**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. 2384 |  | Соц Е.А. |
| --- | --- | --- |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2024

## **Цель работы**

Реализовать ранее спроектированную базу данных с использованием ORM, сделать запросы в соответствии с заданием.

## **Задание**

Вариант 18

В данной лабораторной работе рекомендуется использовать Sequelize (Node.js).

Необходимо развернуть Sequelize на своем ПК и выполнить следующие задачи:

* Описать в виде моделей Sequelize таблицы из 1-й лабораторной работы
* Написать скрипт заполнения тестовыми данными: 5-10 строк на каждую таблицу, обязательно наличие связи между ними, данные приближены к реальности.
* Написать запросы к БД, отвечающие на вопросы из 1-й лабораторной работы с использованием ORM. Вывести результаты в консоль (или иной человеко-читабельный вывод)
* Запушить в репозиторий исходный код проекта, соблюсти .gitignore, убрать исходную базу из проекта (или иные нагенерированные данные бд если они есть).
* Описать процесс запуска: команды, зависимости
* В отчете описать цель, текст задания в соответствии с вариантом, выбранную ORM, инструкцию по запуску, скриншоты (код) моделей ORM, скриншоты на каждый запрос (или группу запросов) на изменение/таблицы с выводом результатов (ответ), ссылку на PR в приложении, вывод

## **Выполнение работы**

В ходе выполнения первой лабораторной работы была описана структура базы данных, предназначенная для менеджера музыкальных групп:

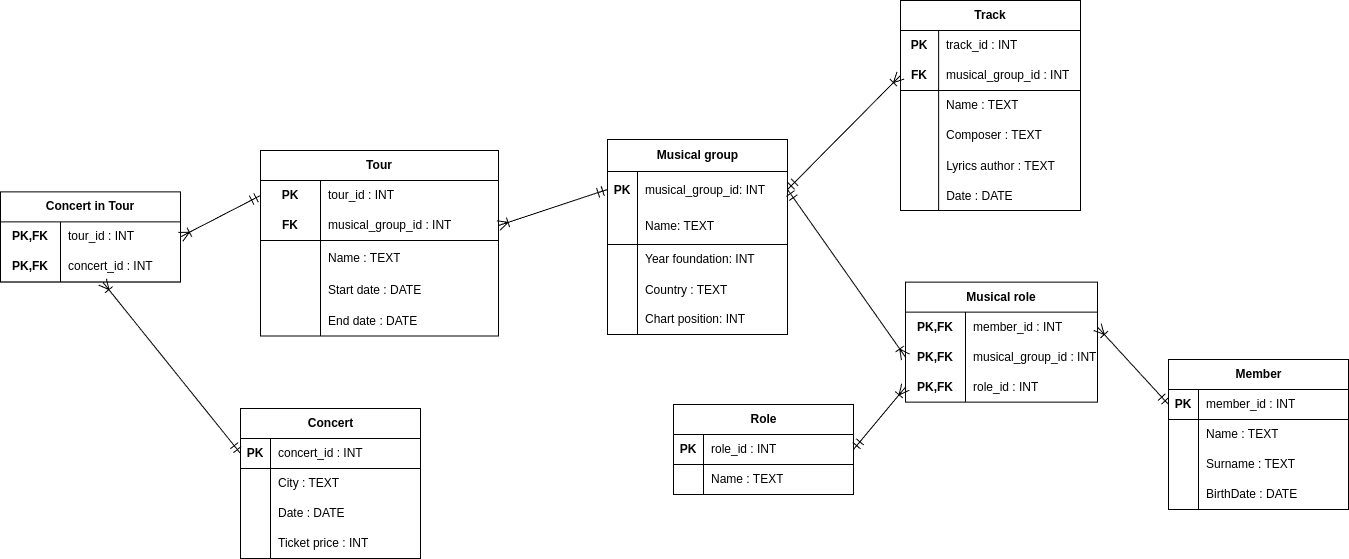


Рисунок 1 – структура БД

Был создан конфигурационный объект, который может используется для настройки подключения к базе данных с использованием Sequelize.

