID = tsh.S\_ID

RIGHT JOIN Trainer tr on tr.T\_ID = tsh.T\_ID

RIGHT JOIN Types\_Sports ts on ts.TS\_ID = tsh.TS\_ID

WHERE TSH\_END\_DATE IS NULL AND tr.T\_NAME = 'Ирина' AND tr.T\_SURNAME = 'Моторкина' AND tr.T\_PATRONYMIC = 'Викторовна'

1. **Получить список спортсменов, занимающихся более чем одним видом спорта с указанием этих видов спорта.**

WITH SPORTS\_SPORTSMEN AS(SELECT count(r.TS\_ID) OVER(partition by s.S\_ID ORDER BY s.S\_ID) as count\_sports, s.S\_ID, ts.TS\_ID

FROM Sportsman s

RIGHT JOIN Ranks r on s.S\_ID = r.S\_ID

RIGHT JOIN Types\_Sports ts on ts.TS\_ID = r.TS\_ID)

select \*

from SPORTS\_SPORTSMEN ss

JOIN Sportsman s on s.S\_ID = ss.S\_ID

JOIN Types\_Sports ts on ts.TS\_ID = ss.TS\_ID

where ss.COUNT\_SPORTS >= 2

1. **Получить список тренеров указанного спортсмена.**

select DISTINCT T\_NAME, T\_SURNAME, T\_PATRONYMIC

from Trainer t

JOIN Trainer\_Sportsman\_HISTORY ts on T.T\_ID = ts.T\_ID

JOIN Sportsman s on s.S\_ID = ts.S\_ID

where s.S\_NAME = 'Климент' and s.S\_SURNAME = 'Колесников' and s.S\_PATRONYMIC = 'Андреевич'

1. **Получить список тренеров указанного спортсмена.**

select COM\_NAME, COM\_START\_DATE, COM\_END\_DATE, ORG\_NAME

from Competition c

left JOIN organizer o on o.ORG\_ID = c.ORG\_ID

where (COM\_START\_DATE >= '19.05.23' and COM\_END\_DATE <= '21.06.23') or ORG\_NAME = 'ВТБ'

1. **Получить список тренеров указанного спортсмена.**

select COM\_NAME as Соревнование, S\_NAME as ИМЯ, S\_SURNAME as Фамилия, S\_PATRONYMIC as Отчество, W\_PLACE as ПРИЗОВОЕ\_МЕСТО

from Competition c

JOIN Winners w on w.COM\_ID = c.COM\_ID

JOIN Sportsman s on s.S\_ID = w.S\_ID

WHERE COM\_NAME = 'Кубок района'

1. **Получить список тренеров указанного спортсмена.**

select TSI\_NAME as Место\_Пров, COM\_NAME as соревнование

from Competition c

LEFT JOIN Types\_Sports ts on ts.TS\_ID = c.TS\_ID

LEFT JOIN Types\_Sports\_Infrastructures tsi on tsi.TSI\_ID = ts.TSI\_ID

where TSI\_NAME = 'Ice\_rink'

1. **Получить список тренеров указанного спортсмена.**

select DISTINCT count(s.S\_ID) OVER(partition by cl.CL\_ID ORDER BY cl.CL\_ID) as count\_sportsmens, cl.CL\_ID, cl.CL\_NAME

from Members\_Competition mc

LEFT JOIN Club cl on cl.CL\_ID = mc.CL\_ID

LEFT JOIN Sportsman s on s.S\_ID = mc.S\_ID

LEFT JOIN Competition c on c.COM\_ID = mc.COM\_ID

WHERE c.COM\_START\_DATE >= '20.05.23' AND c.COM\_END\_DATE <= '21.06.23'

1. Получить список тренеров указанного спортсмена.

select T\_NAME, T\_SURNAME, T\_PATRONYMIC, TS\_NAME

from Trainer t

join Profile\_Info\_Trainer pit on t.T\_ID = pit.T\_ID

left join Types\_Sports ts on ts.TS\_ID = pit.TS\_ID

where ts.TS\_NAME = 'плавание'

1. **Получить список тренеров указанного спортсмена.**

WITH count\_competition AS(select count(c.COM\_ID) OVER(partition by s.S\_ID ORDER BY s.S\_ID) as count\_com, s.S\_ID

from Competition c

JOIN Members\_Competition mc on c.COM\_ID = mc.COM\_ID

RIGHT JOIN Sportsman s on s.S\_ID = mc.S\_ID

WHERE COM\_START\_DATE >= '25.05.23' and COM\_END\_DATE <= '20.06.23')

select S\_NAME, S\_SURNAME, S\_PATRONYMIC

from count\_competition cc

right join Sportsman s on s.S\_ID = cc.S\_ID

WHERE COUNT\_COM is null OR COUNT\_COM = 0

1. **Получить список тренеров указанного спортсмена.**

Select DISTINCT count(c.COM\_ID) OVER(partition by o.ORG\_ID ORDER BY o.ORG\_ID) as count\_com, ORG\_NAME

from organizer o

left join Competition c on c.ORG\_ID = o.ORG\_ID

WHERE COM\_START\_DATE >= '20.05.23' AND (COM\_START\_DATE <= '20.06.23' OR COM\_END\_DATE is NULL)

1. **Получить список тренеров указанного спортсмена.**

Select SFI\_NAME as Название\_Соор, COM\_NAME as Название\_сор, COM\_START\_DATE as Дата\_Начала, COM\_END\_DATE as Дата\_Окон

from Sports\_Facility\_Information sfi

left join Competition c on c.SFI\_ID = sfi.SFI\_ID

WHERE COM\_START\_DATE >= '01.03.23' AND (COM\_END\_DATE <= '20.03.23' or COM\_END\_DATE IS NULL)

1. **Триггеры и Хранимые процедуры**

* **1.** Проверка, чтобы участник соревнования числился в клубе в момент регистрации, иначе вызываем ошибку.

**CREATE OR REPLACE** TRIGGER Is\_the\_athlete\_in\_club

BEFORE INSERT OR UPDATE ON Members\_Competition

FOR EACH ROW

DECLARE

start\_date DATE;

end\_date DATE;

BEGIN

SELECT MCH\_END\_DATE INTO end\_date

FROM Members\_Club\_History

WHERE S\_ID = :NEW.S\_ID AND CL\_ID = :NEW.CL\_ID;

SELECT MCH\_START\_DATE INTO start\_date

FROM Members\_Club\_History

WHERE S\_ID = :NEW.S\_ID AND CL\_ID = :NEW.CL\_ID;

IF :NEW.MC\_REG\_DATE > end\_date AND :NEW.MC\_REG\_DATE < start\_date AND end\_date IS NOT NULL THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Registration not allowed after club membership end date');

END IF;

END;

* **2.** Проверяем, есть ли ID мототрека в других аналогичных таблицах. Если есть вызываем исключение. Далее по 13 триггер включительно логика сохраняется.

