МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Базы данных»

Тема: Реализация базы данных с использованием ORM.

Студентка гр. 1303 _____ Королева П.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

Цель работы.

Развернуть Sequalize, написать запросы для создания и заполнения таблиц, написать запросы к БД, отвечающие на вопросы в задания.

Задание.

Вариант 12

Пусть требуется создать программную систему, предназначенную для организаторов соревнований по футболу в рамках первенства страны. Такая система должна обеспечивать хранение сведений о командах, участвующих в первенстве, об игроках команд, о расписании встреч и их результатах, о цене билетов на игры. Сведения о команде — название команды, город, где она базируется, имя главного тренера, место в таблице прошлого сезона, расписание встреч. В один день команда может участвовать только в одной встрече. Сведения об игроке включают в себя фамилию и имя игрока, его возраст, номер и амплуа в команде. Сведения о стадионе, на котором происходит встреча содержат город, в котором он находится, название стадиона, и его вместимость. Цена билета на матч зависит от вместимости стадиона и положения встречающихся команд в турнирной таблице прошлого сезона (наибольшая - при игре тройки призеров, наименьшая — при игре тройки аутсайдеров). Организаторы соревнований должны иметь возможность внести изменения в данные о составе команд, перенести встречу. Им могут потребоваться следующие сведения:

- Даты встреч указанной команды, ее противники и счет?
- Номера и фамилии игроков команд, участвовавших во встрече, которая проходила в указанный день в указанном городе?
- Цена, билета на матч между указанными командами?
- Игрок, забивший в турнире наибольшее количество мячей?
- Команды, имеющие наилучшую и наихудшую разницу забитых и пропущенных мячей?
- Самый молодой участник турнира?

- Команды, занявшие призовые места?

Выполнение работы.

Создана база данных, состоящая из 6 схем «Player», «Team», «TeamGame», «Sponsor», «Schedule», «Stadium».

Схемы заполнены значениями, в каждой 5-10 кортежей. Результат представлен на рисунках 1-6.

```
export const Player = sequelize.define("Player",
{
    id_player: {
        type: DataTypes.SMALLINT,
        primaryKey: true
    },
    teamName: {
        type: DataTypes.TEXT,
    },
    lastName:{
        type: DataTypes.TEXT,
    },
    firstName:{
        type: DataTypes.TEXT,
    },
    age: {
        type: DataTypes.SMALLINT,
    },
    role: {
        type: DataTypes.TEXT,
    },
    number: {
        type: DataTypes.SMALLINT,
    },
    goals: {
        type: DataTypes.SMALLINT,
    }},{
        freezeTableName: true
})
```

Рисунок 1 – Наполнение схемы «Player»

```
export const Team = sequelize.define("Team",
{
    teamName: {
        type: DataTypes.TEXT,
        primaryKey: true
    },
    city:{
        type: DataTypes.TEXT,
    },
    coach:{
        type: DataTypes.TEXT,
    },
    place: {
        type: DataTypes.SMALLINT,
    }},{
        freezeTableName: true
})
```

Рисунок 2 – Наполнение схемы «Team»

```
export const TeamGame = sequelize.define("TeamGame",
{
    teamName1: {
        type: DataTypes.TEXT,
        references: {
            model: Team,
            key: "teamName"
        }
    },
    teamName2:{
        type: DataTypes.TEXT,
        references: {
            model: Team,
            key: "teamName"
        }
    },
    id_game: {
        type: DataTypes.SMALLINT,
        primaryKey: true,
        references: {
            model: Schedule,
            key: "id_game"
        }
    }},{
        freezeTableName: true
})
```

Рисунок 3 – Наполнение схемы «TeamGame»

```
export const Sponsor = sequelize.define("Sponsor",
{
    sponsorName: {
        type: DataTypes.TEXT,
        primaryKey: true
    }},{
        freezeTableName: true
})
```