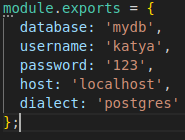


Рисунок 2 – config.js

Также был создан файл, где происходит создание экземпляра Sequelize для подключения к базе данных PostgreSQL с использованием конфигурационных параметров, загруженных из файла ./config/config.

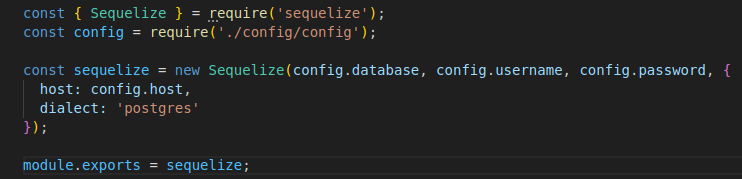


Рисунок 3 – sequelize.js

Рассмотрим модели, созданные с помощью Sequelize: используется метод sequelize.define(). Модель в Sequelize представляет собой таблицу в базе данных, а её атрибуты (поля) представляют столбцы этой таблицы.

Код с описанием моделей прикреплен в приложении А.

Можно заметить, что каждый файл модели начинается с импорта необходимых модулей: DataTypes – объект, содержащий различные типы данных, которые могут быть использованы для определения полей модели;

sequelize – экземпляр Sequelize, который уже настроен для подключения к базе данных.

Также файлах моделей описаны связи между ними:

Метод *hasMany()* определяет связь "один ко многим" (one-to-many). Это означает, что одна запись в одной таблице может быть связана с несколькими записями в другой таблице.

Метод *belongsTo()* определяет связь "многие к одному" (many-to-one). Это означает, что несколько записей в одной таблице могут быть связаны с одной записью в другой таблице.

Для заполнения базы данных был написан скрипт: он создает таблицы, если они не существуют, и заполняет их тестовыми данными. Метод *bulkCreate* используется для массового создания записей в таблицах. Он принимает массив объектов, каждый из которых представляет запись, которую нужно добавить в таблицу.

Код скрипта прикреплен в приложении А.

В файле queries.js реализованы запросы, которые описаны в тексте задания.

Код запросов тоже прикреплен в приложении А.

Ниже прикреплены снимки экрана с выводом запросов.

1. Автор текста, композитор и дата создания песни с данным названием? В репертуар какой группы она входит?

Для теста была выбрана песня “Дворцовый мост”.

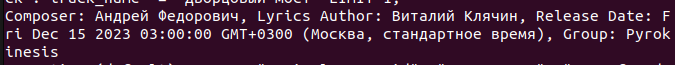


Рисунок 4 – Информация об определенной песне

1. Репертуар наиболее популярной группы?

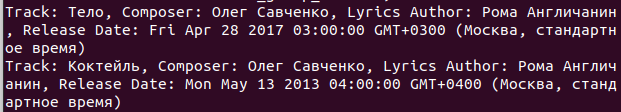


Рисунок 5 – Репертуар самой популярной группы

1. Цена билета на последний концерт указанной группы?

Для теста была выбрана группа “Монеточка”.



Рисунок 6 – Цена билета

1. Состав исполнителей группы с заданным названием, их возраст и амплуа?

Для теста была выбрана группа “The Cure”.

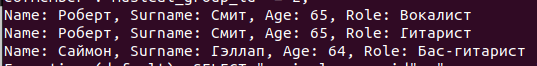


Рисунок 7 – Информация о составе определенной группы

1. Место и продолжительность гастролей группы с заданным названием?

Для теста была выбрана группа “ЛСП”.



Рисунок 8 – Информация о турах заданной группы

1. Какие группы в текущем году отмечают юбилей?

