

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 в СУБД PostgreSQL

Студент гр. 1303 _____ Коренев Д. А.

Преподаватель _____ Заславский М. М.

Санкт-Петербург

2023

Цель работы.

Развернуть локально PostgreSQL, написать запросы для создания и заполнения таблиц, написать запросы к БД, отвечающие на вопросы в заданиях.

Задание.

Вариант 11

Пусть требуется создать программную систему, предназначенную для работников почтового отделения. Такая система должна обеспечивать хранение сведений о подписчиках газет и журналов, обслуживаемых отделением связи, и о почтальонах. Каждое подписное издание характеризуется индексом, названием и подписной ценой. Данные о подписчиках включают в себя: фамилию, имя, отчество, домашний адрес, индексы получаемых изданий, дату, начиная с которой оформлена подписка, и срок подписки на каждое издание. Несколько домов объединяются в участок, который обслуживается одним почтальоном. Каждый почтальон может обслуживать несколько участков. В БД должны содержаться сведения о том, к каким участкам относятся подписчики газет, и об обслуживающем их почтальоне. Заведующий почтовым отделением может принять на работу и уволить почтальона, при этом участки не должны оставаться без обслуживания. Оператор почтовой связи должен иметь возможность по просьбе клиента оформить подписку, а также добавить в БД сведения о новом подписном издании. Оформление подписки связано с выдачей клиенту квитанции, в которой указывается общая стоимость подписки, что выписано, и на какой срок. Возможны следующие запросы к БД:

- Определить наименование и количество экземпляров всех изданий, получаемых отделением связи.
- По заданному адресу определить фамилию почтальона, обслуживающего подписчика.

- Какие газеты выписывает гражданин с указанной фамилией, именем, отчеством?
- Сколько почтальонов работает в почтовом отделении?
- На каком участке количество экземпляров подписных изданий максимально?
- Каков средний срок подписки по каждому изданию?

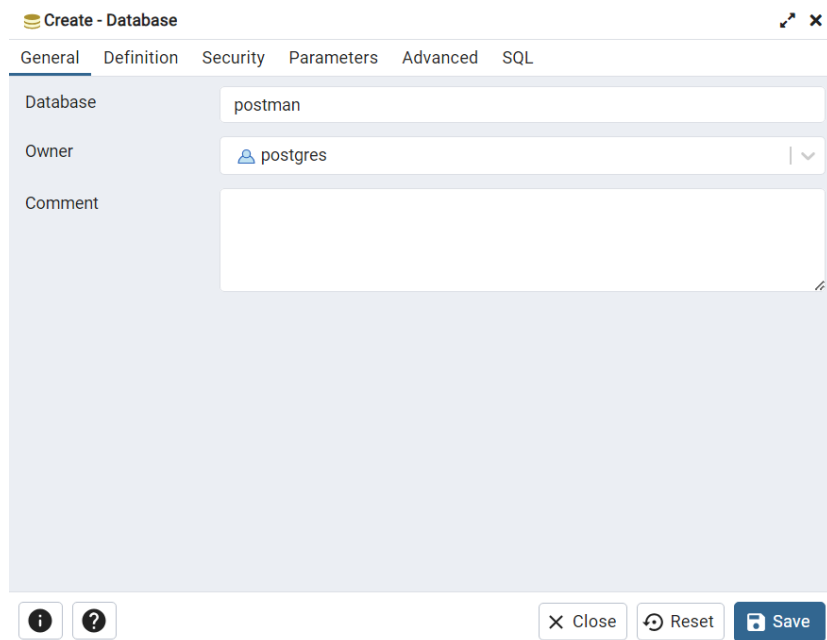
Необходимо развернуть PostgreSQL локально:

- Написать запросы для создания таблиц из предыдущей лабораторной работы
- Заполнить 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 предыдущей лабораторной работы
- Исходный код выложить на www.db-fiddle.com для проверки работоспособности
- Исходный код в виде .sql файла загрузить в виде PR в репо

Выполнение работы.