**CREATE OR REPLACE** TRIGGER Mototrack\_ID\_trigger

BEFORE INSERT OR UPDATE ON Mototrack

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.M\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.M\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.M\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.M\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.M\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.M\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.M\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.M\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.M\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.M\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.M\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **3...**

**CREATE OR REPLACE TRIGGER** Sports\_ground\_ID\_trigger

BEFORE INSERT OR UPDATE ON Sports\_ground

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.SG\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.SG\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.SG\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.SG\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.SG\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.SG\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.SG\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.SG\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.SG\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.SG\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.SG\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **4…**

**CREATE OR REPLACE TRIGGER** Swimming\_pool\_ID\_trigger

BEFORE INSERT OR UPDATE ON Swimming\_pool

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.SP\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.SP\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.SP\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.SP\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.SP\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.SP\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.SP\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.SP\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.SP\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.SP\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.SP\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **5…**

**CREATE OR REPLACE TRIGGER** Cycle\_track\_ID\_trigger

BEFORE INSERT OR UPDATE ON Cycle\_track

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.CT\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.CT\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.CT\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.CT\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.CT\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.CT\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.CT\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.CT\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.CT\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.CT\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.CT\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **6…**

**CREATE OR REPLACE TRIGGER** Court\_ID\_trigger

BEFORE INSERT OR UPDATE ON Court

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.C\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.C\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.C\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.C\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.C\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.C\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.C\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.C\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.C\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.C\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.C\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **7…**

**CREATE OR REPLACE TRIGGER** Stadium\_ID\_trigger

BEFORE INSERT OR UPDATE ON Stadium

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.S\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.S\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.S\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.S\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.S\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.S\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.S\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.S\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.S\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.S\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.S\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **8…**

**CREATE OR REPLACE TRIGGER** Gym\_ID\_trigger

BEFORE INSERT OR UPDATE ON Gym

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.GYM\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.GYM\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.GYM\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.GYM\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.GYM\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.GYM\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.GYM\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.GYM\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.GYM\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.GYM\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.GYM\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **9…**

**CREATE OR REPLACE TRIGGER** Rowing\_channel\_ID\_trigger

BEFORE INSERT OR UPDATE ON Rowing\_channel

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.RC\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.RC\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.RC\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.RC\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.RC\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.RC\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.RC\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.RC\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.RC\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.RC\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.RC\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **10…**

**CREATE OR REPLACE TRIGGER** Shooting\_range\_ID\_trigger

BEFORE INSERT OR UPDATE ON Shooting\_range

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.SR\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.SR\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.SR\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.SR\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.SR\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.SR\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.SR\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.SR\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.SR\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.SR\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.SR\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **11…**

**CREATE OR REPLACE TRIGGER** Autodrom\_ID\_trigger

BEFORE INSERT OR UPDATE ON Autodrom

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.A\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.A\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.A\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.A\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.A\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.A\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.A\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.A\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.A\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.A\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.A\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **12…**

**CREATE OR REPLACE TRIGGER** Ice\_rink\_ID\_trigger

BEFORE INSERT OR UPDATE ON Ice\_rink

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.IR\_ID

UNION

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.IR\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.IR\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.IR\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.IR\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.IR\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.IR\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.IR\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.IR\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.IR\_ID

UNION

SELECT HP\_ID FROM Hippodrome WHERE HP\_ID = :NEW.IR\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **13…**

**CREATE OR REPLACE TRIGGER** Hippodrome\_ID\_trigger

BEFORE INSERT OR UPDATE ON Hippodrome

FOR EACH ROW

DECLARE

count\_id NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_id FROM (

SELECT SG\_ID FROM Sports\_ground WHERE SG\_ID = :NEW.HP\_ID

UNION

SELECT SP\_ID FROM Swimming\_pool WHERE SP\_ID = :NEW.HP\_ID

UNION

SELECT CT\_ID FROM Cycle\_track WHERE CT\_ID = :NEW.HP\_ID

UNION

SELECT C\_ID FROM Court WHERE C\_ID = :NEW.HP\_ID

UNION

SELECT S\_ID FROM Stadium WHERE S\_ID = :NEW.HP\_ID

UNION

SELECT GYM\_ID FROM Gym WHERE GYM\_ID = :NEW.HP\_ID

UNION

SELECT RC\_ID FROM Rowing\_channel WHERE RC\_ID = :NEW.HP\_ID

UNION

SELECT SR\_ID FROM Shooting\_range WHERE SR\_ID = :NEW.HP\_ID

UNION

SELECT A\_ID FROM Autodrom WHERE A\_ID = :NEW.HP\_ID

UNION

SELECT IR\_ID FROM Ice\_rink WHERE IR\_ID = :NEW.HP\_ID

UNION

SELECT M\_ID FROM Mototrack WHERE M\_ID = :NEW.HP\_ID

);

IF count\_id > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ID already exists in other tables!');

END IF;

END;

* **14.** Проверка того, чтобы дата регистрации участника, находилась в диапазоне дат начала и окончания регистрации в соревновании

**CREATE OR REPLACE TRIGGER** check\_mc

BEFORE INSERT OR UPDATE ON Members\_Competition

FOR EACH ROW

DECLARE

com\_start\_reg\_date DATE;

com\_end\_reg\_date DATE;

BEGIN

SELECT COM\_START\_REG\_DATE, COM\_END\_REG\_DATE INTO com\_start\_reg\_date, com\_end\_reg\_date

FROM Competition

WHERE COM\_ID = :new.COM\_ID;

IF (:new.MC\_REG\_DATE < com\_start\_reg\_date OR :new.MC\_REG\_DATE > com\_end\_reg\_date) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Registration date is not within competition registration period');

END IF;

END;

* **15.** Если мы не указываем дату начала и окончания регистрации в клубе, то мы дату начала выбираем текущую, а дату окончания NULL(можно указать дату окончания без даты начала, но обязательно нужно, чтоб дата окончания была больше текущего времени)

**CREATE OR REPLACE TRIGGER** INSERT\_HISTORY\_TRIGGER

BEFORE INSERT ON Members\_Club\_History

FOR EACH ROW

DECLARE

MAX\_MCH\_ID NUMBER;

BEGIN

SELECT NVL(MAX(MCH\_ID),0) + 1 INTO MAX\_MCH\_ID FROM Members\_Club\_History;

:NEW.MCH\_ID := MAX\_MCH\_ID;

IF :NEW.MCH\_START\_DATE IS NULL THEN

:NEW.MCH\_START\_DATE := SYSDATE;

END IF;

IF :NEW.MCH\_END\_DATE IS NULL THEN

:NEW.MCH\_END\_DATE := NULL;

END IF;

END;

* **16.** Проверка того, что спортсмен не может быть добавлен в клуб, если он уже состоит в нём же. Если данное событие имеется вызываем исключение.

**CREATE OR REPLACE TRIGGER** MCH\_CHECK\_END\_DATE\_Trigger

BEFORE INSERT ON Members\_Club\_History

FOR EACH ROW

DECLARE

cnt INTEGER;

BEGIN

SELECT COUNT(\*) INTO cnt

FROM Members\_Club\_History

WHERE S\_ID = :NEW.S\_ID AND CL\_ID = :NEW.CL\_ID AND MCH\_END\_DATE IS NULL;

IF cnt > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'User cannot add record with same S\_ID and CL\_ID combination while MCH\_END\_DATE is NULL');

END IF;

END;

* **17.**  Проверяем, если спортсмен, клуб и дата начала регистрации остались неизменны, а так же дата окончания регистрации была и осталась NULL, тогда мы меняем NULL на текущую дату.

**CREATE OR REPLACE TRIGGER** update\_end\_date\_trigger

BEFORE UPDATE ON Members\_Club\_History

FOR EACH ROW

BEGIN

IF :old.S\_ID = :new.S\_ID

AND :old.CL\_ID = :new.CL\_ID

AND :old.MCH\_START\_DATE = :new.MCH\_START\_DATE

AND (:old.MCH\_END\_DATE IS NULL AND :new.MCH\_END\_DATE IS NULL) THEN

:new.MCH\_END\_DATE := SYSDATE;

END IF;

END;

**Пример:**

UPDATE Members\_Club\_History

SET MCH\_START\_DATE = MCH\_START\_DATE

WHERE S\_ID = 1

AND CL\_ID = 2

AND MCH\_START\_DATE = TO\_DATE('2021-08-01', 'YYYY-MM-DD')

AND MCH\_END\_DATE IS NULL;

