




DAY 2

План

-  JOINS
-  Что такое ORM?
-  Sequelize ORM

JOIN's

Что такое JOIN?

JOIN — объединение нескольких таблиц из базы данных в одну с использованием первичных и внешних ключей.

```
SELECT * FROM users  
JOIN posts ON posts.user_id = users.id  
WHERE users.name = "Антон"
```

Обратите внимание, что в результат попадают только такие строки, у которых есть значения и в `posts.user_id` и в `users.id`.

LEFT JOIN vs. RIGHT JOIN

Существует несколько видов **JOIN**, они отличаются тем, как взаимодействуют с отсутствующими значениями.

При использовании **LEFT JOIN** в результат попадут все значения из левой таблицы.

LEFT JOIN vs. RIGHT JOIN

При использовании **RIGHT JOIN** в результат попадут все значения из правой таблицы.

```
SELECT * FROM users  
RIGHT JOIN posts ON posts.user_id = users.id  
WHERE users.name = "Антон"
```

Даже если **posts.user_id** будет **NULL** — в результат попадут все строки из **users**.

ORM & Sequelize

ORM, object-relational mapping

ORM (объектно-реляционное отображение) связывает базу данных с объектно-ориентированным кодом.

С помощью ORM можно:

- добавлять объекты в БД
- получать объекты из БД
- удалять из БД
- обновлять в БД

Sequelize ORM

Библиотека для Node.js, основана на промисах.

Это ORM для PostgreSQL и других СУБД (MySQL, MariaDB, SQLite и MS SQL).

Установка Sequelize

```
npm i sequelize pg pg-hstore
```

Кроме самой библиотеки sequelize нужно установить драйверы для СУБД, которую планируем использовать.

В нашем случае это драйверы PostgreSQL — pg и pg-hstore.

Подключение, вариант 1

```
const { Sequelize } = require('sequelize');
```

```
const sequelize = new Sequelize('dbname', 'dbuser', 'pass', {  
  host: 'localhost',  
  dialect: 'postgres',  
});
```

Подключение, вариант 2 (лучше)

```
npm i sequelize-cli  
npx sequelize init
```

CLI-утилита для работы с Sequelize ORM.

Помогает создать и настроить конфигурацию подключения, а также работу с моделями, миграциями, седами (seeders).

Подключение, вариант 2

config/config.json

```
{  
  "development": {  
    "username": "dbuser",  
    "password": "pass",  
    "database": "dbname",  
    "host": "127.0.0.1",  
    "dialect": "postgres"  
  },  
}
```

Подключение, вариант 2

```
const { sequelize } = require('./models');
```

Выполнение “сырого” запроса

```
const [results, metadata] = await sequelize.query(  
    "SELECT * FROM students"  
);
```

```
console.log(results);  
console.log(metadata);
```

Получение результатов без метаданных

```
const results = await sequelize.query(  
  "SELECT * FROM students",  
  { type: QueryTypes.SELECT }  
  // нужно подтянуть QueryTypes из sequelize  
);  
  
console.log(results);
```


Замена параметров через массив

Можно передать массив `replacements`, значения из которого заменят знаки вопроса (параметры) в “сыром” запросе. Значения будут экранированы:

```
await sequelize.query(  
  'SELECT * FROM projects WHERE status = ? AND lang = ?;',  
  {  
    replacements: ['active', 'js'],  
    type: QueryTypes.SELECT  
  }  
);
```

Замена параметров через объект

Если `replacements` передать в виде объекта, можно использовать его ключи как именованные параметры:

```
await sequelize.query(  
  'SELECT * FROM projects WHERE status = :status AND lang =  
  :lang;',  
  {  
    replacements: { status: 'active', lang: 'js' },  
    type: QueryTypes.SELECT  
  }  
);
```