Setting Up Apollo Server with GraphQL, Sequelize, and Oracle Database

# Prerequisites

1. Node.js installed on your machine.  
2. Oracle database up and running with a sample table.  
3. Oracle Instant Client installed if not using Docker for the Oracle database.

# Step-by-Step Guide

## 1. Set Up a New Node.js Project

Open your terminal and run the following commands to set up a new Node.js project:

mkdir graphql-oracle  
cd graphql-oracle  
npm init -y

## 2. Install Required Dependencies

Install the necessary packages:

npm install apollo-server graphql sequelize oracledb sequelize-oracle

## 3. Set Up Sequelize

### Folder Structure

Create the following folder structure:

graphql-oracle/  
├── config/  
├── models/  
├── resolvers/  
├── schema/  
├── index.js  
├── .sequelizerc  
└── package.json

### Configure Sequelize

Create a `.sequelizerc` file in the root directory:

{  
 "config": "./config/config.json",  
 "models-path": "./models",  
 "seeders-path": "./seeders",  
 "migrations-path": "./migrations"  
}

Create a `config/config.json` file with your Oracle database configuration:

{  
 "development": {  
 "username": "your\_username",  
 "password": "your\_password",  
 "database": "your\_database",  
 "host": "your\_host",  
 "dialect": "oracle"  
 }  
}

## 4. Define Your Model

Create a `models/index.js` to initialize Sequelize:

const Sequelize = require('sequelize');  
const config = require('../config/config.json').development;  
  
const sequelize = new Sequelize(config.database, config.username, config.password, {  
 host: config.host,  
 dialect: config.dialect,  
});  
  
const db = {};  
db.Sequelize = Sequelize;  
db.sequelize = sequelize;  
  
db.Sample = require('./sample')(sequelize, Sequelize);  
  
module.exports = db;

Create a `models/sample.js` for your sample table:

module.exports = (sequelize, DataTypes) => {  
 const Sample = sequelize.define('Sample', {  
 id: {  
 type: DataTypes.INTEGER,  
 primaryKey: true,  
 autoIncrement: true  
 },  
 name: {  
 type: DataTypes.STRING,  
 allowNull: false  
 }  
 }, {  
 tableName: 'sample\_table', // Your actual table name  
 timestamps: false  
 });  
  
 return Sample;  
};

## 5. Define GraphQL Schema

Create a `schema/typeDefs.js` for GraphQL type definitions:

const { gql } = require('apollo-server');  
  
const typeDefs = gql`  
 type Sample {  
 id: Int  
 name: String  
 }  
  
 type Query {  
 samples: [Sample]  
 }  
`;  
  
module.exports = typeDefs;

## 6. Define GraphQL Resolvers

Create a `resolvers/index.js` for GraphQL resolvers:

const { Sample } = require('../models');  
  
const resolvers = {  
 Query: {  
 samples: async () => {  
 return await Sample.findAll();  
 }  
 }  
};  
  
module.exports = resolvers;

## 7. Set Up Apollo Server

Create an `index.js` to initialize the Apollo Server:

const { ApolloServer } = require('apollo-server');  
const typeDefs = require('./schema/typeDefs');  
const resolvers = require('./resolvers');  
const db = require('./models');  
  
const server = new ApolloServer({ typeDefs, resolvers });  
  
db.sequelize.authenticate()  
 .then(() => {  
 console.log('Connection has been established successfully.');  
 return db.sequelize.sync();  
 })  
 .then(() => {  
 server.listen().then(({ url }) => {  
 console.log(`🚀 Server ready at ${url}`);  
 });  
 })  
 .catch(err => {  
 console.error('Unable to connect to the database:', err);  
 });

## 8. Create the Sample Table

To create a table in your Oracle database, you need to execute an SQL `CREATE TABLE` statement. Here's a simple example of how to create a `sample\_table`:

CREATE TABLE sample\_table (  
 id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
 name VARCHAR2(255) NOT NULL  
);

3. Verify the table creation:

DESC sample\_table;

## 9. Run Your Server

Start the Apollo Server:

If everything is set up correctly, your Apollo Server should start, and you can query your Oracle database using GraphQL at the server URL displayed in the console.

## 10. Sample Query

In GraphQL Playground or any GraphQL client, you can run the following query to fetch data from your `sample\_table`:

query {  
 samples {  
 id  
 name  
 }  
}

This will return a list of records from the `sample\_table` in your Oracle database.