Рисунок 4 – Наполнение схемы «Sponsor»

```
export const Schedule = sequelize.define("Schedule",
{
    id_game: {
        type: DataTypes.SMALLINT,
        primaryKey: true
    },
    date: {
        type: DataTypes.DATEONLY,
    },
    score1:{
        type: DataTypes.SMALLINT,
    },
    score2:{
        type: DataTypes.SMALLINT,
    },
    price: {
        type: DataTypes.INTEGER,
    },
    stadiumName: {
        type: DataTypes.TEXT,
    },
    sponsorName: {
        type: DataTypes.TEXT,
    }},{
        freezeTableName: true
})
```

Рисунок 5 – Наполнение схемы «Schedule»

```
export const Stadium = sequelize.define("Stadium",
{
    stadiumName: {
        type: DataTypes.TEXT,
        primaryKey: true
    },
    city:{
        type: DataTypes.TEXT,
    },
    capacity: {
        type: DataTypes.INTEGER,
    }},{
        freezeTableName: true
})
```

Рисунок 6 – Наполнение схемы «Stadium»

Написаны запросы для ответа на вопросы в задании. Результаты приведены на рисунках 7-13.

```
await Schedule.findAll({
    attributes: ["date", "score1", "score2"],
    include: {
        model: TeamGame,
        attributes: ["teamName1", "teamName2"],
        required: true,
        where: {
            [Op.or]: [
                { teamName1: "Локомотив" },
                { teamName2: "Локомотив" },
            ]
        }
    },
    raw: true
}).then(res =>
    console.log("\nДаты встреч команды Локомотив, ее противники и счет:", JSON.stringify(res, null, 2), "\n"))
```

```
Даты встреч команды Локомотив, ее противники и счет:
  {
    "date": "2023-12-17",
    "score1": 1,
    "score2": 0,
    "TeamGames.teamName1": "Зенит",
    "TeamGames.teamName2": "Локомотив"
  },
  {
    "date": "2024-02-05",
    "score1": 2,
    "score2": 7,
    "TeamGames.teamName1": "Локомотив",
    "TeamGames.teamName2": "Рубин"
  }
]
```

Рисунок 7. Даты встреч команды, ее противники и счет.

```
await Player.findAll({
    attributes: ["number", "lastName"],
    where: {
        [Op.and]: [
            {
                [Op.or]: [{ "$Team->tN1->Schedule->Stadium.city$": "Санкт-Петербург" }, { "$Team->tN2->Schedule->Stadium.city$": "Санкт-Петербург" }]
            },
            {
                [Op.or]: [{ "$Team->tN1->Schedule.date$": "2024-02-05" }, { "$Team->tN2->Schedule.date$": "2024-02-05" }]
            }
        ]
    },
    include: {
        model: Team,
        required: true,
        attributes: [],
```

```
        include: [
            {
                model: TeamGame,
                as: "tN1",
                attributes: [],
                required: true,
                include: {
                    model: Schedule,
                    attributes: [],
                    required: true,
                    include: {
                        model: Stadium,
                        required: true,
                        attributes: [],
                    }
                }
            },
            {
                model: TeamGame,
                as: "tN2",
                attributes: [],
                required: true,
```

```
                include: {
                    model: Schedule,
                    attributes: [],
                    required: true,
                    include: {
                        model: Stadium,
                        attributes: [],
                        required: true,
                    }
                }
            },
        ]
    },
    raw: true
}).then(res =>
    console.log("\nНомера и Фамилии игроков команд, участвовахших во встрече, которая проходила 2024-02-05 в Санкт-Петербурге:", JSON.stringify(res,
```

```
Номера и Фамилии игроков команд, участвовахших во встрече, которая проходила 2024-02-05 в Санкт-Петербурге: [
  {
    "number": 7,
    "lastName": "Дзюба"
  },
  {
    "number": 6,
    "lastName": "Баринов"
  },
  {
    "number": 31,
    "lastName": "Рыбус"
  },
  {
    "number": 37,
    "lastName": "Кашуба"
  }
]
```

