

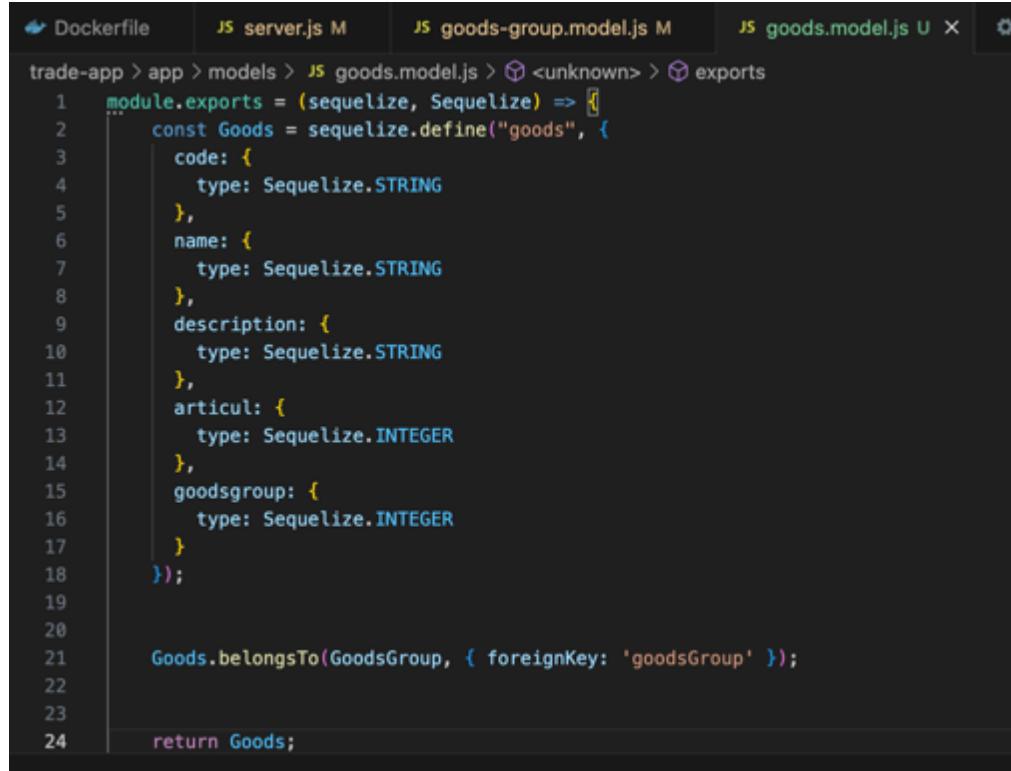
Лабораторная работа 10

Формирование связей сущностей

2.6 Доработать остальные таблицы БД

2.6.1 Товар

Добавим в модель файл app/models/goods.model.js



```
trade-app > app > models > JS goods.model.js > <unknown> > exports
1 module.exports = (sequelize, Sequelize) => [
2   const Goods = sequelize.define("goods", {
3     code: {
4       type: Sequelize.STRING
5     },
6     name: {
7       type: Sequelize.STRING
8     },
9     description: {
10       type: Sequelize.STRING
11     },
12     articul: {
13       type: Sequelize.INTEGER
14     },
15     goodsgroup: {
16       type: Sequelize.INTEGER
17     }
18   );
19
20   Goods.belongsTo(GoodsGroup, { foreignKey: 'goodsGroup' });
21
22
23
24   return Goods;
25 ];
```

Зарегистрируем модель в файле app/models/index.js



```
20 db.Sequelize = Sequelize;
21 db.sequelize = sequelize;
22
23 db.goodsGroup = require("./goods-group.model.js")(sequelize, Sequelize)
24 db.goods = require("./goods.model.js")(sequelize, Sequelize)
25
```

Создадим новый файл app/models/references.model.js для определения связей между сущностями приложения.



```
trade-app > app > models > JS references.model.js > <unknown> > exports
1 module.exports = (db) => {
2   //GoodsGroup references
3   db.goodsGroup.belongsTo(db.goodsGroup, { foreignKey: 'baseGoodsGroup' });
4
5   //Goods references
6   db.goods.belongsTo(db.goodsGroup, { foreignKey: 'goodsGroup' });
7 };
8
```

NOTE!!

