Sequelize ORM

Relevel
by Unacademy

Topics to be Covered

- ORM
- Sequelize
- Basic Model Querying Techniques
- Associations

What is ORM?

ORM stands for Object Relational Mapping, and is popular among the Object oriented programming developers. It's a technique that performs the mapping of the software objects into the tables of a database. So developers can interact with these objects to perform the CRUD operations rather than writing the sql queries.

Working

ORM creates a model using the OOP providing a high level of abstraction. Mapping describes the relationship of these objects to the database. Once the model is created it can be used to establish the connection between the application and the sql code that is needed to manage the data.

What is ORM?

Advantage:

We now know that ORM minimizes the amount of knowledge needed to write the sql by automatically generating the sql code still there are three major advantage of using it:

1. Application Designing : A well written ORM will force the developer to follow the best practice of application designing.
2. Productivity : Since now the developer need not to focus much on the data access code which used to be time consuming, developers can save their development time and can focus more on the business logic.
3. Reduced Testing Time : Negligible amount of time will be needed to test the data access code as the code generated by the ORM is already tested.

What is ORM?

Types of ORM:

Few of the popular ORM’s used in the market alongside nodejs are:

1. NodeORM: It's an ORM that works with MySQL, SQLlite and PostgresSQL
2. Prisma2 : It's an open source ORM used for Nodejs and Typescript and works with databases such as MySQL, PostgresQL and SQLite.
3. Sequelize: Promise based ORM module used for NodeJS and works with databases such as PostgreSQL, MySQL, SQLite, MSSQL and MariaDB

Sequelize

Without Sequelize

```
> CREATE DATABASE ecom_db;
> USE ecom_db;
> CREATE TABLE `Categories` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL,
  `Description` varchar(255) NOT NULL,
  `CreatedAt` datetime DEFAULT CURRENT_TIMESTAMP,
  `UpdatedAt` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`)
)
```

Sequelize

With Sequelize

```
const Category = sequelize.define("category", {
        id: {
            type: Sequelize.INTEGER,
            primaryKey: true,
            autoIncrement: true
        },
        name: {
            type: Sequelize.STRING,
            allowNull: false
        },
        description: {
            type: Sequelize.STRING
        },
```

Sequelize

With Sequelize

```
},{
       tableName: 'categories'
  /**
  * This helps you to provie a custom name to the table
  * If above is not provided, model name is converted into
plural and set as the table name
  *
  * If we want to just use the model name provided, we can
provide the below option :
  *
  * freezeTableName: true
  */
   });
   return Category;
```

Sequelize

```
const Sequelize = require("sequelize");
const sequelize = new Sequelize(
    ‘ecom_db’,
    ‘root’,
    ‘password’,    {
        host: localhost,
        dialect: mysql,
        operatorsAliases: false,
        pool: {
            max: 5,
            min: 0,
            acquire:30000,
            idle: 10000       }   }   );
```

Sequelize

where:

1. Host: is the ip of the system where the database is located. Currently its on local system so its set to `localhost`
2. dialect: It specifies the connector library that will be used by the sequelize for the database. Since we are working with MySQL we specify 'mysql'
3. operatorAliases: It is a flag that allows a specific symbol operator. Ideally we should use sequelize without any aliases as it improves the security, therefore we have set the flag to false. Few of the default alias that sequelize uses are $eq, $ne, $lte,$gt,$lt, etc.

Sequelize

where:

4. Pool: It contains the parameter of the connection pool between the application and the database that is created while initializing the sequelize. When the connection to the database is made from more than one process then we might have to create more number of instances in each process. But in each instance we have to define a maximum connection for the pool size so that the total pool size is met for each instance. For eg. If we need to create a max connection pool size of 60 and in total we had 6 process then the Sequelize instance of each process should have a maximum of size 10 connection pool.
   Max and min are used to define the threshold for such number of connection pool size.
5. Acquire: max time in ms that a pool will try to get connection before throwing error for that instance of Sequelize.
6. idle: max time in ms that a pool will stay idle before releasing the connection.

Sequelize

Benefits of Sequelize:

Lets summarize the advantage of sequelize:

1. Helps in writing less and consistent code
2. Avoid SQL queries in most of the cases
3. Abstracts database engine

Basic Model Querying Techniques

Now that we have an understanding of Sequelize lets look at the implementation of standard CRUD queries that we earlier implemented using sql, this will assist you in querying the database when diving more into the ecommerce backend application using the sequelize module.

