# ****GraphQL Practice Lab Handout****

**Overview**

In this practice lab, you will learn how to set up a GraphQL server using Node.js, Sequelize, and MySQL. You will create a simple employee management system with departments, designations, and employees. This handout will guide you through the process step-by-step.

**Prerequisites**

Before starting, make sure you have the following installed on your machine:

* Node.js
* npm or yarn
* MySQL

**MySQL Database**

CREATE TABLE Departments (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

CREATE TABLE Employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

designation\_id INT,

department\_id INT,

manager\_id INT,

FOREIGN KEY (designation\_id) REFERENCES Designations(id),

FOREIGN KEY (department\_id) REFERENCES Departments(id),

FOREIGN KEY (manager\_id) REFERENCES Employees(id)

);

CREATE TABLE Designations (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

-- Insert into Departments

INSERT INTO Departments (name) VALUES ('HR');

INSERT INTO Departments (name) VALUES ('Engineering');

INSERT INTO Departments (name) VALUES ('Marketing');

-- Insert into Designations

INSERT INTO Designations (name) VALUES ('Manager');

INSERT INTO Designations (name) VALUES ('Developer');

INSERT INTO Designations (name) VALUES ('Designer');

-- Insert into Employees

INSERT INTO Employees (name, email, designation\_id, department\_id, manager\_id) VALUES ('Alice Johnson', 'alice@example.com', 1, 1, NULL);

INSERT INTO Employees (name, email, designation\_id, department\_id, manager\_id) VALUES ('Bob Smith', 'bob@example.com', 2, 2, 1);

INSERT INTO Employees (name, email, designation\_id, department\_id, manager\_id) VALUES ('Charlie Brown', 'charlie@example.com', 3, 3, 1);

INSERT INTO Employees (name, email, designation\_id, department\_id, manager\_id) VALUES ('David Williams', 'david@example.com', 2, 2, 2);

INSERT INTO Employees (name, email, designation\_id, department\_id, manager\_id) VALUES ('Eva Green', 'eva@example.com', 3, 3, 3);

**Project Structure**

Your project will have the following structure:

project-root/

├── config/

│ └── config.json

├── graphql/

│ ├── resolvers.js

│ └── typeDefs.graphql

├── models/

│ ├── department.js

│ ├── designation.js

│ ├── employee.js

│ ├── init-models.js

│ └── index.js

├── node\_modules/

├── package.json

├── package-lock.json

└── server.js

**Step-by-Step Instructions**

**Step 1: Initialize the Project**

1. Create a new directory for your project and navigate into it:

***mkdir graphql-sample-app***

***cd graphql-sample-app***

1. Initialize a new Node.js project:

***npm init -y***

1. Install the necessary dependencies:

***npm install express apollo-server apollo-server-express sequelize mysql2***

**Step 2: Set Up Configuration**

1. Create a config directory and add a config.json file:

{

"development": {

"username": "root",

"password": "root1234",

"database": "employeedb",

"host": "127.0.0.1",

"dialect": "mysql"

},

"test": {

"username": "root",

"password": "root1234",

"database": "employeedb",

"host": "127.0.0.1",

"dialect": "mysql"

},

"production": {

"username": "root",

"password": "root1234",

"database": "employeedb",

"host": "127.0.0.1",

"dialect": "mysql"

}

}

**Step 3: Define the Models**

1. Create a models directory and add the following files:

**department.js**

const Sequelize = require('sequelize');

module.exports = function(sequelize, DataTypes) {

const Department = sequelize.define('Department', {

id: {

autoIncrement: true,

type: DataTypes.INTEGER,

allowNull: false,

primaryKey: true

},

name: {

type: DataTypes.STRING(45),

allowNull: true

}

}, {

sequelize,

tableName: 'departments',

timestamps: false,

indexes: [

{

name: "PRIMARY",

unique: true,

using: "BTREE",

fields: [

{ name: "id" },

]

},

]

});

Department.findByName = async function(name) {

return this.findAll({

where: {

name: {

[Sequelize.Op.like]: `%${name}%`

}

}

});

};

Department.associate = function(models) {

Department.hasMany(models.Employee, { foreignKey: 'department\_id' });

};

return Department;

};

**designation.js**

const Sequelize = require('sequelize');

module.exports = function(sequelize, DataTypes) {

const Designation = sequelize.define('Designation', {

id: {

autoIncrement: true,

type: DataTypes.INTEGER,

allowNull: false,

primaryKey: true

},

name: {

type: DataTypes.STRING(255),

allowNull: false

}

}, {

sequelize,

tableName: 'designations',

timestamps: false,

indexes: [

{

name: "PRIMARY",

unique: true,

using: "BTREE",

fields: [

{ name: "id" },

]

},

]

});

Designation.findByName = async function(name) {

return this.findAll({

where: {

name: {

[Sequelize.Op.like]: `%${name}%`

}

}

});

};

Designation.associate = function(models) {

Designation.hasMany(models.Employee, { foreignKey: 'designation\_id' });

};

return Designation;

};

**employee.js**

const Sequelize = require('sequelize');

module.exports = function(sequelize, DataTypes) {

const Employee = sequelize.define('Employee', {

id: {

autoIncrement: true,

type: DataTypes.INTEGER,

allowNull: false,

primaryKey: true

},

name: {

type: DataTypes.STRING(255),

allowNull: false

},

email: {

type: DataTypes.STRING(255),

allowNull: false

},

designation\_id: {

type: DataTypes.INTEGER,

allowNull: true,

references: {

model: 'designations',

key: 'id'