Дополнительно прочитать про виды связей сущностей в фреймворке Sequelize можно по ссылке <https://sequelize.org/docs/v6/core-concepts/assocs/>

В частности,

Отношения «один ко многим»

Философия

Связи «один ко многим» связывают один источник с несколькими целевыми объектами, при этом все эти целевые объекты связаны только с этим единственным источником.

Это означает, что, в отличие от связи «один к одному», в которой нам приходилось выбирать, где будет расположен внешний ключ, в связях «один ко многим» есть только один вариант. Например, если у одного объекта Foo есть несколько объектов Bar (и, таким образом, каждый объект Bar принадлежит одному объекту Foo), то единственна разумная реализация — наличие столбца `fooid` в таблице Bar. Обратное невозможно, поскольку у одного объекта Foo есть несколько объектов Bar.

Цель

В этом примере у нас есть модели Team и Player. Мы хотим сообщить Sequelize, что между ними существует связь «один ко многим», то есть у одной команды есть несколько игроков, в то время как каждый игрок принадлежит только одной команде.

Реализация

Основной способ сделать это следующий:

```
Team.hasMany(Player); Player.belongsTo(Team);
```

Например, в PostgreSQL при выполнении `sync()` указанная выше конфигурация выдаст следующий SQL-запрос:

```
CREATE TABLE IF NOT EXISTS "Teams" (
/* ... */
);
CREATE TABLE IF NOT EXISTS "Players" (
/* ... */
"TeamId" INTEGER REFERENCES "Teams" ("id") ON DELETE SET NULL ON UPDATE CASCADE,
/* ... */
);
```

Добавим вызов файла связей в `app/models/index.js`

```
20 db.Sequelize = Sequelize;
21 db.sequelize = sequelize;
22
23 db.goodsGroup = require("./goods-group.model.js")(sequelize, Sequelize)
24 db.goods = require("./goods.model.js")(sequelize, Sequelize)
25
26 require('./references.model.js')(db)
27
28
```

Добавить файлы в репозиторий

```
git add .
```

Выполнить комит изменений

```
git commit -am "goods model added"
```

6.2.2 Список цен

Добавить файл app/models/pricelist.model.js

```
JS index.js M JS pricelist.model.js M X JS references.model.js
trade-app > app > models > JS pricelist.model.js > <unknown> > exports > Pricelist
1  const { DataTypes } = require('sequelize');
2
3  module.exports = (sequelize, Sequelize) => {
4    const Pricelist = sequelize.define("pricelist", {
5      category: {
6        type: Sequelize.STRING
7      },
8      date: {
9        type: DataTypes.DATE
10     }
11   });
12
13
14   return Pricelist;
15 }
```

NOTE!!

Мы использовали вспомогательный класс с описание типов данных для работы с типом данных DATA DataTypes.DATE

Зарегистрировать модель в app/models.index.js

```
20 db.Sequelize = Sequelize;
21 db.sequelize = sequelize;
22
23 db.goodsGroup = require("./goods-group.model.js")(sequelize, Sequelize)
24 db.goods = require("./goods.model.js")(sequelize, Sequelize)
25 db.pricelist = require("./pricelist.model.js")(sequelize, Sequelize)
26 |
27 require('./references.model.js')(db)
28
```

NOTE!!

Поскольку сущность pricelist не содержит внешних связей, то изменять файл app/models/references.js не нужно.