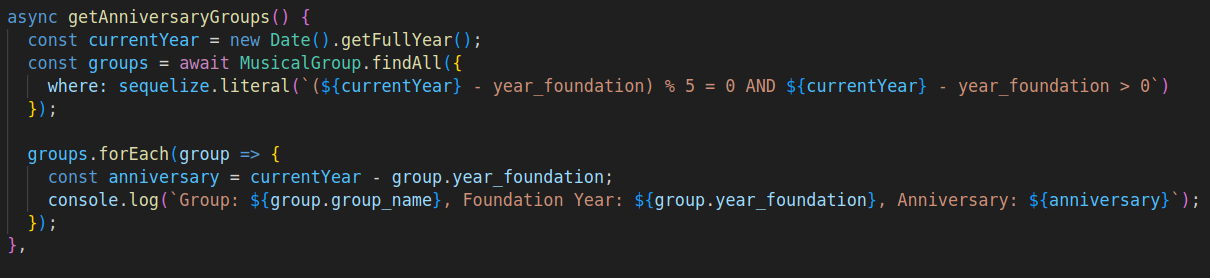


Рисунок 9 – Запрос “Юбилей в текущем году”

Никакой информации не получено, но это правильный ответ, ведь по моим тестовым данным никакая группа не отмечает юбилей текущем году.

Для корректной проверки запроса можно задать какой-то конкретный год юбилея:

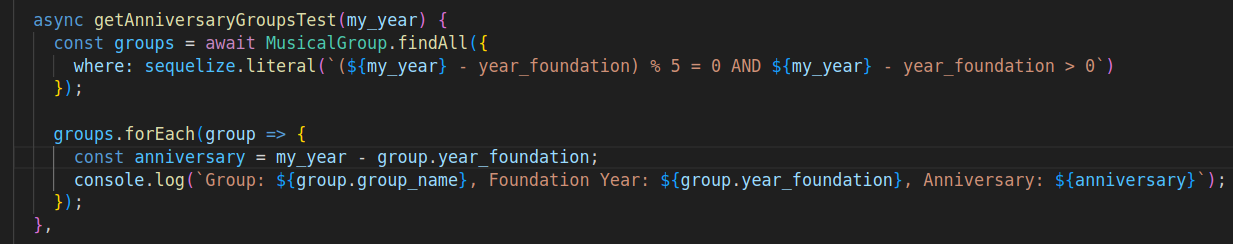


Рисунок 10 – Запрос “Юбилей в конкретном году”

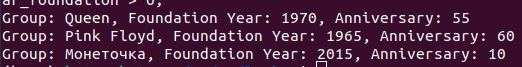


Рисунок 11 – Юбилей в конкретном году

1. Самый молодой вокалист? Какую группу он представляет?



Рисунок 12 – Самый молодой вокалист

Для выполнения набора запросов был реализован файл app.js:

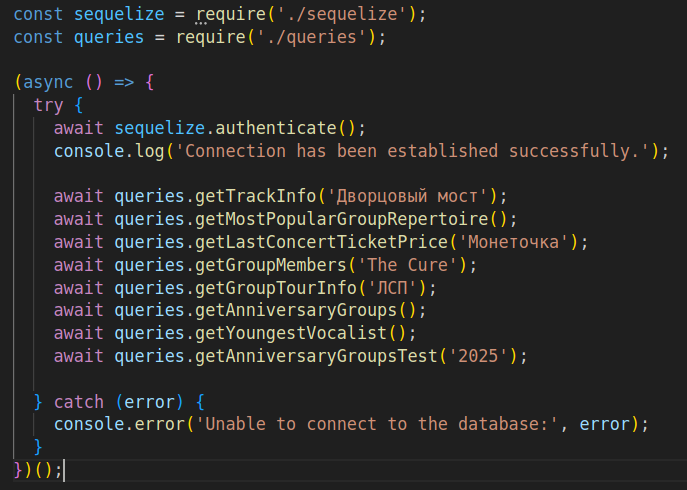


Рисунок 13 – Запуск запросов

В приложении В представлена ссылка на PR.

