Лабораторная работа 18

ПСКП

ПОИТ-3

**Задание 01**

1. Ознакомьтесь с диаграммой базы данных (БД)



1. Параметры SQL-сервера: ***172.16.193.223, student/fitfit***
2. Скрипт для создания и заполнения таблиц базы данных ***C:/CommonSpace/10.sql***.
3. Создайте собственный экземпляр БД с именем: **XYZ**, где XYZ – инициалы студента. После сдачи лабораторной работы ***БД должна обязательно быть удалена***.

**Задание 02**

1. Разработайте приложение **18-01**, представляющее собой HTTP-сервер, прослушивающий порт ***3000***. Сервер должен обрабатывать запросы, описанные в следующих таблицах. **Для выполнения SQL-запросов используйте пакет sqquelize**.

|  |  |  |
| --- | --- | --- |
| Метод  запроса | URI | Назначение |
| GET | / | получить статический HTML-файл, назначение файла описывается в задании 03 |
| GET | api/faculties | получить список всех факультетов в json-формате |
| GET | api /pulpits | получить список всех кафедр в json-формате |
| GET | api/subjects | получить список всех учебных дисциплин в json-формате |
| GET | api/auditoriumstypes | получить список всех типов учебных аудиторий в json-формате |
| GET | api/auditorims | получить список всех учебных аудиторий в json-формате |
| POST | api/faculties | добавить новый факультет, данные в json-формате |
| POST | api/pulpits | добавить новую кафедру, данные в json-формате |
| POST | api/subjects | добавить новую учебную дисциплину, данные в json-формате |
| POST | api/auditoriumstypes | добавить новый тип учебной дисциплины, данные в json-формате |
| POST | api/auditoriums | добавить новую учебную аудиторию, данные в json-формате |
| PUT | api/faculties | корректировать информацию о факультете, данные в json-формате |
| PUT | api/pulpits | корректировать информацию о кафедре, данные в json-формате |
| PUT | api/subjects | корректировать информацию об учебной дисциплине, данные в json-формате |
| PUT | api/auditoriumstypes | корректировать информацию о типе учебной аудитории, данные в json-формате |
| PUT | /auditorims | корректировать информацию об учебной дисциплине, данные в json-формате |
| DELETE | api/faculties/xyz | xyz - код факультета, удалить факультет с кодом xyz |
| DELETE | api/pulpits/xyz | xyz - код кафедры, удалить кафедру с кодом xyz |
| DELETE | api/subjects/xyz | xyz - код дисциплины, удалить дисциплину с кодом xyz |
| DELETE | api/auditoriumtypes/xyz | xyz - код типа аудитории, удалить тип аудитории с кодом xyz |
| DELETE | api/auditorims/xyz | xyz - код аудитории, удалить аудиторию с кодом xyz |

|  |  |
| --- | --- |
| Метод  запроса | Назначение |
| GET | Возвращает HTML-файл, найденные в БД данные или сообщение об ошибке в json-формате |
| POST | принимает данные в json-формате,  возвращает добавленные в БД данные или сообщение об ошибке в json-формате |
| PUT | принимает данные в json-формате,  возвращает измененные в БД данные или сообщение об ошибке в json-формате |
| DELETE | возвращает удаленные в БД данные или сообщение об ошибке в json-формате |

1. Для хранения данных приложение **18-01** должно использовать БД **XYZ.** Для соединения с БД сервердолжен применять пул соединений.
2. Для проверки работоспособности приложения, используйте **POSTMAN**.

**Задание 03** Ответьте на следующие вопросы

1. Поясните понятие «ORM»

**ORM** – это аббревиатура для **Object Relational Mapping (Объектно-реляционное отображение)**. Это такая методика или техника программирования, предназначенная для преобразования между **несовместимыми типами данных в объектно-ориентированных языках программирования.** Суть этого достаточно сложного определения заключается в том, что создается некая абстракция - “**виртуальная объектная база**”, запросы к которой, преобразуются в **SQL команды,** т.е. больше не нужно писать **SQL-запросы к базе данных вручную** (хотя в некоторых, достаточно нетривиальных случаях, писать запрос вручную все-таки придется, благо таких случаев не так много).

**ORM фреймворк** может быть написан на каком-либо объектно-ориентированном языке ( **PHP, Python, Ruby** ) и представлять обертку над некой реляционной базой данных. **Классы будут соответствовать таблицам в базе, а экземпляры этих классов – конкретным строкам таблицы**.

1. Какой дополнительный пакет необходимо установить для работы sequelize c Microsoft SQL Server?

npm i tedious

1. Поясните понятие «sequelize-схема».

Сейчас у нас есть схемы табличек, но нет базы данных, для того, чтобы sequelize синхронизировал схему данных с реальной (но пока не созданной) БД

1. Какая функция sequelize позволяет выполнить SELECT-запрос?

## **Поиск по ID**

let data = await sequelize.models.posts.findOne({ where: { id: 1 } });

console.log(data.dataValues);

При использовании findOne вернётся либо undefined, если ничего не найдено, либо объект с данными.

## **Поиск всех записей по параметрам**

Метод вернёт массив. В случае, если ничего не найдено- пустой массив.

let data = await sequelize.models.posts.findAll();

Комбинируем условия(where) и лимит+офсет

let data = await sequelize.models.posts.findAll({ where: {}, limit: 2, offset: 1});

1. Какая функция sequelize позволяет выполнить INSERT-запрос?

## **Создание записей**

Добавим новый пост:

let newPost = {

title: 'Post title 1',

body: 'Ololo ololo ya voditel NLO'

}

let newDBRecord = await sequelize.models.posts.create(newPost);

В newDBRecord будет содержаться новая запись, которую вернула БД.

И то же самое для таблички комментов:

let newComment = {

posts\_id: 1,

userName: 'Ololoev',

comment: 'Preved medved'

}

let newDBComments = await sequelize.models.comments.create(newComment);

1. Какая функция sequelize позволяет выполнить UPDATE-запрос?

## **Обновление данных**

let newDBComments = await sequelize.models.comments.update({ userName: 'Ololoev\_O' }, { where: { id: 1 } });

В качестве параметров НЕ нужно передавать полный объект, а достаточно передать только те поля, именно которые требуется обновить. Условия можно использовать любые, как и в findAll/findOne

1. Какая функция sequelize позволяет выполнить DELETE-запрос?

## **Удаление записей**

await sequelize.models.posts.destroy({ where: { id: 1 } });

1. Поясните назначение функции sequelize *Схема*.**hasMany**.

Например, одна компания может производить множество товаров. То есть мы имеем отношение одни-ко-многим (1 компания - много товаров). Для содания подобной связи в Sequelize применяется метод **hasMany()**

Метод **hasMany()** вызывается именно у главной модели - Company. А в сам метод передается зависимая модели - Product.

1. Поясните понятие «raw query».

*const [results, metadata] = await sequelize.query("UPDATE users SET y = 42 WHERE x = 12");*

// Results will be an empty array and metadata will contain the number of affected rows.

1. Поясните понятие «code first».
2. Поясните понятие «транзакция».

let transaction;

try {

// get transaction

transaction = await sequelize.transaction();

// step 1

await Model.destroy({where: {id}, transaction});

// step 2

await Model.create({}, {transaction});

// commit

await transaction.commit();

} catch (err) {

// Rollback transaction if any errors were encountered

await transaction.rollback();

}