Создание базы данных:

Создана база данных «postman»



Create - Database

General Definition Security Parameters Advanced SQL

Database: postman

Owner: postgres

Comment:

Close Reset Save

Рисунок 1 Создание БД

С помощью Sequelize можно подключиться к созданной БД:

```
1 import {Sequelize} from 'sequelize';
2
3 6+ usages
4 export const sequelize :Sequelize = new Sequelize(
5   database: 'postman',
6   username: 'postgres',
7   password: '0000', options: {
8     host: 'localhost',
9     dialect: 'postgres'
10  });
```

Рисунок 2 Подключение к БД с помощью sequelize

Для создания таблиц были написаны следующие файлы:
District.js

```

4  export const District : ModelCtor<...> = sequelize.define( modelName: 'district', attributes: {
5    districtid: {
6      type: Sequelize.INTEGER,
7      primaryKey: true,
8      autoIncrement: true
9    },
10   postmanid: {
11     type: Sequelize.INTEGER,
12     allowNull: false
13   },
14   name: {
15     type: Sequelize.TEXT,
16     allowNull: false
17   }
18 });

```

Рисунок 3 District.js

House.js

```

4  export const House : ModelCtor<Model> = sequelize.define( modelName: 'house', attributes: {
5    houseid: {
6      type: Sequelize.INTEGER,
7      primaryKey: true,
8      autoIncrement: true
9    },
10   districtid: {
11     type: Sequelize.INTEGER,
12     allowNull: false
13   },
14   address: {
15     type: Sequelize.TEXT,
16     allowNull: false
17   }

```

Рисунок 4 House.js

Postman.js

```

4   export const Postman : ModelCtor<Model> = sequelize.define( modelName: 'postman', attributes: {
5       postmanid: {
6           type: Sequelize.INTEGER,
7           primaryKey: true,
8           autoIncrement: true
9       },
10      firstname: {
11          type: Sequelize.TEXT,
12          allowNull: false
13      },
14      middlename: {
15          type: Sequelize.TEXT,
16          allowNull: true
17      },
18      lastname: {
19          type: Sequelize.TEXT,
20          allowNull: true
21      }
22  });

```

Рисунок 5 Postman.js

Publication.js

```

4   export const Publication : ModelCtor<Model> = sequelize.define( modelName: 'publication', attributes: {
5       publicationid: {
6           type: Sequelize.INTEGER,
7           primaryKey: true,
8           autoIncrement: true
9       },
10      index: {
11          type: Sequelize.INTEGER,
12          allowNull: false
13      },
14      title: {
15          type: Sequelize.TEXT,
16          allowNull: false
17      },
18      price: {
19          type: Sequelize.DECIMAL(10, 2)
20      }
21  });

```

Рисунок 6 Publication.js

Subscriber.js

```

4 export const Subscriber : ModelCtor<Model> = sequelize.define( modelName: 'subscriber', attributes: {
5   subscriberid: {
6     type: Sequelize.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  houseid: {
11    type: Sequelize.INTEGER,
12    allowNull: false
13  },
14  firstname: {
15    type: Sequelize.TEXT,
16    allowNull: false
17  },
18  middlename: {
19    type: Sequelize.TEXT,
20    allowNull: true
21  },
22  lastname: {
23    type: Sequelize.TEXT,
24    allowNull: true
25  }
26 });

```

Рисунок 7 Subscriber.js

Subscription.js

```

4 export const Subscription : ModelCtor<Model> = sequelize.define( modelName: 'subscription', attributes: {
5   subscriptionid: {
6     type: Sequelize.INTEGER,
7     primaryKey: true,
8     autoIncrement: true
9   },
10  publicationid: {
11    type: Sequelize.INTEGER,
12    allowNull: false
13  },
14  subscriberid: {
15    type: Sequelize.INTEGER,
16    allowNull: false
17  },
18  startdate: {
19    type: Sequelize.DATE,
20    allowNull: false
21  },
22  duration: {
23    type: Sequelize.INTEGER,
24    allowNull: false
25  }
26 });

```

Рисунок 8 Subscription.js

Для создания ассоциаций между таблицами, заполнения их данным и выполнения необходимых запросов был создан файл Queries.js