## **Вывод**

В ходе лабораторной работы были созданы модели ранее спроектированной базы данных с использование Sequelize. Был написан скрипт для заполнения тестовыми данными с помощью NodeJS и описаны связи между таблицами. Также были реализованы запросы в соответствии с заданием и был проведен анализ ответов: ответы на запросы со второй лабораторной работы совпадают с ответами из третьей лабораторной работы, что говорит о правильности написания запросов (тестовые данные для наполнения таблиц совпадают).

## **ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД**

Модель MusicalGroup:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const MusicalGroup = sequelize.define('MusicalGroup', {

musical\_group\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true

},

group\_name: {

type: DataTypes.STRING(50),

allowNull: false

},

year\_foundation: {

type: DataTypes.INTEGER,

allowNull: false

},

country: {

type: DataTypes.STRING(30),

allowNull: false

},

chart\_position: {

type: DataTypes.INTEGER,

allowNull: false

}

}, {

tableName: 'musical\_group',

timestamps: false

});

module.exports = MusicalGroup;

Модель Track:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const MusicalGroup = require('./MusicalGroup');

const Track = sequelize.define('Track', {

track\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true

},

musical\_group\_id: {

type: DataTypes.INTEGER,

allowNull: false,

references: {

model: MusicalGroup,

key: 'musical\_group\_id'

}

},

track\_name: {

type: DataTypes.STRING(50),

allowNull: false

},

composer: {

type: DataTypes.STRING(50),

allowNull: false

},

lyrics\_author: {

type: DataTypes.STRING(50),

allowNull: false

},

release\_date: {

type: DataTypes.DATE,

allowNull: false

}

}, {

tableName: 'track',

timestamps: false

});

Track.belongsTo(MusicalGroup, { foreignKey: 'musical\_group\_id' });

MusicalGroup.hasMany(Track, { foreignKey: 'musical\_group\_id' });

module.exports = Track;

Модель Member:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const Member = sequelize.define('Member', {

member\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true

},

member\_name: {

type: DataTypes.STRING(30),

allowNull: false

},

surname: {

type: DataTypes.STRING(30),

allowNull: false

},

birth\_date: {

type: DataTypes.DATE,

allowNull: false

}

}, {

tableName: 'member',

timestamps: false

});

module.exports = Member;

Модель MusicalRole:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const MusicalRole = sequelize.define('MusicalRole', {

musical\_role\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true

},

role\_name: {

type: DataTypes.STRING(30),

allowNull: false

}

}, {

tableName: 'musical\_role',

timestamps: false

});

module.exports = MusicalRole;

Модель RoleOfMember:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const Member = require('./Member');

const MusicalGroup = require('./MusicalGroup');

const MusicalRole = require('./MusicalRole');

const RoleOfMember = sequelize.define('RoleOfMember', {

member\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

references: {

model: Member,

key: 'member\_id'

}

},

musical\_group\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

references: {

model: MusicalGroup,

key: 'musical\_group\_id'

}

},

musical\_role\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

references: {

model: MusicalRole,

key: 'musical\_role\_id'

}

}

}, {

tableName: 'role\_of\_member',

timestamps: false

});

RoleOfMember.belongsTo(Member, { foreignKey: 'member\_id' });

RoleOfMember.belongsTo(MusicalGroup, { foreignKey: 'musical\_group\_id' });

RoleOfMember.belongsTo(MusicalRole, { foreignKey: 'musical\_role\_id' });

Member.hasMany(RoleOfMember, { foreignKey: 'member\_id' });

MusicalGroup.hasMany(RoleOfMember, { foreignKey: 'musical\_group\_id' });

MusicalRole.hasMany(RoleOfMember, { foreignKey: 'musical\_role\_id' });

module.exports = RoleOfMember;

Модель Tour:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const MusicalGroup = require('./MusicalGroup');

const Tour = sequelize.define('Tour', {

tour\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true

},

musical\_group\_id: {

type: DataTypes.INTEGER,

allowNull: false,

references: {

model: MusicalGroup,

key: 'musical\_group\_id'

}

},

tour\_name: {

type: DataTypes.STRING(50),

allowNull: false

},

start\_day: {

type: DataTypes.DATE,

allowNull: false

},

end\_day: {

type: DataTypes.DATE,

allowNull: false

}

}, {

tableName: 'tour',

timestamps: false

});