Добавить файлы в репозиторий

```
git add .
```

Выполнить комит изменений

```
git commit -am "pricelist model added"
```

6.2.3 Товар в списке цен

Добавить файл app/models/pricelistgoods.model.js

```
JS index.js M JS pricelistgoods.model.js M X JS references.model.js M
trade-app > app > models > JS pricelistgoods.model.js > ✎ <unknown> > ✎ exports > [x] PricelistGoods > ↗ goods
  1  const { DataTypes } = require('sequelize');
  2
  3  module.exports = (sequelize, Sequelize) => {
  4    const PricelistGoods = sequelize.define("pricelistgoods", {
  5      price: {
  6        type: DataTypes.DOUBLE
  7      },
  8      priceList: {
  9        type: Sequelize.INTEGER
 10      },
 11      goods: {
 12        type: Sequelize.INTEGER
 13      }
 14    });
 15
 16    return PricelistGoods;
 17  };

```

Зарегистрировать файл модели в app/models/index.js

```
18  const db = {};
19
20  db.Sequelize = Sequelize;
21  db.sequelize = sequelize;
22
23  db.goodsGroup = require("./goods-group.model.js")(sequelize, Sequelize)
24  db.goods = require("./goods.model.js")(sequelize, Sequelize)
25  db.pricelist = require("./pricelist.model.js")(sequelize, Sequelize)
26  db.pricelistGoods = require("./pricelistgoods.model.js")(sequelize, Sequelize)
27
28  require('./references.model.js')(db)
29
30
```

Добавить связи в файл app/models/references.model.js

```
JS index.js M JS pricelistgoods.model.js M X JS references.model.js M
trade-app > app > models > JS references.model.js > ✎ <unknown> > ✎ exports > ↗ foreignKey
  1
  2  module.exports = (db) => {
  3
  4    //GoodsGroup references
  5    db.goodsGroup.belongsTo(db.goodsGroup, { foreignKey: 'baseGoodsGroup' });
  6
  7    //Goods references
  8    db.goods.belongsTo(db.goodsGroup, { foreignKey: 'goodsGroup' });
  9
 10   //pricelistgoods
 11   db.pricelistGoods.belongsTo(db.pricelist, { foreignKey: 'pricelist' });
 12   db.pricelistGoods.belongsTo(db.goods, { foreignKey: 'goods' });
 13
 14 };

```

Добавить файлы в репозиторий

```
git add .
```

Выполнить комит изменений

```
git commit -am "pricelistgoods model added"
```

6.2.4 Продажа

Добавить файл модели app/models/purchase.model.js

```
JS index.js M JS references.model.js M JS purchase.model.js M X
trade-app > app > models > JS purchase.model.js > ⚡ <unknown> > ⚡ exports > ⚡ Purchase > ⚡ priceList
  1 const { DataTypes } = require('sequelize');
  2
  3 module.exports = (sequelize, Sequelize) => {
  4   const Purchase = sequelize.define("purchase", [
  5     purchaseDate: {
  6       type: DataTypes.DATE
  7     },
  8     invoiceDate: {
  9       type: DataTypes.DATE
 10     },
 11     pricelist: {
 12       type: Sequelize.INTEGER
 13     }
 14   ]);
 15
 16   return Purchase;
 17 };
 18
```

Зарегистрировать модель продажи app/models/index.js

```
18 const db = {};
19
20 db.Sequelize = Sequelize;
21 db.sequelize = sequelize;
22
23 db.goodsGroup = require("./goods-group.model.js")(sequelize, Sequelize)
24 db.goods = require("./goods.model.js")(sequelize, Sequelize)
25 db.pricelist = require("./pricelist.model.js")(sequelize, Sequelize)
26 db.pricelistGoods = require("./pricelistgoods.model.js")(sequelize, Sequelize)
27 db.purchase = require("./purchase.model.js")(sequelize, Sequelize)
28
29 require('../references.model.js')(db)
30
```