Задание ассоциаций:

```

31  ✓ async function setAssociations() : Promise<void> {
32      await District.belongsTo(Postman, options: {foreignKey: 'postmanid'});
33      await Postman.hasMany(District, options: {foreignKey: 'postmanid'});
34
35      await House.belongsTo(District, options: {foreignKey: 'districtid'});
36      await District.hasMany(House, options: {foreignKey: 'districtid'});
37
38      await Subscriber.belongsTo(House, options: {foreignKey: 'houseid'});
39      await House.hasMany(Subscriber, options: {foreignKey: 'houseid'});
40
41      await Subscription.belongsTo(Publication, options: {foreignKey: 'publicationid'});
42      await Publication.hasMany(Subscription, options: {foreignKey: 'publicationid'});
43
44      await Subscription.belongsTo(Subscriber, options: {foreignKey: 'subscriberid'});
45      await Subscriber.hasMany(Subscription, options: {foreignKey: 'subscriberid'});
46  }

```

Рисунок 9 Queries.js

Заполнение таблиц данными:

Postman:

```

49      await Postman.bulkCreate( records: [
50          { firstname: 'Рубеус', lastname: 'Хагрид' },
51          { firstname: 'Букля' },
52          { firstname: 'Альбус', middlename: 'Персиваль', lastname: 'Дамблдор' },
53          { firstname: 'Миневра', lastname: 'МакГонагалл' },
54          { firstname: 'Северус', lastname: 'Снегг' }
55      ]).then(() : void => {
56          console.log('Значения вставлены в таблицу postman');
57      }).catch((error) : void => {
58          console.error('Ошибка вставки значений в таблицу postman:', error);
59      });

```

Рисунок 10 заполнение таблицы Postman

District:

```

61      await District.bulkCreate( records: [
62          { postmanid: 3, name: 'Хогвартс' },
63          { postmanid: 4, name: 'Косой Переулок' },
64          { postmanid: 1, name: 'Хогсмид' },
65          { postmanid: 5, name: 'Азкабан' },
66          { postmanid: 1, name: 'Литтл Уингин' },
67          { postmanid: 2, name: 'Оттери-Сент-Кэчпоул' }
68      ]).then(() : void => {
69          console.log('Значения вставлены в таблицу district');
70      }).catch((error) : void => {
71          console.error('Ошибка вставки значений в таблицу district:', error);
72      });

```

Рисунок 11 заполнение таблицы District

House:


```

74     await House.bulkCreate( records: [
75         { districtid: 3, address: 'Хогсмид, Три Метлы' },
76         { districtid: 3, address: 'Хогсмид, Волшебные палочки от Олливандера' },
77         { districtid: 5, address: 'Литтл Уингин, Тисовая улица, дом 4' },
78         { districtid: 4, address: 'Азкабан, 7 камера' },
79         { districtid: 6, address: 'Оттери-Сент-Кэчпоул, Нора' },
80         { districtid: 2, address: 'Косой Переулок, Гринготтс' },
81         { districtid: 2, address: 'Косой Переулок, Лавка Олливандера' },
82         { districtid: 1, address: 'Хогвартс, Башня Гриффиндор' }
83     ]).then(() :void => {
84         console.log('Значения вставлены в таблицу house');
85     }).catch((error) :void => {
86         console.error('Ошибка вставки значений в таблицу postman:', error);
87     });

```

Рисунок 12 заполнение таблицы House

Subscriber:

```

89     await Subscriber.bulkCreate( records: [
90         { houseid: 1, firstname: 'Мадам', lastname: 'Розмерта' },
91         { houseid: 2, firstname: 'Гаррик', lastname: 'Олливандер' },
92         { houseid: 3, firstname: 'Гарри', middlename: 'Джеймс', lastname: 'Поттер' },
93         { houseid: 4, firstname: 'Сириус', lastname: 'Блэк' },
94         { houseid: 5, firstname: 'Молли', lastname: 'Уизли' },
95         { houseid: 6, firstname: 'Гринготт' },
96         { houseid: 7, firstname: 'Джервейс', lastname: 'Олливандер' },
97         { houseid: 8, firstname: 'Гермиона', lastname: 'Грейнджер' },
98         { houseid: 8, firstname: 'Рон', lastname: 'Уизли' }
99     ]).then(() :void => {
100         console.log('Значения вставлены в таблицу subscriber');
101     }).catch((error) :void => {
102         console.error('Ошибка вставки значений в таблицу postman:', error);
103     });