Процедура автоматического заполнения конкретных единиц спортивных сооружения

**CREATE OR REPLACE PROCEDURE** insert\_into\_related\_table (

ti\_name IN VARCHAR2,

sfi\_addr IN VARCHAR2,

sfi\_name IN VARCHAR2,

sg\_material IN VARCHAR2 DEFAULT NULL, -- параметры для Sports\_ground

sg\_h\_size IN INT DEFAULT NULL,

sg\_w\_size IN INT DEFAULT NULL,

sp\_h\_size IN INT DEFAULT NULL, -- параметры для Swimming\_pool

sp\_w\_size IN INT DEFAULT NULL,

sp\_l\_size IN INT DEFAULT NULL,

ct\_material IN VARCHAR2 DEFAULT NULL, -- параметры для Cycle\_track

ct\_length IN INT DEFAULT NULL,

c\_type IN VARCHAR2 DEFAULT NULL, -- параметры для Court

s\_capasity IN INT DEFAULT NULL, -- параметры для Stadium

gym\_inventory IN SYS.ODCIVARCHAR2LIST DEFAULT NULL, -- параметры для Gym

ir\_openness IN CHAR DEFAULT NULL, -- параметры для Ice\_rink

ir\_natural IN CHAR DEFAULT NULL,

hp\_types IN VARCHAR2 DEFAULT NULL -- параметры для Hippodrome

) AS

sfi\_id NUMBER;

tsi\_id NUMBER;

count\_gym\_inv NUMBER := 0;

BEGIN

-- находим tsi\_id по имени из таблицы Types\_Sports\_Infrastructures

SELECT TSI\_ID INTO tsi\_id FROM Types\_Sports\_Infrastructures WHERE TSI\_NAME = ti\_name;

-- находим максимальный sfi\_id из таблицы Sports\_Facility\_Information

SELECT NVL(MAX(SFI\_ID), 0) + 1 INTO sfi\_id FROM Sports\_Facility\_Information;

-- вставляем новую запись в таблицу Sports\_Facility\_Information

INSERT INTO Sports\_Facility\_Information (SFI\_ID, TSI\_ID, SFI\_ADDR, SFI\_NAME)

VALUES (sfi\_id, tsi\_id, sfi\_addr, sfi\_name);

-- в зависимости от значения tsi\_name, вставляем запись в соответствующую таблицу

CASE ti\_name

WHEN 'Mototrack' THEN

INSERT INTO Mototrack (M\_ID, M\_TYPE) VALUES (sfi\_id, NULL);

WHEN 'Sports\_ground' THEN

INSERT INTO Sports\_ground (SG\_ID, SG\_MATERIAL, SG\_H\_SIZE, SG\_W\_SIZE)

VALUES (sfi\_id, sg\_material, sg\_h\_size, sg\_w\_size);

WHEN 'Swimming\_pool' THEN

INSERT INTO Swimming\_pool (SP\_ID, SP\_H\_SIZE, SP\_W\_SIZE, SP\_L\_SIZE)

VALUES (sfi\_id, sp\_h\_size, sp\_w\_size, sp\_l\_size);

WHEN 'Cycle\_track' THEN

INSERT INTO Cycle\_track (CT\_ID, CT\_MATERIAL, CT\_LENGTH)

VALUES (sfi\_id, ct\_material, ct\_length);

WHEN 'Court' THEN

INSERT INTO Court (C\_ID, C\_TYPE) VALUES (sfi\_id, c\_type);

WHEN 'Stadium' THEN

INSERT INTO Stadium (S\_ID, S\_CAPASITY) VALUES (sfi\_id, s\_capasity);

WHEN 'Gym' THEN

SELECT MAX(COUNT\_GYM\_INV) INTO count\_gym\_inv FROM Gym WHERE GYM\_ID = sfi\_id;

IF count\_gym\_inv IS NULL THEN count\_gym\_inv := 0; END IF;

IF gym\_inventory IS NULL OR gym\_inventory.COUNT = 0 THEN -- проверяем что массив не пустой

RAISE\_APPLICATION\_ERROR (-20001, 'Error: Gym\_INVENTORY is empty');

END IF;

FOR i IN 1..gym\_inventory.COUNT LOOP -- перебираем массив с Gym\_INVENTORY и добавляем элементы в таблицу Gym

count\_gym\_inv := count\_gym\_inv + 1;

INSERT INTO Gym (GYM\_ID, COUNT\_GYM\_INV, GYM\_INVENTORY) VALUES (sfi\_id, count\_gym\_inv, gym\_inventory(i));

END LOOP;

WHEN 'Rowing\_channel' THEN

INSERT INTO Rowing\_channel (RC\_ID, RC\_H\_SIZE, RC\_W\_SIZE, RC\_L\_SIZE)

VALUES (sfi\_id, sp\_h\_size, sp\_w\_size, sp\_l\_size);

WHEN 'Shooting\_range' THEN

INSERT INTO Shooting\_range (SR\_ID, SR\_TYPE) VALUES (sfi\_id, NULL);

WHEN 'Autodrom' THEN

INSERT INTO Autodrom (A\_ID, A\_CAPASITY) VALUES (sfi\_id, s\_capasity);

WHEN 'Ice\_rink' THEN

INSERT INTO Ice\_rink (IR\_ID, IR\_OPENNESS, IR\_NATURAL) VALUES (sfi\_id, ir\_openness, ir\_natural);

WHEN 'Hippodrome' THEN

INSERT INTO Hippodrome (HP\_ID, HP\_TYPES) VALUES (sfi\_id, hp\_types);

END CASE;

END;

END;

**Пример:**  
DECLARE

inv SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST('велотренажеры', 'эллиптические тренажеры', 'беговые дорожки');

BEGIN

insert\_into\_related\_table(

ti\_name => 'Gym',

sfi\_addr => 'ул. Спортивная, 10',

sfi\_name => 'Фитнес-центр "Спорт"',

gym\_inventory => inv

);

END;

1. **Разработка приложения клиента**

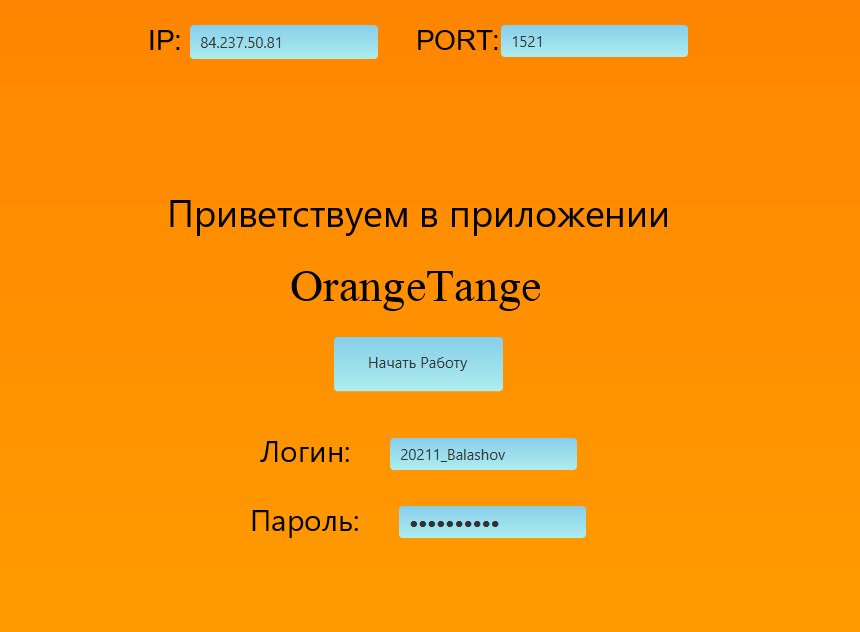
Ссылка на проект: https://github.com/DivineSilverWolf/6semDB/tree/db

В ходе разработки данного приложения были использованы и реализованы в полной мере следующие технологии:

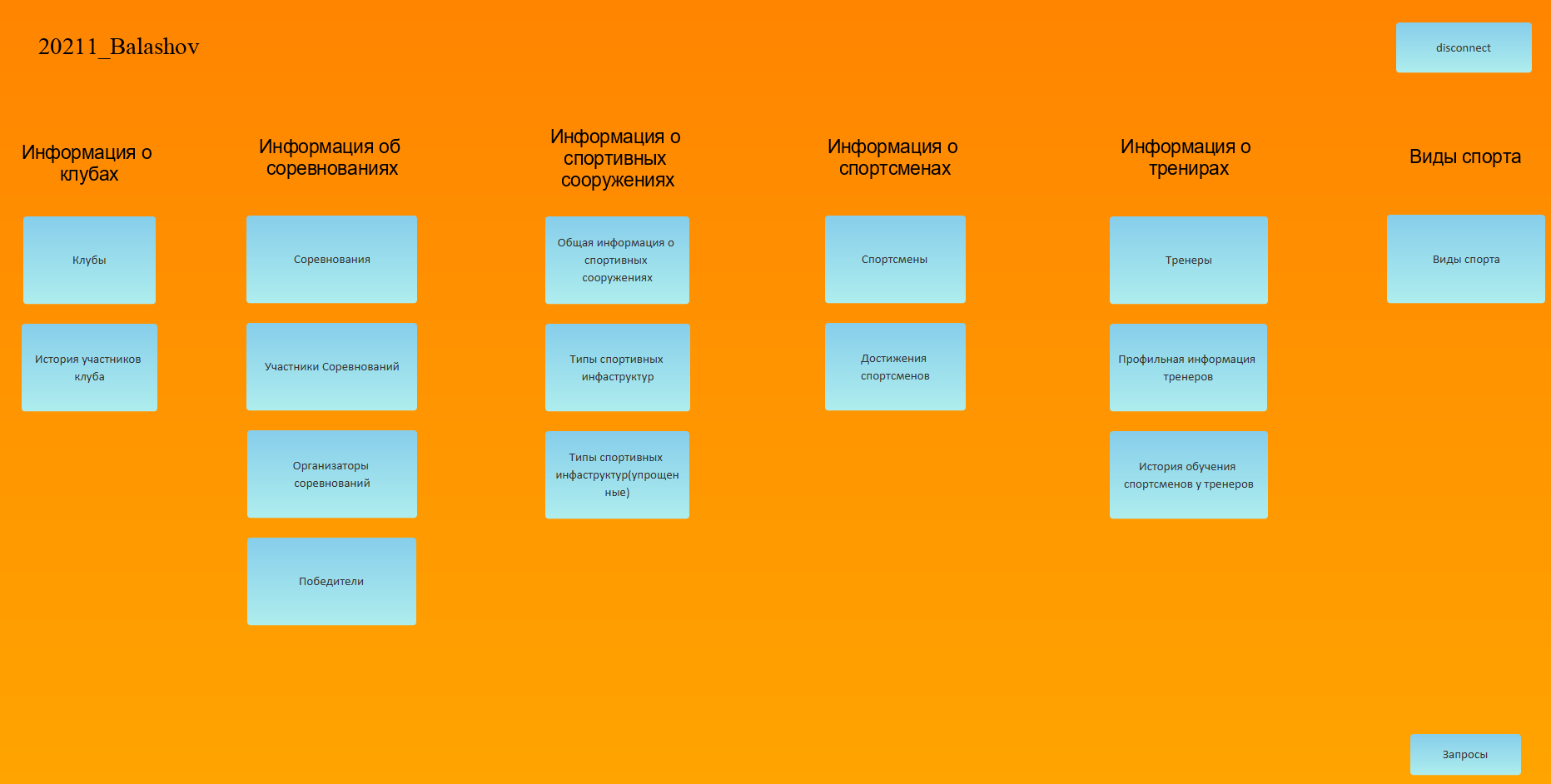
1. **Lombok** - это библиотека Java, которая упрощает разработку программного обеспечения, уменьшая объем написанного кода.
2. **ojdbc8** - это JDBC (Java Database Connectivity) драйвер, предназначенный для взаимодействия между приложениями, написанными на языке Java, и СУБД (системами управления базами данных) Oracle версии 8 и выше.
3. **java.sql** - это пакет (библиотека) Java, предназначенный для работы с базами данных. Он содержит набор классов и интерфейсов, которые позволяют Java-приложениям подключаться к различным СУБД (системам управления базами данных) и выполнять операции чтения, записи и обновления данных.
4. **javafx.fxml** - это пакет (библиотека) в JavaFX, предназначенный для отделения пользовательского интерфейса от логики приложения.
5. **javafx.controls** - это пакет (библиотека) в JavaFX, предоставляющий графические элементы для создания пользовательского интерфейса. Он содержит классы и интерфейсы для создания таких элементов, как кнопки, текстовые поля, выпадающие списки, таблицы, деревья и многое другое.

В данном проекте были реализованы следующие таблицы:

1. Вход:



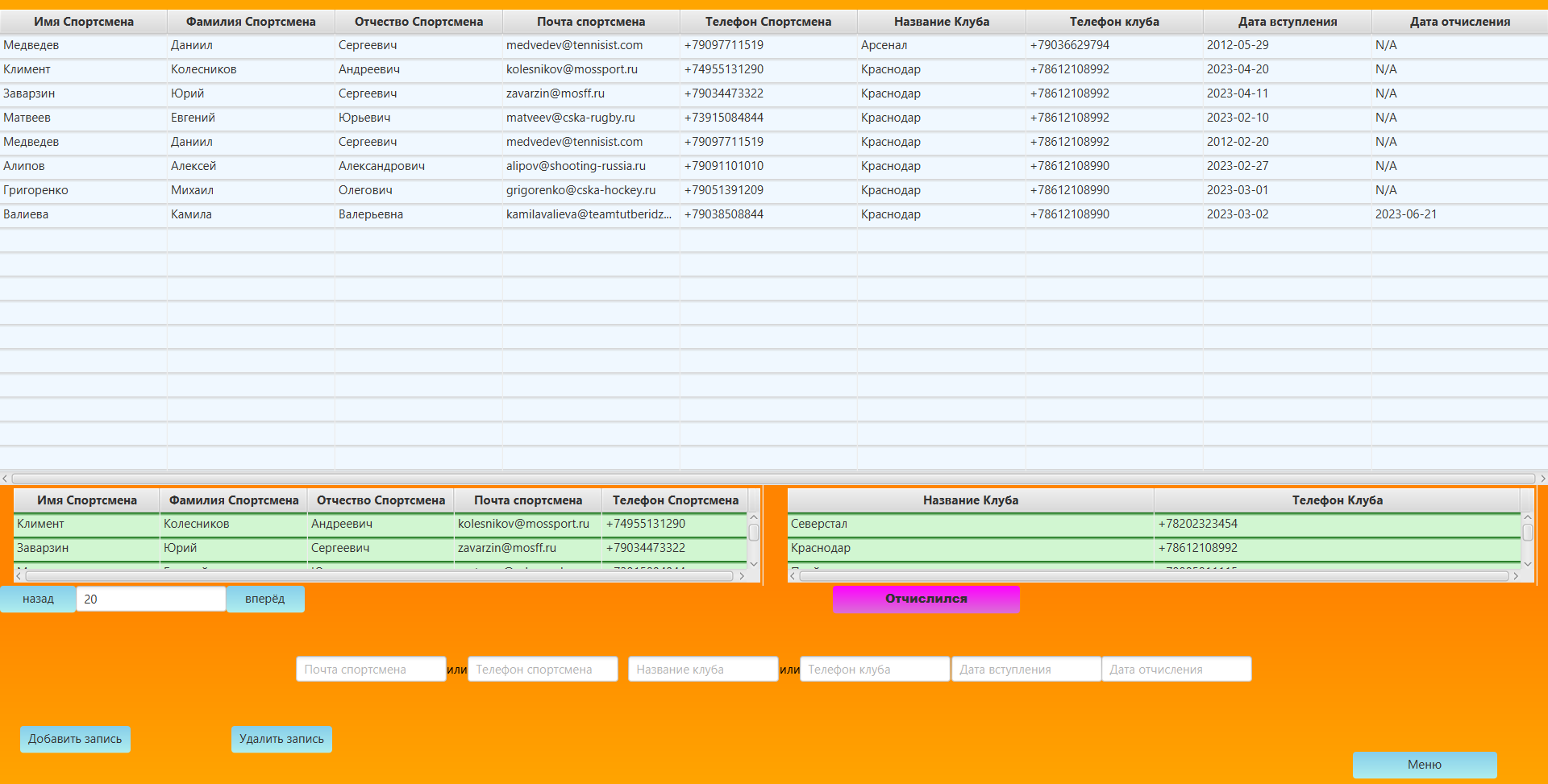
1. Меню:



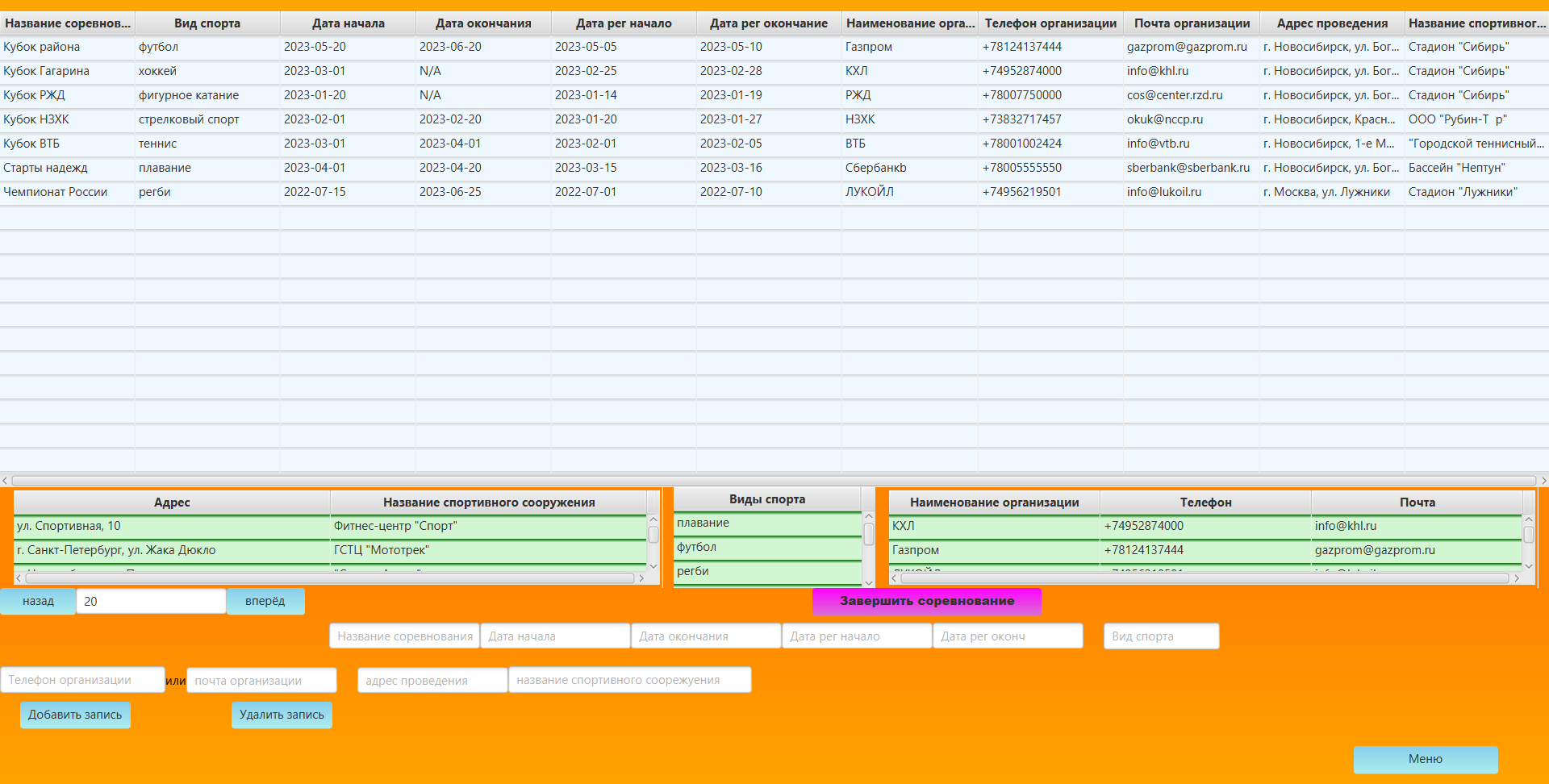
1. Клубы:



1. История участников клуба:



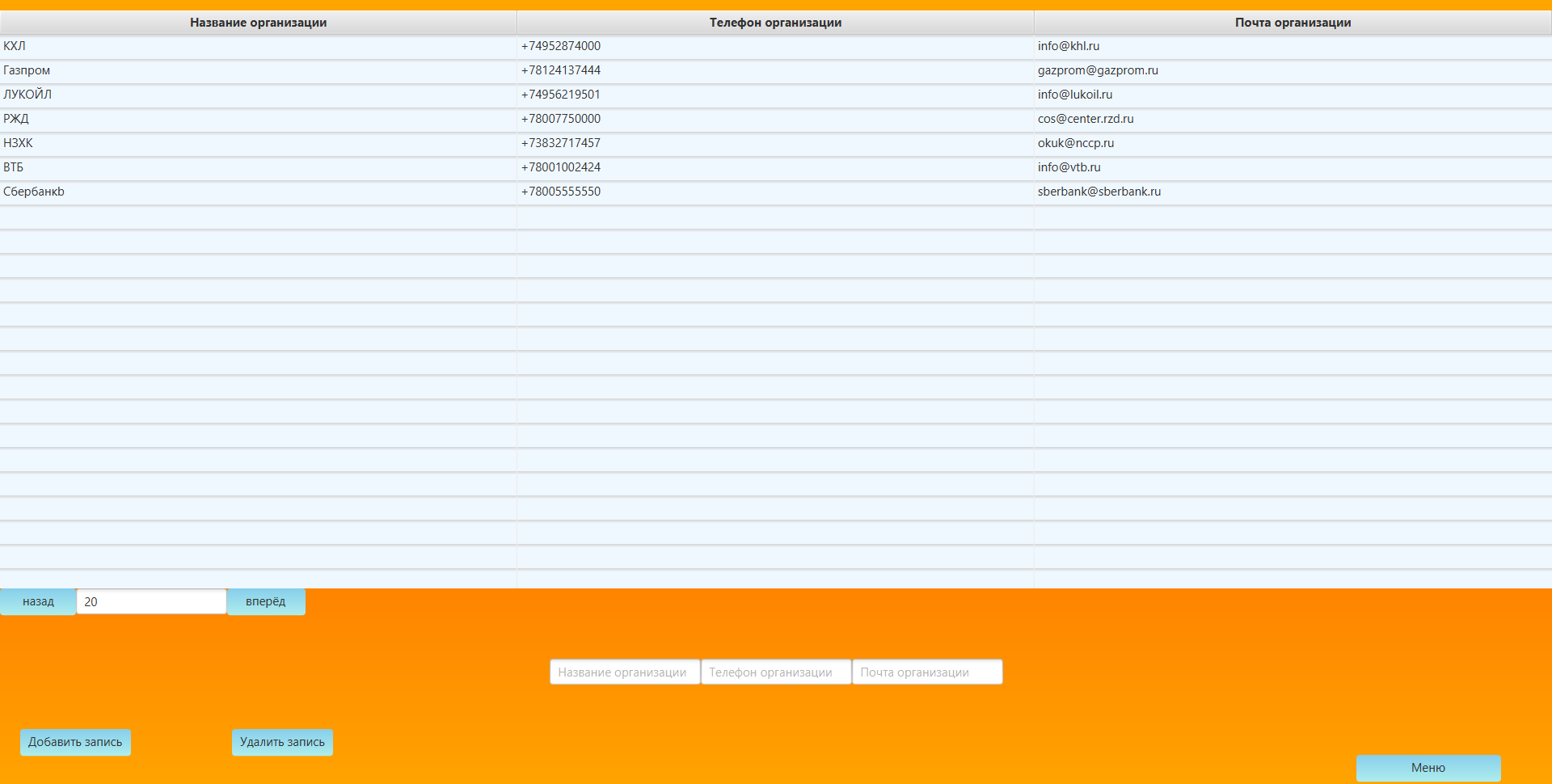
1. Соревнования:



1. Участники соревнований:



1. Организаторы:



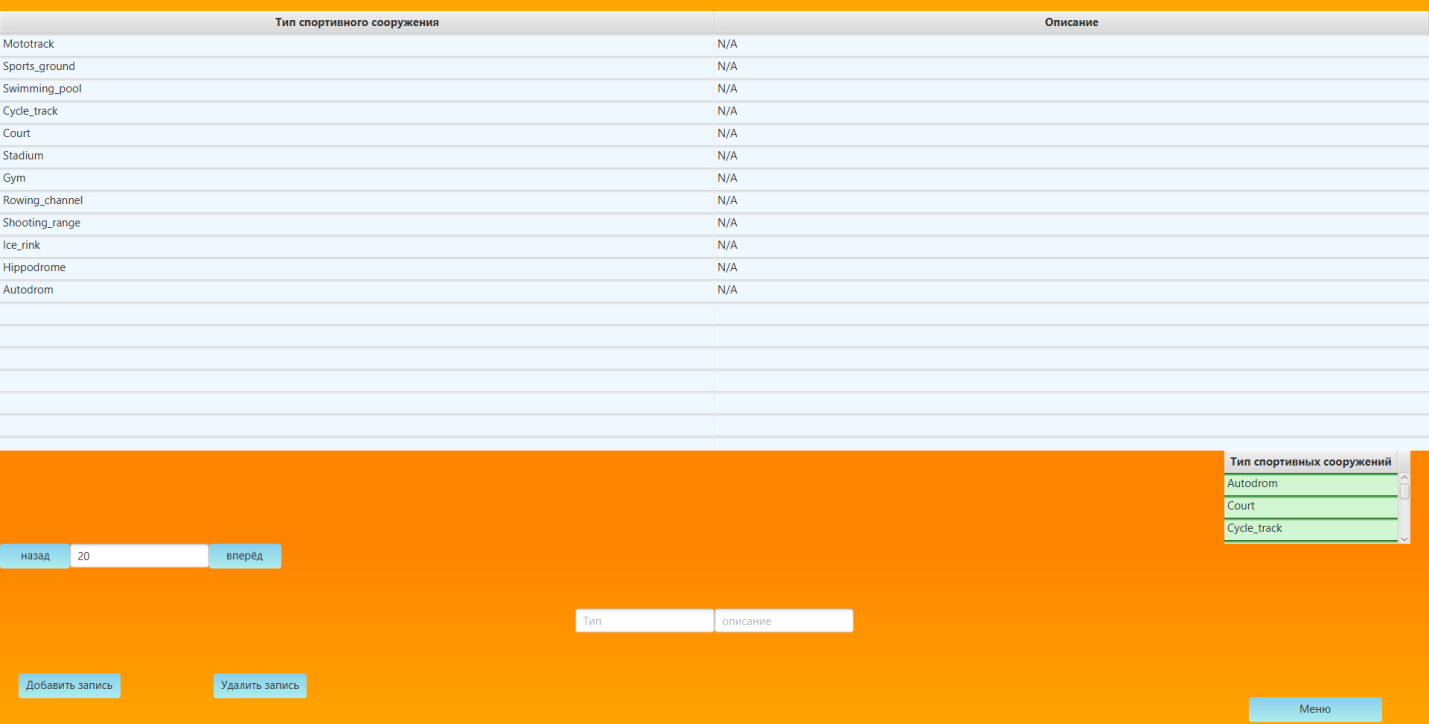
1. Победители соревнований:



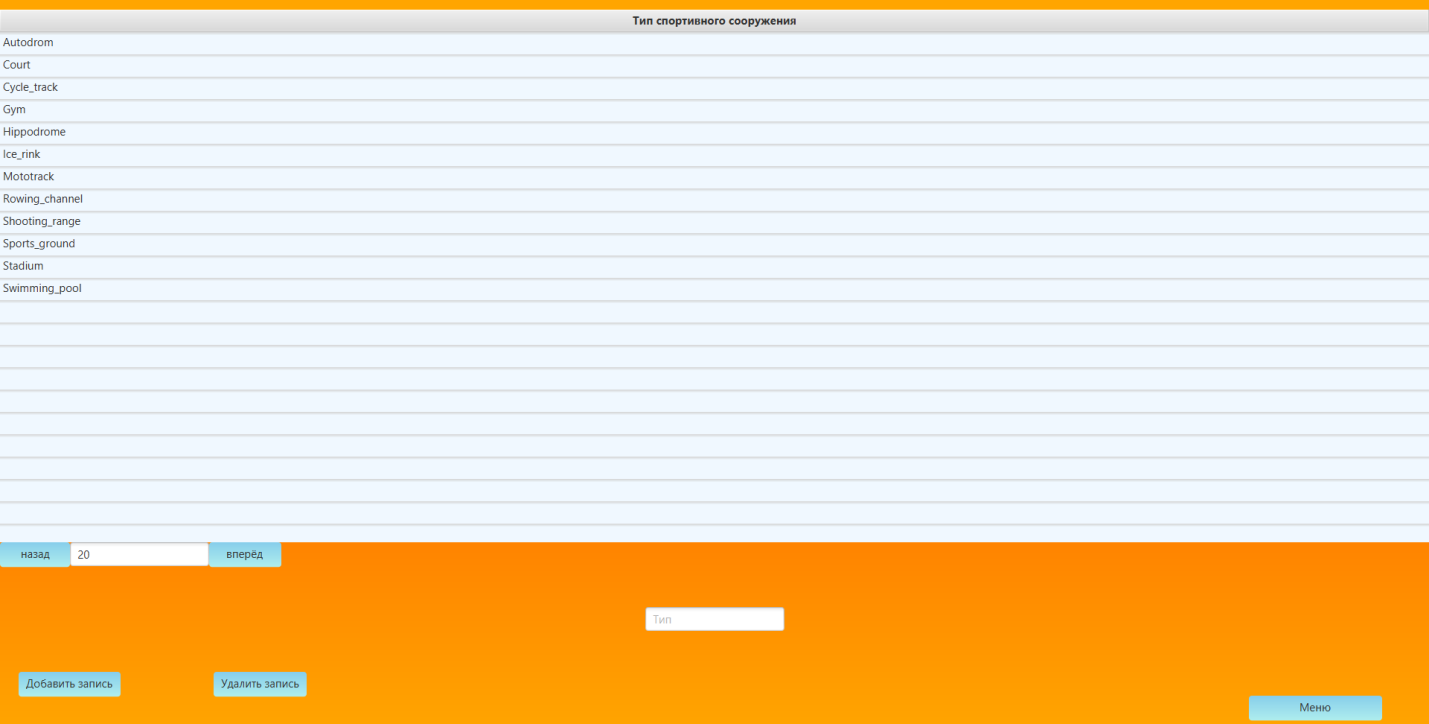
1. Информация о спортивных сооружениях:



1. Типы спортивных инфраструктур



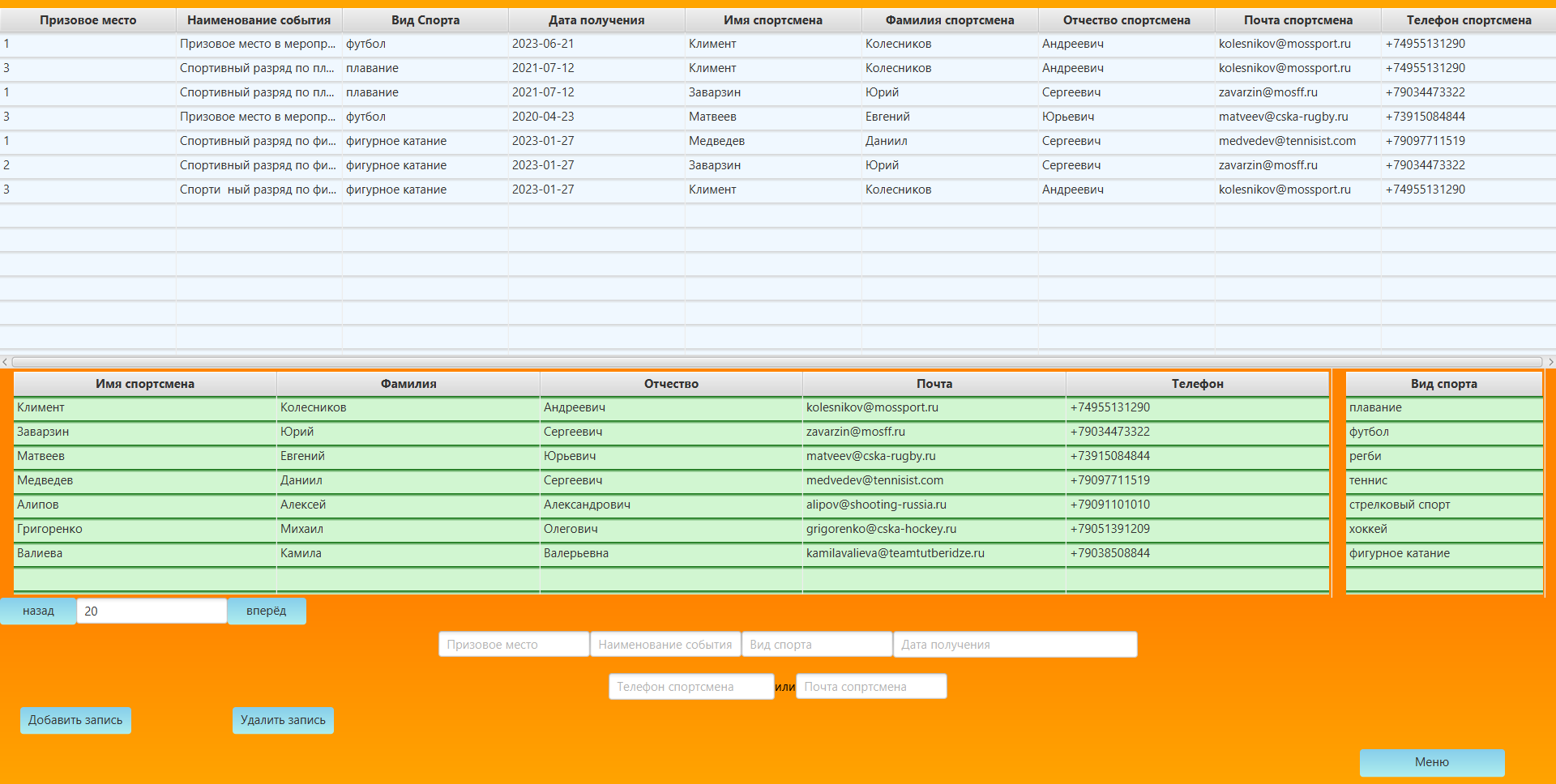
1. Типы спортивных инфраструктур(упрощенные)



1. Спортсмены:



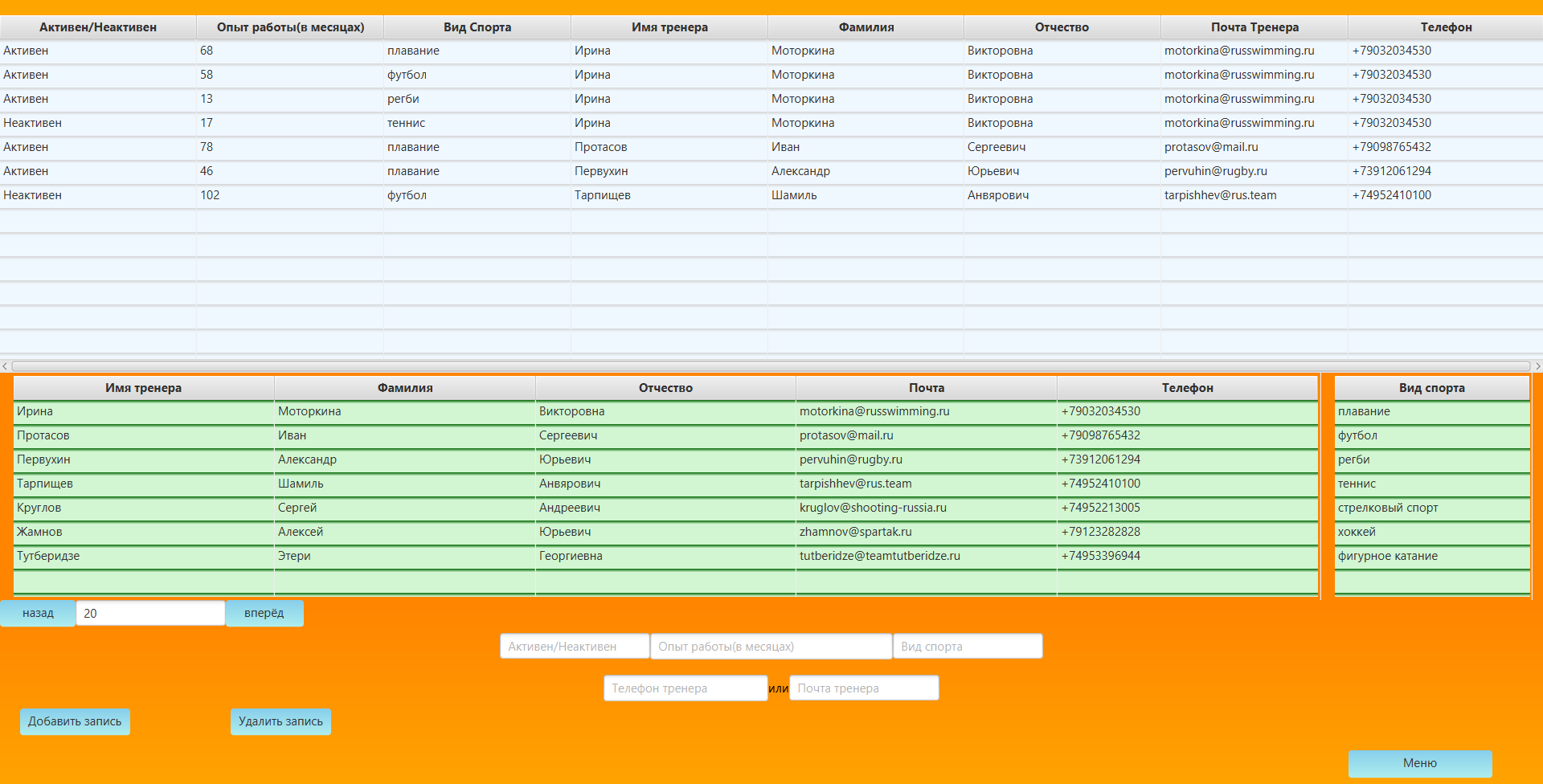
1. Достижения спортсменов:



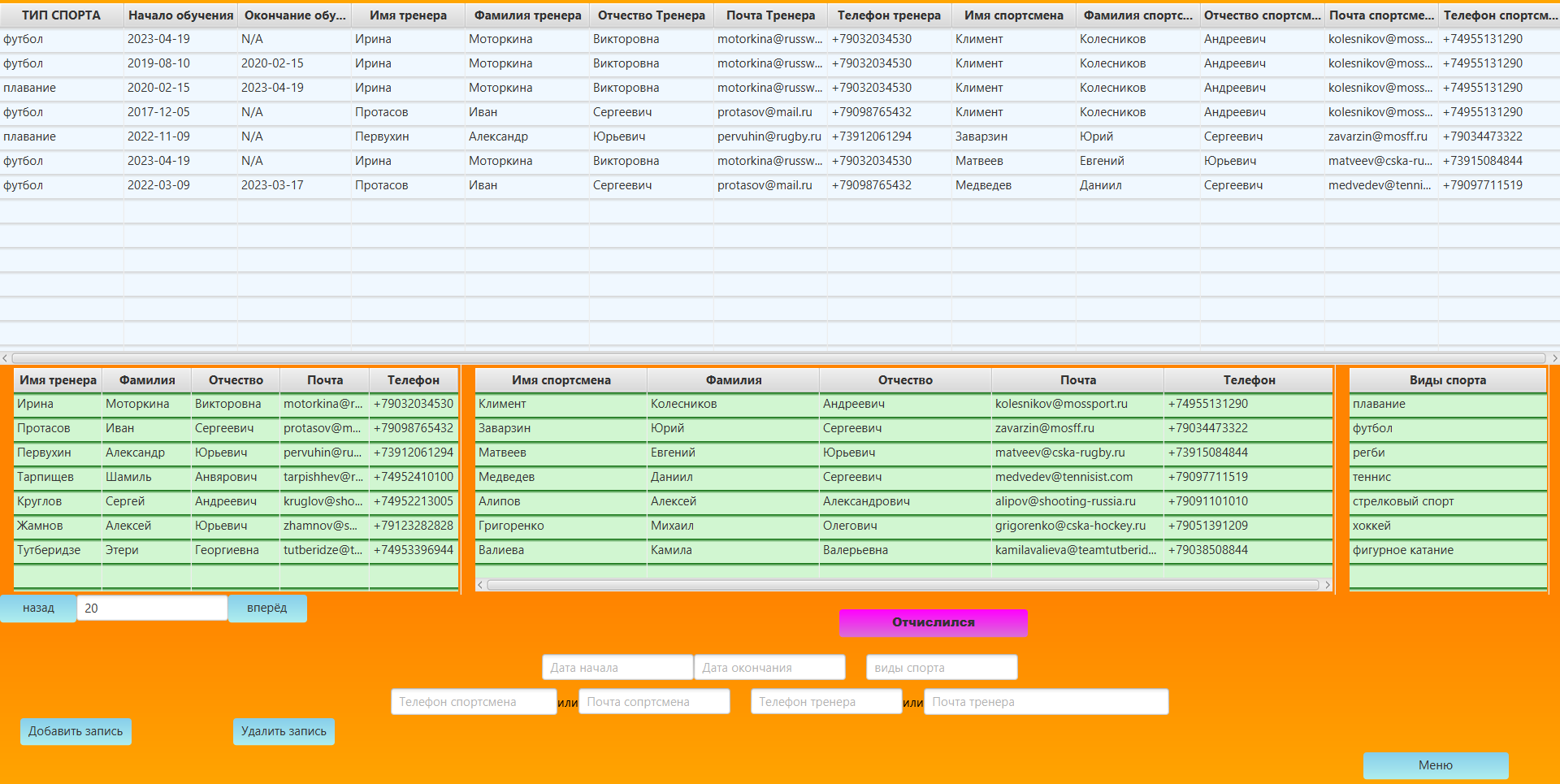
1. Тренеры:



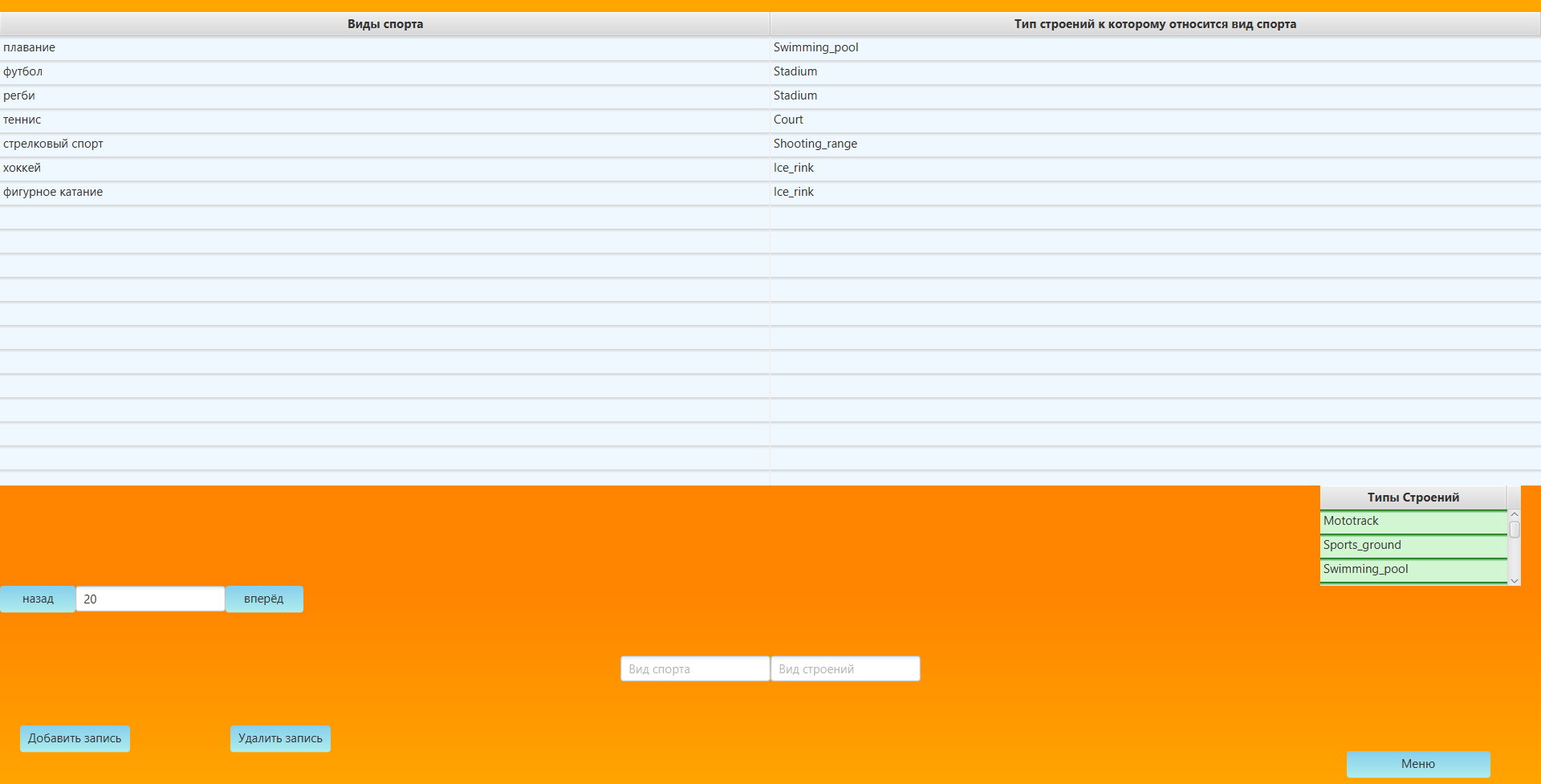
1. Профильная информация тренеров



1. История обучения спортсменов у тренеров



1. Виды спорта



**Заключение**

В результате выполнения проекта была разработана структура базы данных, которая включает в себя инфологическую модель задачи, схему таблиц и SQL скрипты для создания БД на стороне сервера. Также были написаны SQL запросы для работы с базой данных в соответствии с представленными требованиями.

Были реализованы триггеры и хранимые процедуры на языке PL/SQL, которые обеспечивают выполнение запросов в автоматическом режиме и обеспечивают целостность данных.

Наконец, была разработана клиентская часть приложения в виде форм для ввода, удаления, редактирования, отображения таблиц, однако не был реализован поиск данных по запросам. Клиентская часть написана на высокоуровневом языке программирования java c помощью инструмента javafx. Данное приложение позволяет эффективно работать с базой данных из интерфейса пользователя.

Все выполненные работы соответствуют требованиям, предъявляемым к проекту, и готовы к использованию в практических условиях.