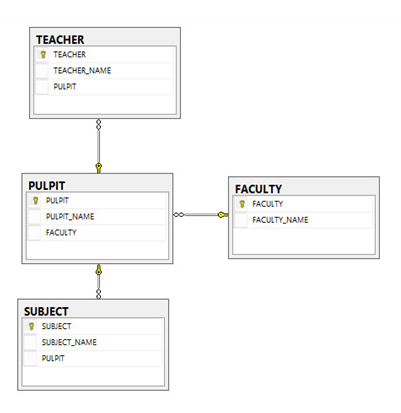
Лабораторная работа 16

ПСКП

ПОИТ-3

**Задание 01**

1. Ознакомьтесь с литературой <https://diskstation.belstu.by:5001> Для\_студентов\_ФИТ\_БГТУ\ЛИТЕРАТУРА\GRAPHQL
2. 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 ***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 GrpaphQL-схему, позволяющую выполнять следующие запросы к БД.

|  |  |  |
| --- | --- | --- |
| Наименование | Параметры | Запрос |
| getFaculties | faculty | Если faculty не задан, то формируется список всех факультетов.  Если faculty задан, то формируются данные по заданному факультету. |
| getTeachers | teacher | Если teacher не задан, то формируется список всех преподавателей.  Если teacher задан, то формируются данные по заданному преподавателю. |
| getPulpits | pulpit | Если pulpit не задан, то формируется список всех кафедр.  Если pulpit задан, то формируются данные по заданной кафедре. |
| getSubjects | subject | Если subject не задан, то формируется список всех дисциплин.  Если subject задан, то формируются данные по заданной дисциплине. |
| setFaculty | faculty | Если факультет с кодом faculty.faculty уже существует, то заменить.  Если факультет с кодом faculty.faculty не существует, то добавить новый факультет.  Сформировать данные о добавленном/измененном факультете. |
| setTeacher | teacher | Если преподаватель с кодом teacher.teacher уже существует, то заменить.  Если преподаватель с кодом teacher.teacher не существует, то добавить нового преподавателя.  Сформировать данные о добавленном/измененном преподавателе. |
| setPulpit | pulpit | Если кафедра с кодом pulpit.pulpit уже существует, то заменить.  Если кафедра с кодом pulpit.pulpit не существует, то добавить новую кафедру.  Сформировать данные о добавленной/измененной кафедре. |
| setSubject | subject | Если дисциплина с кодом subject.subject уже существует, то заменить.  Если кафедра с кодом subject.subject не существует, то добавить новую дисциплину.  Сформировать данные о добавленной/измененной дисциплине. |
| delFaculty | faculty | Если факультет с кодом faculty.faculty существует, то удалить и сформировать true.  Если факультет с кодом faculty.faculty не существует, то сформировать false. |
| delTeacher | teacher | Если преподаватель с кодом teacher.teacher существует, то удалить и сформировать true.  Если преподаватель с кодом teacher.teacher не существует, то сформировать false. |
| delPulpit | pulpit | Если кафедра с кодом pulpit.pulpit существует, то удалить и сформировать true.  Если кафедра с кодом pulpit.pulpit не существует, то сформировать false. |
| delSubject | subject | Если дисциплина с кодом subject.subject существует, то удалить и сформировать true.  Если кафедра с кодом subject.subject не существует, то сформировать false. |
| getTeachersByFaculty | faculty | Сформировать список всех преподавателей заданного факультета faculty. |
| getSubjectsByFaculties | faculty | Сформировать список всех кафедр и прикрепленных к ним дисциплин заданного факультета faculty. |

1. Разработайте приложение **16-01**, представляющее собой HTTP-сервер, прослушивающий порт ***3000***. Сервер должен обрабатывать запросы, описанные в представленной выше таблице. Данные должны сохраняться в БД MSSQL. (п.3).

**Задание 02** Ответьте на следующие вопросы

1. Поясните понятие **GraphQL**.
2. Поясните понятие схема **GraphQL**.
3. Расшифруйте аббревиатуру **SDL GraphQL**.
4. Поясните понятие **resolver** **GraphQL**.
5. Поясните понятие **query GraphQL**.
6. Поясните понятие **mutation** **GraphQL**.
7. Поясните понятие **subscription GraphQL**.
8. Поясните понятие **context GraphQL**.
9. Поясните схему работы модуля **graphql**
10. Поясните следующие компоненты **interface, enum, fragment, union схемы GraphQL.**
11. Поясните понятие **GraphQL**.

это язык запросов, используемый клиентскими приложениями для работы с данными;

синтаксис, который описывает как запрашивать данные

1. Поясните понятие схема **GraphQL**.

это то, что позволяет организовывать создание, чтение, обновление и удаление данных в вашем приложении

1. Поясните понятие **resolver** **GraphQL**.

это распознаватель, который объясняет действие при запросе

1. Поясните понятие **query GraphQL**.

выборка данных (get)

1. Поясните понятие **mutation** **GraphQL**.

изменение бд (create, update, delete)

1. Поясните понятие **subscription GraphQL**.

подписка, полностью аналогичны queries и все что применимо к queries, подходит для subscriptions

1. Поясните понятие **context GraphQL**.

механизм для передачи информации каждому преобразователю resolve

1. Поясните схему работы модуля **graphql**

GraphQL-клиент отправляет запрос на получение данных или на их изменение, составленный в соответствии со схемой, на GraphQL-сервер. GraphQL-сервер, в свою очередь, представляет собой HTTP-сервер, с которым связана схема GraphQL. То есть имеется в виду, что через эту схему «пропускаются» все запросы, полученные от клиента, и возвращаемые ответы.

Сервер GraphQL не может знать, что делать с запросом, если ему не «объяснить» это с помощью специальных функций. Благодаря им GraphQL понимает, как получить данные для запрашиваемых полей. Эти функции связаны с соответствующими полями и называются распознавателями, или резолверами (resolvers). После этого клиенту возвращается ответ, который отражает запрашиваемую с клиента структуру данных, обычно в формате JSON.

1. Поясните следующие компоненты

**Subscription -** подписка, полностью аналогичны queries и все что применимо к queries, подходит для subscriptions

**Interface -** способом построения и использования схем GraphQL посредством использования абстрактных типов, про extends;

это тип абстракции, который включает определенный набор полей, которые тип должен включить для внедрения интерфейса.

**Enum -** типы перечисления, особый вид скалярных типов, который может содержать только значение из определенного набора значений.

**fragment** позволяют создавать наборы полей, а затем включать их в запросы, где это необходимо

**union** - указывает на то, что поле может возвращать более одного типа объекта

union SearchResult = Human | Droid | Starship

Когда возвращается тип SearchResult в нашей схеме, мы можем получить Human, Droid или Starship. Отметим, что члены типа union должны быть конкретными типами объекта;