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.

Студент гр. 1303 _____ Ягодаров М. А.

Преподаватель _____ Заславский М. М.

Санкт-Петербург

2023

Цель работы.

Создание базы данных с использованием ORM.

Задание.

- 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

Вариант 4.

Пусть требуется создать программную систему, предназначенную для организаторов выставки собак. Она должна обеспечивать хранение сведений о собаках - участниках выставки и экспертах. Для каждой собаки в БД должны храниться сведения, о том, к какому клубу она относится, кличка, порода и возраст, сведения о родословной (номер документа, клички родителей), дата последней прививки, фамилия, имя, отчество и паспортные данные хозяина. На каждый клуб отводится участок номеров, под которыми будут выступать участники выставки. Сведения об эксперте должны включать фамилию и имя, номер ринга, который он обслуживает; клуб, название клуба, в котором он состоит. Каждый ринг могут обслуживать несколько экспертов. Каждая порода

собак выступает на своем ринге, но на одном и том же ринге в разное время могут выступать разные породы. Итогом выставки является определение медалистов по каждой породе. Организатор выставки должен иметь возможность добавить в базу нового участника или нового эксперта, снять эксперта с судейства, заменив его другим, отстранить собаку от участия в выставке. Организатору выставки могут потребоваться следующие сведения:

- На каком ринге выступает заданный хозяин со своей собакой?
- Какими породами представлен заданный клуб?
- Какие медали и сколько заслужены клубом?
- Какие эксперты обслуживают породу?
- Количество участников по каждой породе?

Выполнение работы.

В качестве ORM выбран sequelize для языка TypeScript и СУБД Postgres. Для установки зависимостей и запуска программы необходимо прописать следующие команды:

```
npm install
npm start
```

Модели:

- breed

```
@Table({
  tableName: 'breeds',
})
export class Breed extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;

  @Unique
  @AllowNull(false)
  @Column(DataType.STRING)
  name: string;
```

```
  @HasMany(() => Show)
  shows: Show[];

  @HasMany(() => Dog)
  dogs: Dog[];
}
```

- club

```
@Table({
  tableName: 'clubs',
})
export class Club extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;

  @Unique
  @AllowNull(false)
  @Column(DataType.STRING)
  name: string;
}
```

- dog

```
@Table({
  tableName: 'dogs'
})
export class Dog extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;

  @AllowNull(false)
  @Column(DataType.STRING)
  name: string;
```

```
  @AllowNull(false)
  @Column(DataType.INTEGER)
  age: number;

  @AllowNull(false)
  @ForeignKey(() => Breed)
  @Column(DataType.BIGINT)
  breed_id: number;

  @BelongsTo(() => Breed, {
    onDelete: 'CASCADE',
  })
  breed: ReturnType<() => Breed>;

  @AllowNull(false)
  @ForeignKey(() => Owner)
  @Column(DataType.BIGINT)
  owner_id: number;

  @BelongsTo(() => Owner, {
    onDelete: 'SET NULL',
  })
  owner: ReturnType<() => Owner>;

  @HasMany(() => Medal)
  medals: Medal[];
}
```

- expert

```
@Table({
  tableName: 'experts',
})
export class Expert extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;
```

```
  @AllowNull(false)
  @Column(DataType.STRING)
  name: string;

  @AllowNull(false)
  @ForeignKey(() => Club)
  @Column(DataType.BIGINT)
  club_id: number;

  @BelongsTo(() => Club, {
    onDelete: 'CASCADE',
  })
  club: ReturnType<() => Club>;

  @AllowNull(false)
  @ForeignKey(() => Ring)
  @Column(DataType.BIGINT)
  ring_id: number;

  @BelongsTo(() => Ring, {
    onDelete: 'SET NULL',
  })
  ring: ReturnType<() => Ring>;
}
```