Tour.belongsTo(MusicalGroup, { foreignKey: 'musical\_group\_id' });

MusicalGroup.hasMany(Tour, { foreignKey: 'musical\_group\_id' });

module.exports = Tour;

Модель Concert:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const Concert = sequelize.define('Concert', {

concert\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true

},

city: {

type: DataTypes.STRING(30),

allowNull: false

},

date\_concert: {

type: DataTypes.DATE,

allowNull: false

},

ticket\_price: {

type: DataTypes.INTEGER,

allowNull: false

}

}, {

tableName: 'concert',

timestamps: false

});

module.exports = Concert;

Модель ConcertINTour:

const { DataTypes } = require('sequelize');

const sequelize = require('../sequelize');

const Tour = require('./Tour');

const Concert = require('./Concert');

const ConcertInTour = sequelize.define('ConcertInTour', {

tour\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

references: {

model: Tour,

key: 'tour\_id'

}

},

concert\_id: {

type: DataTypes.INTEGER,

primaryKey: true,

references: {

model: Concert,

key: 'concert\_id'

}

}

}, {

tableName: 'concert\_in\_tour',

timestamps: false

});

ConcertInTour.belongsTo(Tour, { foreignKey: 'tour\_id' });

ConcertInTour.belongsTo(Concert, { foreignKey: 'concert\_id' });

Tour.hasMany(ConcertInTour, { foreignKey: 'tour\_id' });

Concert.hasMany(ConcertInTour, { foreignKey: 'concert\_id' });

module.exports = ConcertInTour;

Скрипт для заполнения моделей seed.js

const sequelize = require('../sequelize');

const MusicalGroup = require('../models/MusicalGroup');

const Track = require('../models/Track');

const Member = require('../models/Member');

const MusicalRole = require('../models/MusicalRole');

const RoleOfMember = require('../models/RoleOfMember');

const Tour = require('../models/Tour');

const Concert = require('../models/Concert');

const ConcertInTour = require('../models/ConcertInTour');

(async () => {

await sequelize.sync({ force: true }); // Удаляет и создает таблицы заново

await MusicalGroup.bulkCreate([

{ group\_name: 'Queen', year\_foundation: 1970, country: 'Великобритания', chart\_position: 5 },

{ group\_name: 'The Cure', year\_foundation: 1978, country: 'Англия', chart\_position: 2 },

{ group\_name: 'Pink Floyd', year\_foundation: 1965, country: 'Англия', chart\_position: 4 },

{ group\_name: 'Монеточка', year\_foundation: 2015, country: 'Россия', chart\_position: 3 },

{ group\_name: 'ЛСП', year\_foundation: 2007, country: 'Беларусь', chart\_position: 1 },

{ group\_name: 'Pyrokinesis', year\_foundation: 2012, country: 'Россия', chart\_position: 6 }

]);

await Track.bulkCreate([

{ musical\_group\_id: 1, track\_name: 'Bohemian Rhapsody', composer: 'Queen', lyrics\_author: 'Фредди Меркьюри', release\_date: '1975-10-31' },

{ musical\_group\_id: 1, track\_name: 'We Will Rock You', composer: 'Брайан Мэй', lyrics\_author: 'Брайан Мэй', release\_date: '1977-10-07' },

{ musical\_group\_id: 2, track\_name: 'Lovesong', composer: 'Роберт Смит', lyrics\_author: 'Роберт Смит', release\_date: '1989-08-21' },

{ musical\_group\_id: 2, track\_name: 'Friday I am in Love', composer: 'Роберт Смит', lyrics\_author: 'Роберт Смит', release\_date: '1992-05-11' },

{ musical\_group\_id: 3, track\_name: 'Another Brick in the Wall', composer: 'Роджер Уотерс', lyrics\_author: 'Роджер Уотерс', release\_date: '1979-11-30' },