Добавить связи в файл app/models/references.model.js

```
JS index.js M JS references.model.js M X JS purchase.model.js M
trade-app > app > models > JS references.model.js > ⚡ <unknown> > ⚡ exports
  1
  2 module.exports = (db) => [
  3
  4   //GoodsGroup references
  5   db.goodsGroup.belongsTo(db.goodsGroup, { foreignKey: 'baseGoodsGroup' });
  6
  7   //Goods references
  8   db.goods.belongsTo(db.goodsGroup, { foreignKey: 'goodsGroup' });
  9
 10   //pricelistgoods
 11   db.pricelistGoods.belongsTo(db.pricelist, { foreignKey: 'priceList' });
 12   db.pricelistGoods.belongsTo(db.goods, { foreignKey: 'goods' });
 13
 14   //purchase
 15   db.purchase.belongsTo(db.pricelist, { foreignKey: 'priceList' });
 16 ];
```

Добавить файлы в репозиторий

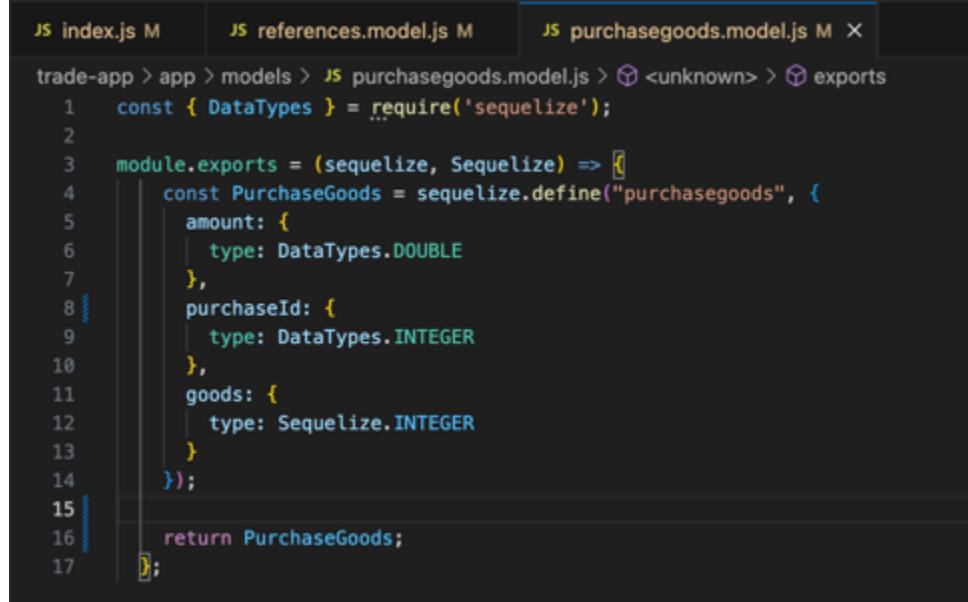
```
git add .
```

Выполнить комит изменений

```
git commit -am "purchase model added"
```

6.2.5 Товар в продаже

Создадим файл модели app/models/purchasegoods.model.js



```
JS index.js M JS references.model.js M JS purchasegoods.model.js M X
trade-app > app > models > JS purchasegoods.model.js > ⌂ <unknown> > ⌂ exports
1 const { DataTypes } = require('sequelize');
2
3 module.exports = (sequelize, Sequelize) => [
4   const PurchaseGoods = sequelize.define("purchasegoods", {
5     amount: {
6       type: DataTypes.DOUBLE
7     },
8     purchaseId: {
9       type: DataTypes.INTEGER
10    },
11     goods: {
12       type: Sequelize.INTEGER
13     }
14   });
15
16   return PurchaseGoods;
17 ];
```