- medal

```
@Table({
  tableName: 'medals'
})
export class Medal extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;

  @AllowNull(false)
  @Column(DataType.INTEGER)
  rank: number;
```

```
  @AllowNull(false)
  @ForeignKey(() => Breed)
  @Column(DataType.BIGINT)
  breed_id: number;

  @BelongsTo(() => Breed, {
    onDelete: 'SET NULL',
  })
  breed: ReturnType<() => Breed>;

  @AllowNull(false)
  @ForeignKey(() => Dog)
  @Column(DataType.BIGINT)
  dog_id: number;

  @BelongsTo(() => Dog, {
    onDelete: 'CASCADE',
  })
  dog: ReturnType<() => Dog>;
}
```

- owner

```
@Table({
  tableName: 'owners'
})
export class Owner extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;

  @AllowNull(false)
  @Column(DataType.STRING)
  name: string;

  @AllowNull(false)
  @Column(DataType.INTEGER)
  age: number;
```

```
  @AllowNull(false)
  @ForeignKey(() => Club)
  @Column(DataType.BIGINT)
  club_id: number;

  @BelongsTo(() => Club, {
    onDelete: 'CASCADE',
  })
  club: ReturnType<() => Club>;

  @HasMany(() => Dog)
  dogs: Dog[];
}
```

- pedigree

```
@Table({
  tableName: 'pedigrees'
})
export class Pedigree extends Model {
  @PrimaryKey
  @AllowNull(false)
  @ForeignKey(() => Dog)
  @Column(DataType.BIGINT)
  father_id: number;

  @BelongsTo(() => Dog, {
    foreignKey: 'father_id',
    onDelete: 'SET NULL',
  })
  father: ReturnType<() => Dog>;

  @PrimaryKey
  @AllowNull(false)
  @ForeignKey(() => Dog)
  @Column(DataType.BIGINT)
  son_id: number;

  @BelongsTo(() => Dog, {
```

```
    foreignKey: 'son_id',
    onDelete: 'CASCADE',
  })
  son: ReturnType<() => Dog>;
}
```

- ring

```
@Table({
  tableName: 'rings',
})
export class Ring extends Model {
  @PrimaryKey
  @AutoIncrement
  @AllowNull(false)
  @Column(DataType.BIGINT)
  declare id: number;

  @HasMany(() => Show)
  shows: Show[];
}
```

- show

```
@Table({
  tableName: 'shows',
})
export class Show extends Model {
  @PrimaryKey
  @AllowNull(false)
  @ForeignKey(() => Breed)
  @Column(DataType.BIGINT)
  breed_id: number;

  @BelongsTo(() => Breed, {
    onDelete: 'CASCADE',
  })
  breed: ReturnType<() => Breed>;

  @PrimaryKey
  @ForeignKey(() => Ring)
  @Column(DataType.BIGINT)
```

```
  ring_id: number;

  @BelongsTo(() => Ring, {
    onDelete: 'SET NULL'
  })
  ring: ReturnType<() => Ring>;

  @PrimaryKey
  @AllowNull(false)
  @Column(DataType.DATE)
  date: Date;
}
```