{ musical\_group\_id: 3, track\_name: 'Time', composer: 'Роджер Уотерс', lyrics\_author: 'Роджер Уотерс', release\_date: '1973-01-01' },

{ musical\_group\_id: 4, track\_name: 'Это было в России', composer: 'Витя Исаев', lyrics\_author: 'Витя Исаев', release\_date: '2024-05-01' },

{ musical\_group\_id: 4, track\_name: 'Каждый раз', composer: 'Елизавета Гырдымова', lyrics\_author: 'Елизавета Гырдымова', release\_date: '2018-05-25' },

{ musical\_group\_id: 5, track\_name: 'Тело', composer: 'Олег Савченко', lyrics\_author: 'Рома Англичанин', release\_date: '2017-04-28' },

{ musical\_group\_id: 5, track\_name: 'Коктейль', composer: 'Олег Савченко', lyrics\_author: 'Рома Англичанин', release\_date: '2013-05-13' },

{ musical\_group\_id: 6, track\_name: 'Черное солнышко', composer: 'Андрей Федорович', lyrics\_author: 'Андрей Федорович', release\_date: '2019-08-15' },

{ musical\_group\_id: 6, track\_name: 'Дворцовый мост', composer: 'Андрей Федорович', lyrics\_author: 'Виталий Клячин', release\_date: '2023-12-15' }

]);

await Member.bulkCreate([

{ member\_name: 'Брайн', surname: 'Мэй', birth\_date: '1947-07-19' },

{ member\_name: 'Роджер', surname: 'Тейлор', birth\_date: '1949-07-26' },

{ member\_name: 'Роберт', surname: 'Смит', birth\_date: '1959-04-21' },

{ member\_name: 'Саймон', surname: 'Гэллап', birth\_date: '1960-06-01' },

{ member\_name: 'Ник', surname: 'Мейсон', birth\_date: '1944-01-27' },

{ member\_name: 'Роджер', surname: 'Уотерс', birth\_date: '1943-09-06' },

{ member\_name: 'Елизавета', surname: 'Гырдымова', birth\_date: '1998-06-01' },

{ member\_name: 'Олег', surname: 'Савченко', birth\_date: '1989-07-10' },

{ member\_name: 'Петр', surname: 'Клюев', birth\_date: '1989-05-28' },

{ member\_name: 'Андрей', surname: 'Федорович', birth\_date: '1995-12-16' }

]);

await MusicalRole.bulkCreate([

{ role\_name: 'Вокалист' },

{ role\_name: 'Барабанщик' },

{ role\_name: 'Гитарист' },

{ role\_name: 'Бэк-вокалист' },

{ role\_name: 'Автор песен' },

{ role\_name: 'Бас-гитарист' }

]);

await RoleOfMember.bulkCreate([

{ member\_id: 1, musical\_group\_id: 1, musical\_role\_id: 3 },

{ member\_id: 1, musical\_group\_id: 1, musical\_role\_id: 5 },

{ member\_id: 2, musical\_group\_id: 1, musical\_role\_id: 1 },

{ member\_id: 2, musical\_group\_id: 1, musical\_role\_id: 2 },

{ member\_id: 3, musical\_group\_id: 2, musical\_role\_id: 1 },

{ member\_id: 3, musical\_group\_id: 2, musical\_role\_id: 3 },

{ member\_id: 4, musical\_group\_id: 2, musical\_role\_id: 6 },

{ member\_id: 5, musical\_group\_id: 3, musical\_role\_id: 2 },

{ member\_id: 6, musical\_group\_id: 3, musical\_role\_id: 1 },

{ member\_id: 6, musical\_group\_id: 3, musical\_role\_id: 6 },

{ member\_id: 7, musical\_group\_id: 4, musical\_role\_id: 1 },

{ member\_id: 7, musical\_group\_id: 4, musical\_role\_id: 5 },

{ member\_id: 8, musical\_group\_id: 5, musical\_role\_id: 1 },

{ member\_id: 8, musical\_group\_id: 5, musical\_role\_id: 5 },

{ member\_id: 9, musical\_group\_id: 5, musical\_role\_id: 4 },

{ member\_id: 10, musical\_group\_id: 6, musical\_role\_id: 1 },

{ member\_id: 10, musical\_group\_id: 6, musical\_role\_id: 5 }

]);

await Tour.bulkCreate([

{ musical\_group\_id: 1, tour\_name: 'Magic Tour', start\_day: '1986-06-07', end\_day: '1986-08-09' },