```

Рисунок 13 заполнение таблицы Suscriber

Publication:

```

105     await Publication.bulkCreate( records: [
106         { index: 93, title: 'Загадки Темного Искусства', price: 10.49 },
107         { index: 89, title: 'Энциклопедия Зельеварения', price: 13.79 },
108         { index: 88, title: 'Дейли Профет', price: 10.99 },
109         { index: 96, title: 'Тайные Существа и Где Они Обитают', price: 8.99 },
110         { index: 83, title: 'Заклинания для Начинающих', price: 6.99 },
111         { index: 84, title: 'Колдовство в Повседневной Жизни', price: 9.99 },
112         { index: 91, title: 'Путеводитель по Магическим Местам', price: 12.99 }
113     ]).then(() :void => {
114         console.log('Значения вставлены в таблицу publication');
115     }).catch((error) :void => {
116         console.error('Ошибка вставки значений в таблицу postman:', error);
117     });

```

Рисунок 14 заполнение таблицы Publication

Subscription:

```

119     await Subscription.bulkCreate( records: [
120         { publicationid: 3, subscriberid: 6, startdate: '2022-11-01', duration: 365 },
121         { publicationid: 6, subscriberid: 5, startdate: '2023-03-20', duration: 300 },
122         { publicationid: 1, subscriberid: 4, startdate: '2020-06-15', duration: 90 },
123         { publicationid: 4, subscriberid: 1, startdate: '2022-04-12', duration: 1000 },
124         { publicationid: 3, subscriberid: 2, startdate: '2023-01-01', duration: 365 },
125         { publicationid: 1, subscriberid: 3, startdate: '2023-10-10', duration: 180 },
126         { publicationid: 2, subscriberid: 8, startdate: '2023-08-21', duration: 90 },
127         { publicationid: 5, subscriberid: 9, startdate: '2023-03-15', duration: 120 },
128         { publicationid: 7, subscriberid: 7, startdate: '2023-05-13', duration: 150 },
129         { publicationid: 3, subscriberid: 3, startdate: '2023-10-09', duration: 90 }
130     ]).then(() :void => {
131         console.log('Значения вставлены в таблицу subscription');
132     }).catch((error) :void => {
133         console.error('Ошибка вставки значений в таблицу postman:', error);
134     });

```

Рисунок 15 заполнение таблицы Subscription

Для выполнения задач были созданы и вызваны следующие методы

1. Определить наименование и количество экземпляров всех изданий, получаемых отделением связи.

```

137     async function task1() :Promise<void> {
138         await Publication.findAll( options: {
139             attributes: ['title']
140         }).then((titles : (Model<...>)[ ] ) :void => {
141             console.log('Titles:', titles.map((item : Model<any, TModelAttributes> ) => item.title));
142         })
143     }

```

Рисунок 16 task1

```

Titles: [
  'Загадки Темного Искусства',
  'Энциклопедия Зельеварения',
  'Дейли Профет',
  'Тайные Существа и Где Они Обитают',
  'Заклинания для Начинающих',
  'Колдовство в Повседневной Жизни',
  'Путеводитель по Магическим Местам'
]

```

Рисунок 17 результат выполнения task1

2. По заданному адресу определить фамилию почтальона, обслуживающего подписчика.

```

145 async function task2() : Promise<void> {
146     await Postman.findAll( options: {
147         attributes: ['lastname'],
148         include: {
149             model: District,
150             attributes: [],
151             required: true,
152             include: {
153                 model: House,
154                 attributes: [],
155                 where: {
156                     address: 'Хогсрид, Три Метлы'
157                 }
158             }
159         },
160         limit: 1
161     }).then((postman : (Model<...>[]) ) : void => {
162         console.log('Postman lastname:', postman.map((item : Model<any, TModelAttributes> ) => item.lastname));
163     })
164 }

```

Рисунок 18 task2

```
Postman lastname: [ 'Хагрид' ]
```

Рисунок 19 результат выполнения task2

3. Какие газеты выписывает гражданин с указанной фамилией, именем, отчеством?

```

166 async function task3() : Promise<void> {
167     await Publication.findAll( options: {
168         attributes: ['title'],
169         include: {
170             model: Subscription,
171             attributes: [],
172             required: true,
173             include: {
174                 model: Subscriber,
175                 attributes: [],
176                 required: true,
177                 where: {
178                     firstname: 'Гарри',
179                     middlename: 'Джеймс',
180                     lastname: 'Поттер'
181                 }
182             }
183         }
184     }).then((titles : (Model<...>[]) ) : void => {
185         console.log('Titles:', titles.map((item : Model<any, TModelAttributes> ) => item.title));
186     })
187 }