Код для создания и заполнения таблиц:

```
export async function insertValues() {
  await Breed.bulkCreate([
    {name: 'Labrador'},
    {name: 'Poodle'},
    {name: 'Bulldog'},
    {name: 'Pug'},
    {name: 'Chihuahua'},
  ], {returning: false});

  await Club.bulkCreate([
    {name: 'Paws and Claws'},
    {name: 'Fluffy Friends'},
    {name: 'Doggy Pals'},
    {name: 'Tails and Scales'},
    {name: 'Loyal Companions'},
  ], {returning: false});

  await Ring.bulkCreate([
    {}, {}, {}, {}, {}
  ], {returning: false});

  await Show.bulkCreate([
    {breed_id: 1, ring_id: 1, date: '2023-10-24'},
    {breed_id: 2, ring_id: 2, date: '2023-10-24'},
```

```
  {breed_id: 1, ring_id: 4, date: '2023-11-10'},
  {breed_id: 2, ring_id: 1, date: '2023-11-10'},
  {breed_id: 3, ring_id: 5, date: '2023-11-10'},
], {returning: false});

await Expert.bulkCreate([
  {name: 'John Smith', club_id: 1, ring_id: 1},
  {name: 'Alex Mercer', club_id: 3, ring_id: 1},
  {name: 'Jane Doe', club_id: 3, ring_id: 2},
  {name: 'Bob Jones', club_id: 2, ring_id: 3},
  {name: 'Sally Brown', club_id: 5, ring_id: 4},
  {name: 'Fred Bloggs', club_id: 4, ring_id: 5},
], {returning: false});

await Owner.bulkCreate([
  {name: 'Alice', age: 20, club_id: 1},
  {name: 'Bob', age: 50, club_id: 2},
  {name: 'Charlie', age: 18, club_id: 3},
  {name: 'Daisy', age: 22, club_id: 4},
  {name: 'Eve', age: 30, club_id: 5},
], {returning: false});

await Dog.bulkCreate([
  {name: 'Rex', age: 6, breed_id: 1, owner_id: 1},
  {name: 'Spot', age: 12, breed_id: 1, owner_id: 2},
  {name: 'Rover', age: 3, breed_id: 2, owner_id: 3},
  {name: 'Fido', age: 6, breed_id: 2, owner_id: 4},
  {name: 'Buddy', age: 7, breed_id: 2, owner_id: 5},
  {name: 'Duke', age: 8, breed_id: 3, owner_id: 5},
], {returning: false});

await Medal.bulkCreate([
  {rank: 1, breed_id: 1, dog_id: 2},
  {rank: 2, breed_id: 1, dog_id: 1},
  {rank: 1, breed_id: 2, dog_id: 4},
  {rank: 2, breed_id: 2, dog_id: 3},
  {rank: 3, breed_id: 2, dog_id: 5},
  {rank: 2, breed_id: 1, dog_id: 1},
```

```
    {rank: 1, breed_id: 2, dog_id: 3},
    {rank: 3, breed_id: 2, dog_id: 5},
    {rank: 2, breed_id: 3, dog_id: 6},
  ], {returning: false});

  await Pedigree.bulkCreate([
    {father_id: 2, son_id: 1},
    {father_id: 5, son_id: 3},
  ], {returning: false});
}
```

Код для выполнения первого задания:

```
  await Ring.findAll({
    attributes: ['id'],
    include: [
      {
        attributes: [],
        model: Show,
        required: true,
        include: [
          {
            attributes: [],
            model: Breed,
            required: true,
            include: [
              {
                attributes: [],
                model: Dog,
                required: true,
                include: [
                  {
                    attributes: [],
                    model: Owner,
                    required: true,
                    where: {
                      id: 5,
                    },
                  },
```

```
              ],
            },
          ],
        },
      ],
      where: {
        date: new Date('2023-11-10'),
      }
    },
  ],
}).then((res) => {
  const task = '1. На каком ринге 2023-11-10 выступает хозяин с идентификатором 5?\n';
  fs.writeFileSync('result.txt', task + JSON.stringify(res, null, 2) + '\n\n');
});
```

Ответ:

```
1. На каком ринге 2023-11-10 выступает хозяин с идентификатором 5?
[
  {
    "id": "1"
  },
  {
    "id": "5"
  }
]
```

Второе задание:

```
await Owner.findAll({
  attributes: ['club_id'],
  include: [
    {
      attributes: ['breed_id'],
      model: Dog,
      required: true,
      include: [
        {
          attributes: ['name'],
          model: Breed,
```

```
            required: true,
          },
        ],
      },
    ],
  }).then((res) => {
    const task = '2. Какими породами представлен клуб?\n';
    fs.appendFileSync('result.txt', task + JSON.stringify(res, null, 2) +
'\n\n');
  });
```

Ответ:

```
2. Какими породами представлен клуб?
[
  {
    "club_id": "1",
    "dogs": [
      {
        "breed_id": "1",
        "breed": {
          "name": "Labrador"
        }
      }
    ]
  },
  {
    "club_id": "2",
    "dogs": [
      {
        "breed_id": "1",
        "breed": {
          "name": "Labrador"
        }
      }
    ]
  },
  {
    "club_id": "3",
    "dogs": [
```

```
      {
        "breed_id": "2",
        "breed": {
          "name": "Poodle"
        }
      }
    ]
  },
  {
    "club_id": "4",
    "dogs": [
      {
        "breed_id": "2",
        "breed": {
          "name": "Poodle"
        }
      }
    ]
  },
  {
    "club_id": "5",
    "dogs": [
      {
        "breed_id": "2",
        "breed": {
          "name": "Poodle"
        }
      },
      {
        "breed_id": "3",
        "breed": {
          "name": "Bulldog"
        }
      }
    ]
  }
]
```