}

},

department\_id: {

type: DataTypes.INTEGER,

allowNull: true,

references: {

model: 'departments',

key: 'id'

}

},

manager\_id: {

type: DataTypes.INTEGER,

allowNull: true,

references: {

model: 'employees',

key: 'id'

}

}

}, {

sequelize,

tableName: 'employees',

timestamps: false,

indexes: [

{

name: "PRIMARY",

unique: true,

using: "BTREE",

fields: [

{ name: "id" },

]

},

]

});

Employee.findByDepartment = async function(departmentId) {

return this.findAll({

where: {

department\_id: departmentId

}

});

};

Employee.associate = function(models) {

Employee.belongsTo(models.Department, { foreignKey: 'department\_id' });

Employee.belongsTo(models.Designation, { foreignKey: 'designation\_id' });

Employee.belongsTo(models.Employee, { as: 'Manager', foreignKey: 'manager\_id' });

Employee.hasMany(models.Employee, { as: 'Subordinates', foreignKey: 'manager\_id' });

};

return Employee;

};

**init-models.js**

var DataTypes = require("sequelize").DataTypes;

var \_department = require("./department");

var \_designation = require("./designation");

var \_employee = require("./employee");

function initModels(sequelize) {

var department = \_department(sequelize, DataTypes);

var designation = \_designation(sequelize, DataTypes);

var employee = \_employee(sequelize, DataTypes);

employee.belongsTo(department, { foreignKey: "department\_id" });

department.hasMany(employee, { foreignKey: "department\_id" });

employee.belongsTo(designation, { foreignKey: "designation\_id" });

designation.hasMany(employee, { foreignKey: "designation\_id" });

employee.belongsTo(employee, { as: 'Manager', foreignKey: "manager\_id" });

employee.hasMany(employee, { as: 'Subordinates', foreignKey: "manager\_id" });

return {

department,

designation,

employee,

};

}

module.exports = initModels;

module.exports.initModels = initModels;

module.exports.default = initModels;

**index.js**

const Sequelize = require('sequelize');

const config = require('../config/config.json');

const env = process.env.NODE\_ENV || 'development';

const dbConfig = config[env];

const sequelize = new Sequelize(dbConfig.database, dbConfig.username, dbConfig.password, dbConfig);

const initModels = require('./init-models');

const models = initModels(sequelize);

models.sequelize = sequelize;

models.Sequelize = Sequelize;

module.exports = models;

**Step 4: Define GraphQL Schema and Resolvers**

1. Create a graphql directory and add the following files:

**typeDefs.graphql**

type Department {

id: Int!

name: String!

}

type Designation {

id: Int!

name: String!

}

type Employee {

id: Int!

name: String!

email: String!

designation: Designation

department: Department

manager: Employee

subordinates: [Employee]

}

type Query {

departments: [Department]

employees: [Employee]

designations: [Designation]

}

type Mutation {

addEmployee(name: String!, email: String!, designationId: Int!, departmentId: Int!, managerId: Int): Employee

}

**resolvers.js**

const { department, employee, designation } = require('../models');

const resolvers = {

Query: {

departments: () => department.findAll(),

employees: () => employee.findAll({ include: [designation, department, { model: employee, as: 'Manager' }] }),

designations: () => designation.findAll(),

},

Mutation: {

addEmployee: async (\_, { name, email, designationId, departmentId, managerId }) => {

return await employee.create({ name, email, designation\_id: designationId, department\_id: departmentId, manager\_id: managerId });

},

},

Employee: {

designation: (employee) => designation.findByPk(employee.designation\_id),

department: (employee) => department.findByPk(employee.department\_id),

manager: (employee) => employee.findByPk(employee.manager\_id),

subordinates: (employee) => employee.findAll({ where: { manager\_id: employee.id } }),

},

};

module.exports = resolvers;

**Step 5: Set Up the Server**

1. Create a server.js file:

const { ApolloServer } = require('apollo-server');

const fs = require('fs');

const path = require('path');

const resolvers = require('./graphql/resolvers');

const models = require('./models');

const typeDefs = fs.readFileSync(path.join(\_\_dirname, 'graphql', 'typeDefs.graphql'), 'utf8');

const server = new ApolloServer({

typeDefs,

resolvers,

context: { models }

});

models.sequelize.sync().then(() => {

server.listen().then(({ url }) => {

console.log(`🚀 Server ready at ${url}`);

});

});

**Step 6: Initialize the Database**

1. Start your MySQL server and create the employeedb database:

CREATE DATABASE employeedb;

**Step 7: Run the Server**

1. Start your GraphQL server:

node server.js

1. You should see a message indicating that the server is running:

arduino

🚀 Server ready at http://localhost:4000/

1. Open your browser and navigate to http://localhost:4000/ to access the Apollo Server playground.

**Step 8: Testing the API**

You can now test your GraphQL API by running queries and mutations in the Apollo Server playground.

**Example Queries**

query {

departments {

id

name

}

employees {

id

name

email

designation {

name

}

department {

name

}

manager {

name

}

subordinates {

name

}

}

designations {

id

name

}

}

**Example Mutation**

mutation {

addEmployee(name: "John Doe", email: "john@example.com", designationId: 1, departmentId: 1, managerId: null) {

id

name

email

designation {

name

}

department {

name

}

manager {

name

}

subordinates {

name

}

}

}

Congratulations! You have successfully set up a GraphQL server with Node.js, Sequelize, and MySQL. You can now extend this server with additional features and practice building more complex GraphQL queries and mutations.

This document should provide a comprehensive guide for setting up a GraphQL server and help users practice and learn GraphQL with a hands-on approach.