CRUD Operation	**SQL**
Create	Insert
Read	Select
Update	Update
Delete	Delete

Basic Model Querying Techniques

INSERT

- Syntax:

```
Model.create()
```

- Sequelize command:

```
Category.create({name:'Electronics', description:`Product Descriptions`})
```

- Sql command:

```
Insert into Category('name','descriptions') values
('Electronics','Product Descriptions')
```

- Sequelize command:

```
Category.create({name:'Electronics',description:`Product Descriptions`}
,{fields:[name]})
```

- Sql command:

```
Insert into Category('name') values ('Electronics')
```

Basic Model Querying Techniques

SELECT

To select all the records from the table we can use the .findAll() function.

Syntax:

```
Model.findAll()
```

Suppose we have to find all the records of category table then we can run the following command:

```
Category.findAll()
```

To select a specific record from the model we can populate the `attributes` argument of `findAll()` function as:

Syntax:

```
Model.findAll({attributes:[‘col1’,’col2’,[‘col3’, `column3`]]});
```

Here we are selecting the values of col1, col2 and col3 from the table.

[‘col3’, `column3`] will rename the col3 as column3 while returning the value.

A similar sql function for the same would be:

Select col1,col2,col3 as column3 from …

Basic Model Querying Techniques

Aggregation:

To perform any aggregation such as count,max, min, etc. while working with findAll we can use sequelize.fn function.

Suppose we have to count the number of categories in the category table:

```
Category.findAll(attributes:[[sequelize.fn('COUNT',sequelize.col('id')),'
num_category']])
```

Similar sql function for the same would be:

Select count(id) as num_category from Category

Basic Model Querying Techniques

WHERE

Where clause as we know is used to perform the filtration in the search query.

Syntax:

`Model.findAll({ where:{ attribute_name:value } })`	`const {Op} = require('@sequelize/core') Model.findAll({ where:{ Attribute_name:{ [Op.eq]:value} } })`

Which is equivalent to the following sql query:

Select * from . . . where attribute_name = value

sequelize considers the Op.eq to be the default.

Basic Model Querying Techniques

Now suppose we need to pass two condition with the equal operator,then that can be done by simply adding it to the where clause object as follow:

```
Model.findAll({
where:{
     attribute_name_1:value_1,
     attribute_name_2:value_2,
}
})
```

Basic Model Querying Techniques

Examples:

Suppose we need to find out the category from the category table whose id is either 2 or 5 then that can be done by running either of the following commands:

```
category.findAll({
where:{
[Op.or]:{
    id:2,
    id:5
}
})
```

```
category.findAll({
where:{
Id:{
   [Op.or]:[2,5]
}
})
```

Basic Model Querying Techniques

OPERATORS:

As we read in where clause that there can be different operators that can be used along with it, let's go through a list of such operators

[Op.and]:{num1:3,num2:5}	num1=3 and num2=5
[Op.or]:{num1:3,num2:5}	num1=3 or num2=5

Basic Model Querying Techniques

Operators that can be used on single attribute values are:

Basics

[Op.eq]:3	=3
[Op.ne]:3	!=3
[Op.is]:null	null
[Op.not]:true	Is not true
[Op.or]:[1,2]	Is 1 or 2

Basic Model Querying Techniques

Number Comparisons:

[Op.gt]:5	>5
[Op.lt]:5	<5
[Op.gte]:5	>=5
[Op.lte]:5	<=5
[Op.between]:[4,5]	Between 4 and 5
[Op.notBetween]:[4,5]	Not Between 4 and 5

Basic Model Querying Techniques

String Comparisons:

[Op.like]:'Elec%'	Starts with elec
[Op.notLike]:'Elec%'	Not like elec%
[Op.startswith]:'Elec'	Starts with elec
[Op.substring]:Elec	Has a substring elec
[Op.iLike]:'%elec'	Case insensitive ends with elec

Sequelize

suppose we need to find out all the categories whose id is neither 1,2 or 3 and contains a substring `product` in its description then the script will be:

```
Category.findAll({
       where: {
              [Op.and]:
                  {    Id:{
                           [Op.gt]:3       }
                  },    {
                      description:{
                            [Op.substring]:'product'
                        }     }     }   })
```

Equivalent sql query will be:

Select * from category where id>3 AND description LIKE %product%

Sequelize

UPDATE

To update the value of the record in the database we can use the .update function

Syntax:

```
    Model.Update({atrrib_1:val_1},
{
where:{
    attrib_1:null
}
})
```

It means that set the value of attrib_1 to val_1 where the value of attrib_1 is null.