{ musical\_group\_id: 2, tour\_name: 'Shows of a lost world', start\_day: '2023-11-19', end\_day: '2023-12-10' },

{ musical\_group\_id: 3, tour\_name: 'Syd BARRET', start\_day: '2007-05-10', end\_day: '2007-05-10' },

{ musical\_group\_id: 4, tour\_name: 'Молитвы. Анекдоты. Тосты', start\_day: '2024-11-11', end\_day: '2024-11-30' },

{ musical\_group\_id: 5, tour\_name: 'Бриллианты', start\_day: '2024-10-12', end\_day: '2024-12-08' },

{ musical\_group\_id: 6, tour\_name: 'Тур-2024', start\_day: '2024-09-06', end\_day: '2024-10-27' }

]);

await Concert.bulkCreate([

{ city: 'Париж', date\_concert: '1986-06-14', ticket\_price: 1000 },

{ city: 'Лондон', date\_concert: '1986-07-11', ticket\_price: 1000 },

{ city: 'Буэнос-Айрес', date\_concert: '2023-11-25', ticket\_price: 5000 },

{ city: 'Монтевидео', date\_concert: '2023-11-27', ticket\_price: 5100 },

{ city: 'Лондон', date\_concert: '2007-05-10', ticket\_price: 2000 },

{ city: 'Барселона', date\_concert: '2024-11-25', ticket\_price: 5800 },

{ city: 'Прага', date\_concert: '2024-11-30', ticket\_price: 7800 },

{ city: 'Тюмень', date\_concert: '2024-11-19', ticket\_price: 2600 },

{ city: 'Омск', date\_concert: '2024-12-08', ticket\_price: 3200 },

{ city: 'Орел', date\_concert: '2024-10-08', ticket\_price: 2500 },

{ city: 'Тамбов', date\_concert: '2024-10-11', ticket\_price: 2000 }

]);

await ConcertInTour.bulkCreate([

{ tour\_id: 1, concert\_id: 1 },

{ tour\_id: 1, concert\_id: 2 },

{ tour\_id: 2, concert\_id: 3 },

{ tour\_id: 2, concert\_id: 4 },

{ tour\_id: 3, concert\_id: 5 },

{ tour\_id: 4, concert\_id: 6 },

{ tour\_id: 4, concert\_id: 7 },

{ tour\_id: 5, concert\_id: 8 },

{ tour\_id: 5, concert\_id: 9 },

{ tour\_id: 6, concert\_id: 10 },

{ tour\_id: 6, concert\_id: 11 }

]);

console.log('Данные успешно загружены');

})();

Запросы queries.js

const { Op } = require('sequelize');

const sequelize = require('./sequelize');

const MusicalGroup = require('./models/MusicalGroup');

const Track = require('./models/Track');

const Member = require('./models/Member');

const MusicalRole = require('./models/MusicalRole');

const RoleOfMember = require('./models/RoleOfMember');

const Tour = require('./models/Tour');

const Concert = require('./models/Concert');

const ConcertInTour = require('./models/ConcertInTour');

module.exports = {

async getTrackInfo(trackName) {

const track = await Track.findOne({

where: { track\_name: trackName },

include: {

model: MusicalGroup,

attributes: ['group\_name']

}

});

console.log(`Composer: ${track.composer}, Lyrics Author: ${track.lyrics\_author}, Release Date: ${track.release\_date}, Group: ${track.MusicalGroup.group\_name}`);

},

async getMostPopularGroupRepertoire() {

const mostPopularGroup = await MusicalGroup.findOne({

order: [['chart\_position', 'ASC']]