Третье задание:

```
  await sequelize.query('' +
    'SELECT    owners.club_id,    medals.rank,    COUNT(medals.rank)    AS
medalCount\n' +
    'FROM medals\n' +
    '    INNER JOIN dogs on medals.dog_id = dogs.id\n' +
    '    INNER JOIN owners on dogs.owner_id = owners.id\n' +
    'GROUP BY owners.club_id, medals.rank\n' +
    'ORDER      BY      owners.club_id,      medals.rank;',      {type:
QueryTypes.SELECT}).then((res) => {
    const task = '3. Какие медали и сколько заслужены клубом??\n';
    fs.appendFileSync('result.txt', task + JSON.stringify(res, null, 2) +
'\n\n');
  });
```

Ответ:

```
3. Какие медали и сколько заслужены клубом??
[
  {
    "club_id": "1",
    "rank": 2,
    "medalcount": "2"
  },
  {
    "club_id": "2",
    "rank": 1,
    "medalcount": "1"
  },
  {
    "club_id": "3",
    "rank": 1,
    "medalcount": "1"
  },
  {
    "club_id": "3",
    "rank": 2,
    "medalcount": "1"
  },
  {
    "club_id": "4",
```

```
    "rank": 1,
    "medalcount": "1"
  },
  {
    "club_id": "5",
    "rank": 2,
    "medalcount": "1"
  },
  {
    "club_id": "5",
    "rank": 3,
    "medalcount": "2"
  }
]
```

Четвёртое задание:

```
 await sequelize.query('' +
  'SELECT  DISTINCT  breeds.name,  string_agg(experts.name,  \',  \')  AS
experts\n' +
  'FROM experts\n' +
  '    INNER JOIN shows on shows.ring_id = experts.ring_id\n' +
  '    INNER JOIN breeds on shows.breed_id = breeds.id\n' +
  'GROUP BY breeds.name\n' +
  'ORDER BY breeds.name;', {type: QueryTypes.SELECT}).then((res) => {
    const task = '4. Какие эксперты обслуживают породу?\n';
    fs.appendFileSync('result.txt', task + JSON.stringify(res, null, 2) +
'\n\n');
  });
```

Ответ:

```
4. Какие эксперты обслуживают породу?
[
  {
    "name": "Bulldog",
    "experts": "Fred Bloggs"
  },
  {
    "name": "Labrador",
    "experts": "Alex Mercer, John Smith, Sally Brown"
  },
```

```
  {
    "name": "Poodle",
    "experts": "Jane Doe, Alex Mercer, John Smith"
  }
]
```

Пятое задание:

```
  await sequelize.query('' +
  'SELECT breeds.name, COUNT(dogs.id) AS dog_count\n' +
  'FROM breeds\n' +
  '    INNER JOIN dogs on breeds.id = dogs.breed_id\n' +
  'GROUP BY breeds.name\n' +
  'ORDER BY dog_count DESC, breeds.name;', {type:
QueryTypes.SELECT}).then((res) => {
    const task = '5. Количество участников по каждой породе?\n';
    fs.appendFileSync('result.txt', task + JSON.stringify(res, null, 2) +
'\n\n');
  });
```

Ответ:

```
5. Количество участников по каждой породе?
[
  {
    "name": "Poodle",
    "dog_count": "3"
  },
  {
    "name": "Labrador",
    "dog_count": "2"
  },
  {
    "name": "Bulldog",
    "dog_count": "1"
  }
]
```

Вывод.

Освоены навыки работы с ORM на примере sequelize-typescript.

ПРИЛОЖЕНИЕ А

Pull Request: https://github.com/moevm/sql-2023-1303/pull/44