```

Рисунок 20 task3

```
Titles: [ 'Загадки Темного Искусства', 'Дейли Профет' ]
```

Рисунок 21 результат выполнения task3

4. Сколько почтальонов работает в почтовом отделении?

```

189   async function task4() : Promise<void> {
190       await Postman.count().then((count : number ) : void => {
191           console.log('Count:', count);
192       })
193   }

```

Рисунок 22 task4

Count: 5

Рисунок 23 результат выполнения task4

5. На каком участке количество экземпляров подписных изданий максимально?

```

195   async function task5() : Promise<void> {
196       var cnt : number = 0
197       await District.findAll( options: {
198           attributes: ['name', [sequelize.fn( fn: 'COUNT', sequelize.col( col: 'houses.subscribers.subscriptions.subscriptionid')), 'subscription_count']],
199           group: ['district.districtid'],
200           include: {
201               model: House,
202               attributes: [],
203               required: true,
204               include: {
205                   model: Subscriber,
206                   attributes: [],
207                   required: true,
208                   include: {
209                       model: Subscription,
210                       attributes: [],
211                       required: true
212                   }
213               }
214           },
215           order: [[sequelize.fn( fn: 'COUNT', sequelize.col( col: 'houses.subscribers.subscriptions.subscriptionid')), 'DESC']],
216       }).then((districts : (Model<any>)[ ] ) : void => {
217           districts.map((district : Model<any, TModelAttributes> ) : number => cnt = Math.max(cnt, parseInt(district.get( key: 'subscription_count'))))
218       })
219   }

```

Рисунок 24 task5 начало

```

await District.findAll( options: {
    attributes: ['name', [sequelize.fn( fn: 'COUNT', sequelize.col( col: 'houses.subscribers.subscriptions.subscriptionid')), 'subscription_count']],
    group: ['district.districtid'],
    having: where (
        sequelize.fn( fn: 'COUNT', sequelize.col( col: 'houses.subscribers.subscriptions.subscriptionid')), cnt
    ),
    include: {
        model: House,
        attributes: [],
        required: true,
        include: {
            model: Subscriber,
            attributes: [],
            required: true,
            include: {
                model: Subscription,
                attributes: [],
                required: true
            }
        }
    }
}).then((districts : (Model<any>)[ ] ) : void => {
    console.log('Districts:', districts.map((item : Model<any, TModelAttributes> ) => item.name));
})
}

```

Рисунок 25 task5 конец

Districts: ['Косой Переулок', 'Хогсמיד', 'Литтл Уингин', 'Хогвартс']

Рисунок 26 результат выполнения task5

6. Каков средний срок подписки по каждому изданию?

а. Для всех существующих подписок:

```
async function task6() : Promise<void> {
  await Subscription.findOne({
    attributes: [[sequelize.fn( fn: 'AVG', sequelize.col( col: 'duration' )), 'avg_duration']]
  }).then((duration : Model<...> | null ) : void => {
    console.log('Avg duration:', duration.get( key: 'avg_duration'));
  })
}
```

Рисунок 27 task6

A terminal window with a dark background showing the output of the task6 function. The text "Avg duration: 275" is displayed in a light blue, monospaced font.

Рисунок 28 результат выполнения task6

б. Только для действующих подписок:

```
async function task6_1() : Promise<void> {
  await Subscription.findOne({
    attributes: [[sequelize.fn( fn: 'AVG', sequelize.col( col: 'duration' )), 'avg_duration']],
    where: {
      startdate: {
        [Op.gte]: sequelize.literal( val: 'NOW() - interval \'1 day\' * duration')
      }
    }
  }).then((duration) : void => {
    console.log('Avg duration:', duration.get('avg_duration'));
  })
}
```

Рисунок 29 task6_1

A terminal window with a dark background showing the output of the task6_1 function. The text "Avg duration: 337" is displayed in a light blue, monospaced font.

Рисунок 30 результат выполнения task6_1

Выводы.

В данной лабораторной работе освоена работа с ORM для Node.js – Sequelize.

ПРИЛОЖЕНИЕ А

ССЫЛКИ

Pull Request: <https://github.com/moevm/sql-2023-1303/pull/50>