});

const tracks = await Track.findAll({

where: { musical\_group\_id: mostPopularGroup.musical\_group\_id }

});

tracks.forEach(track => {

console.log(`Track: ${track.track\_name}, Composer: ${track.composer}, Lyrics Author: ${track.lyrics\_author}, Release Date: ${track.release\_date}`);

});

},

async getLastConcertTicketPrice(groupName) {

const group = await MusicalGroup.findOne({

where: { group\_name: groupName }

});

const tours = await Tour.findAll({

where: { musical\_group\_id: group.musical\_group\_id }

});

const concerts = await ConcertInTour.findAll({

where: { tour\_id: tours.map(tour => tour.tour\_id) },

include: {

model: Concert,

attributes: ['date\_concert', 'ticket\_price']

},

order: [[Concert, 'date\_concert', 'DESC']]

});

console.log(`Last Concert Ticket Price: ${concerts[0].Concert.ticket\_price}`);

},

async getGroupMembers(groupName) {

const group = await MusicalGroup.findOne({

where: { group\_name: groupName }

});

const members = await RoleOfMember.findAll({

where: { musical\_group\_id: group.musical\_group\_id },

include: [

{

model: Member,

attributes: ['member\_name', 'surname', 'birth\_date']

},

{

model: MusicalRole,

attributes: ['role\_name']

}

]

});

members.forEach(member => {

const age = new Date().getFullYear() - new Date(member.Member.birth\_date).getFullYear();

console.log(`Name: ${member.Member.member\_name}, Surname: ${member.Member.surname}, Age: ${age}, Role: ${member.MusicalRole.role\_name}`);

});

},

async getGroupTourInfo(groupName) {

const group = await MusicalGroup.findOne({

where: { group\_name: groupName }

});

const tours = await Tour.findAll({

where: { musical\_group\_id: group.musical\_group\_id },

include: {

model: ConcertInTour,

include: {

model: Concert,

attributes: ['city']

}

}

});

tours.forEach(tour => {

const duration = new Date(tour.end\_day) - new Date(tour.start\_day);

tour.ConcertInTours.forEach(concertInTour => {

console.log(`Tour: ${tour.tour\_name}, City: ${concertInTour.Concert.city}, Duration: ${duration} ms`);

});

});

},

async getAnniversaryGroups() {

const currentYear = new Date().getFullYear();

const groups = await MusicalGroup.findAll({

where: sequelize.literal(`(${currentYear} - year\_foundation) % 5 = 0 AND ${currentYear} - year\_foundation > 0`)

});

groups.forEach(group => {

const anniversary = currentYear - group.year\_foundation;

console.log(`Group: ${group.group\_name}, Foundation Year: ${group.year\_foundation}, Anniversary: ${anniversary}`);

});

},

async getAnniversaryGroupsTest(my\_year) {

const groups = await MusicalGroup.findAll({

where: sequelize.literal(`(${my\_year} - year\_foundation) % 5 = 0 AND ${my\_year} - year\_foundation > 0`)

});

groups.forEach(group => {

const anniversary = my\_year - group.year\_foundation;

console.log(`Group: ${group.group\_name}, Foundation Year: ${group.year\_foundation}, Anniversary: ${anniversary}`);

});

},

async getYoungestVocalist() {

const vocalist = await RoleOfMember.findOne({

where: { musical\_role\_id: 1 },

include: [

{

model: Member,

attributes: ['member\_name', 'surname', 'birth\_date']

},

{

model: MusicalGroup,

attributes: ['group\_name']

}

],

order: [[Member, 'birth\_date', 'DESC']]

});

const age = new Date().getFullYear() - new Date(vocalist.Member.birth\_date).getFullYear();

console.log(`Name: ${vocalist.Member.member\_name}, Surname: ${vocalist.Member.surname}, Age: ${age}, Group: ${vocalist.MusicalGroup.group\_name}`);

}

};

## **ПРИЛОЖЕНИЕ B**

**ССЫЛКА НА PR**

<https://github.com/moevm/sql-2024-2384/pull/19>