Sequelize

DELETE

To delete a record from the table, or to truncate the table we can use the .destroy function of sequelize along with the where clause as follow:

Syntax:

```
    Model.destroy({
        where:{
                 attrib:val          }    })
```

To delete the record with `attrib` as `val`

To truncate the table one can set the truncate flag of destroy function:

```
Model.destroy({
    truncate:true    });
```

Associations

The standard three associations that is, one to one, one to many and many to many is supported by the sequelize. The only thing is that there are four different functions that are used to form either of these three associations. Such functions are:

1. HasOne
2. Belongs to
3. HasMany
4. BelongstoMany

Suppose we have two models(tables) A & B then the formation of the associations using the above forms are as follows:

- One-to-One association is formed by combining the hasOne and belongsTo

Syntax:

```
A.hasOne(B,{/* options */});
         A.belongsTo(B);
```

Associations

- One-to-Many association is formed by combining the hasMany and belongsTo

Syntax:

```
A.hasMany(B,{/* options */});
A.belongsTo(B);
```

- Many-to-Many association is formed by combining two belongs to many.

Syntax:

```
A.belongsToMany(B,{/* options */});
A.belongsToMany(B);
```

Associations

In all the above syntax the arguments for /* options */ can be:

- foreignKey : attrib_name // Column that is used for performing the relationship
- through : model_name //Determines the junction between two models
- otherKey : attrib // It is the attribute name that is used in the junction table

Note: Here junction is actually a model that helps in establishing the relationship between two strong entities.

Summary

As we are to the end of today's class, let's recall the topics we have covered:

ORM: Technique to map the software objects in to the database tables

Advantage of ORM: Application Designing, Productivity and reduced testing time.

Type of ORM: Node ORM, Prisma2 & Sequelize

Sequelize Benefits:

1. Helps in writing less and consistent code
2. Avoid SQL queries in most of the cases
3. Abstracts database engine

Basic Model Querying Techniques: Create, Read, Update & Delete

Summary

Associations:

1. HasOne
2. Belongs to
3. HasMany
4. BelongstoMany

MCQs

1. **Which of the following is a type of ORM:**
   A. NodeORM
   B. Prisma2
   C. Sequelize
   D. All of the above

MCQs

1. **Which of the following is a type of ORM:**
    A. NodeORM
    B. Prisma2
    C. Sequelize
    D. All of the above

Answer: D

MCQs

2. **FindAll function is equivalent to which of the sql command**
    - A. Insert
    - B. Select
    - C. Update
    - D. Delete

MCQs

2. **FindAll function is equivalent to which of the sql command**
   A. Insert
   B. Select
   C. Update
   D. Delete

Answer: B

MCQs

3. **define function is equivalent to which of the sql command**
    A. Insert
    B. Create
    C. Update
    D. Delete

MCQs

3. **define function is equivalent to which of the sql command**
   A. Insert
   B. Create
   C. Update
   D. Delete

Answer: B

MCQs

4. **Which function is used to insert more than one record in the table**
   A. Bulk
   B. BulkCreate
   C. Create
   D. None of the above

MCQs

4. **Which function is used to insert more than one record in the table**
   - A. Bulk
   - B. BulkCreate
   - C. Create
   - D. None of the above

Answer: B

MCQs

5. **Which of the following is a proper form of association**
   A. HasOne
   B. Belongs to
   C. HasMany
   D. All of the above

MCQs

5. **Which of the following is a proper form of association**
    A. HasOne
    B. Belongs to
    C. HasMany
    D. All of the above

Answer: D

Assignment

1. Create a table name Product with the following attribute :

Product

id	name	description	cost	categoryId

where id is the primary key, and its value is auto incremented on each insert.

Assignment

2. insert the following values:

Name	Descriptions	Cost	CategoryId
iphone15	Apple product	89000	1
Mi12	ReadMe product	15000	1
Prestige Stove	Cooking Stove	14500	2

3. Mention few String Operators of sequelize

Thank You!