Зарегистрировать модель в файле app/models/index.js



```
18 const db = {};
19
20 db.Sequelize = Sequelize;
21 db.sequelize = sequelize;
22
23 db.goodsGroup = require("./goods-group.model.js")(sequelize, Sequelize)
24 db.goods = require("./goods.model.js")(sequelize, Sequelize)
25 db.pricelist = require("./pricelist.model.js")(sequelize, Sequelize)
26 db.pricelistGoods = require("./pricelistgoods.model.js")(sequelize, Sequelize)
27 db.purchase = require("./purchase.model.js")(sequelize, Sequelize)
28 db.purchaseGoods = require("./purchasegoods.model.js")(sequelize, Sequelize)
29
30 require('./references.model.js')(db)
31
```

Добавить связи в файл app/models/references.model.js

```
JS index.js M JS references.model.js M X JS purchasegoods.model.js M
trade-app > app > models > JS references.model.js >  <unknown> >  exports >  foreignKey
1
2 module.exports = (db) => {
3
4     //GoodsGroup references
5     db.goodsGroup.belongsTo(db.goodsGroup, { foreignKey: 'baseGoodsGroup' });
6
7     //Goods references
8     db.goods.belongsTo(db.goodsGroup, { foreignKey: 'goodsGroup' });
9
10    //pricelistgoods
11    db.pricelistGoods.belongsTo(db.pricelist, { foreignKey: 'priceList' });
12    db.pricelistGoods.belongsTo(db.goods, { foreignKey: 'goods' });
13
14    //purchase
15    db.purchase.belongsTo(db.pricelist, { foreignKey: 'priceList' });
16
17    //purchasegoods
18    db.purchaseGoods.belongsTo(db.purchase, { foreignKey: 'purchaseId' });
19    db.purchaseGoods.belongsTo(db.goods, { foreignKey: 'goods' });
20
21};
```

Добавить файлы в репозиторий

```
git add .
```

Выполнить комит изменений

```
git commit -am "purchasegoods model added"
```

6.3 Запуск приложения

6.3.1 Синхронизировать данные в github

```
git push -u origin main
```

6.3.2 Запуск приложения

```
docker-compose up -d --build
```

6.4 Проверка корректности созданных сущностей и связей в БД

Подключиться клиентом БД к самой БД и проверить наличие сущностей и связей

> Operators
> Procedures
> 1.3 Sequences
▼ Tables (6)
 > goods
 > goodsgroups
 > pricelistgoods
 > pricelists
 > purchasegoods
 > purchases
> Trigger Functions
> Types
> Views

Проверяем связи

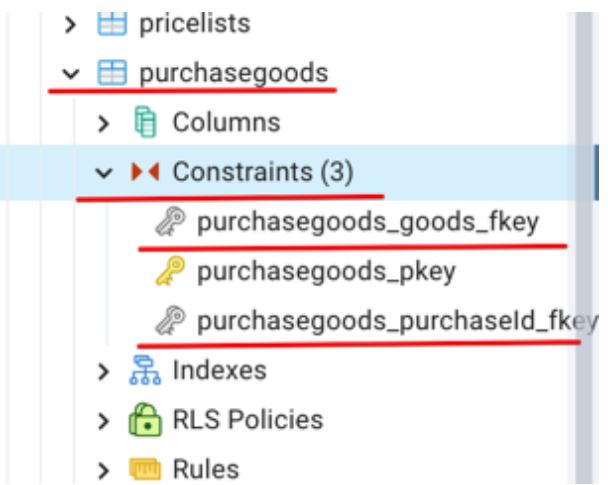
Внешний ключ в таблице Goods

▼ Tables (6)
 ▼ goods
 > Columns
 ▼ Constraints (2)
 goods_goodsGroup_fkey
 goods_pkey
 > Indexes
 > RLS Policies
 > Rules
 > Triggers

Внешний ключ в таблице purchase

▼ purchases
 > Columns
 ▼ Constraints (2)
 purchases_pkey
 purchases_priceList_fkey
 > Indexes
 > RLS Policies

Внешние ключи в таблице purchasegoods



Задание для самостоятельного выполнения:

Определить связи для сущностей своего проекта