Рисунок 8. Номера и фамилии игроков команд, участвовавших во встрече, которая проходила в указанный день в указанном городе.

```
await Schedule.findAll({
    attributes: ["price"],
    include: {
        model: TeamGame,
        attributes: [],
        required: true,
        where: {
            [Op.or]: [
                { [Op.and]: [{ teamName1: "Балтика" }, { teamName2: "Динамо" }] },
                { [Op.and]: [{ teamName2: "Балтика" }, { teamName1: "Динамо" }] }
            ]
        }
    },
    raw: true
}).then(res =>
    console.log("\nЦена билета на матч между командами 'Балтика' и 'Динамо':", JSON.stringify(res, null, 2), "\n"))
```

```
Цена билета на матч между командами 'Балтика' и 'Динамо': [
  {
    "price": 1000
  }
]
```

Рисунок 9. Цена, билета на матч между указанными командами.

```
await Player.findAll({
    attributes: ["firstName", "lastName", "goals"],
    where: {
        goals: await Player.max("goals") //sequelize.fn("MAX", sequelize.col("goals"))
    }
}).then(res =>
    console.log("\nИгрок, забивший в турнире наибольшее количество мячей:", JSON.stringify(res, null, 2), "\n"))
```

```
Игрок, забивший в турнире наибольшее количество мячей: [
  {
    "firstName": "Максим",
    "lastName": "Дудко",
    "goals": 28
  }
]
```

Рисунок 10. Игрок, забивший в турнире наибольшее количество мячей.

```
    await sequelize.query(
        `CREATE TEMPORARY TABLE "Difference" AS
        SELECT "teamName1" AS "team", ("Schedule"."score1" - "Schedule"."score2") AS "diff"
        FROM "TeamGame"
        JOIN "Schedule" ON "TeamGame"."id_game" = "Schedule"."id_game"
        UNION
        SELECT "teamName2" AS "team", ("Schedule"."score2" - "Schedule"."score1") AS "diff"
        FROM "TeamGame"
        JOIN "Schedule" ON "TeamGame"."id_game" = "Schedule"."id_game";


        SELECT "team", "diff"
        FROM "Difference"
        WHERE "diff" = (SELECT MIN("diff") FROM "Difference")
            OR
            "diff" = (SELECT MAX("diff") FROM "Difference");`,
        { type: sequelize.QueryTypes.SELECT }).then(res => console.log("\nКомманды с наилучшей и наихудшей разницей забитых и пропущенных мячей", JSON.string
}
```

```
Комманды с наилучшей и наихудшей разницей забитых и пропущенных мячей: [
  {
    "teamName": "Локомотив",
    "diff": -5
  },
  {
    "teamName": "Рубин",
    "diff": 5
  }
]
```

Рисунок 11. Команды, имеющие наилучшую и наихудшую разницу забитых и пропущенных мячей.

```
await Player.findAll({
    attributes: ["firstName", "lastName", "age"],
    where: {
        age: await Player.min("age")
    }
}).then(res =>
    console.log("\nСамый молодой участник турнира:", JSON.stringify(res, null, 2), "\n"))
```

```
Самый молодой участник турнира: [
  {
    "firstName": "Данил",
    "lastName": "Кашуба",
    "age": 19
  }
]
```

Рисунок 12. Самый молодой участник турнира.

```
await Team.findAll({
    attributes: ["teamName", "place"],
    where: {
        place: {
            [Op.lte]: 3
        }
    }
}).then(res =>
    console.log("\nКоманды, занявшие призовые места:", JSON.stringify(res, null, 2), "\n"))
```

```
Команды, занявшие призовые места: [
  {
    "teamName": "Балтика",
    "place": 1
  },
  {
    "teamName": "Зенит",
    "place": 2
  },
  {
    "teamName": "Динамо",
    "place": 3
  }
]
```

Рисунок 13. Команды, занявшие призовые места.

Для запуска необходимо:

- Установить зависимости

- Написать в терминале *node .\src\queries\complete_query.js* для запуска создания и заполнения таблиц, и для выполнения запросов.

Вывод.

Был развернут Sequalize, написаны запросы для создания и заполнения таблиц в соответствии со структурой БД, написаны запросы к БД, отвечающие на вопросы в задания.