**КТ № 13. ORM-библиотека Sequelize. Определение моделей. Запросы к базе данных**

**Время на выполнение практической работы:** 6 часов.

**Теоретические сведения к выполнению практической работы:**

Sequelize - это ORM-библиотека для приложений на Node.js, которая осуществляет сопоставление таблиц в БД и отношений между ними с классами. При использовании Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами. Причем Sequelize может работать с рядом СУБД - MySQL, Postgres, MariaDB, SQLite, MS SQL Server.

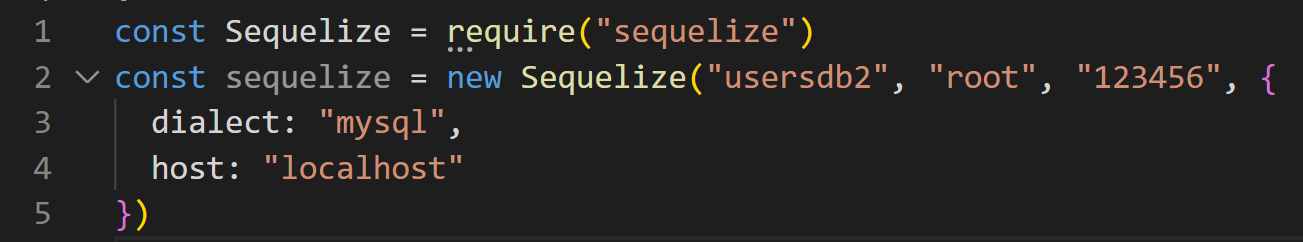
В данном случае мы сосредоточимся на основных моментах работы с данными на примере взаимодействия с БД MySQL.

Вначале установим пакет sequelize.



**Подключение к базе данных**

Для подключения к базе данных прежде всего необходимо создать объект Sequelize:



Для создания объекта sequelize используется функция Sequelize, которая принимает ряд параметров. Первый параметр - имя базы данных. Второй параметр - логин к БД, третий параметр - пароль. Это обязательные параметры.

Кроме того, с помощью четвертого параметра мы можем задать ряд дополнительных опций конфигурации. Четвертый параметр представляет объект, который имеет множество свойств. В данном случае используются только два. Первое свойство - dialect указывает на диалект языка SQL, который используется в запросах - в данном случае "mysql". Второе свойство host представляет адрес, по которому запущен сервер. По умолчанию host имеет значение "localhost", поэтому для подключения к локальной базе данных это свойство в принципе можно не указывать.

Для свойства dialect в зависимости от того, с какой СУБД предстоит работать, могут использоваться следующие значения:

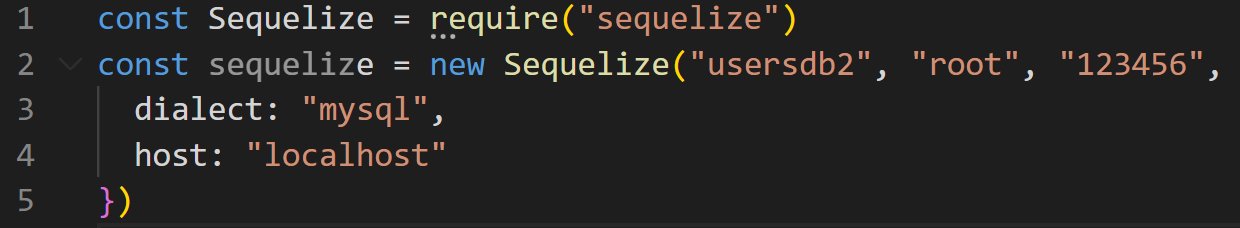
* mysql
* mariadb
* sqlite
* postgres
* mssql

**Подключение к mysql**

Для работы с MySQL также должен быть установлен пакет mysql2:



И при создании объекта Sequelize в параметрах указывается dialect: "mysql":

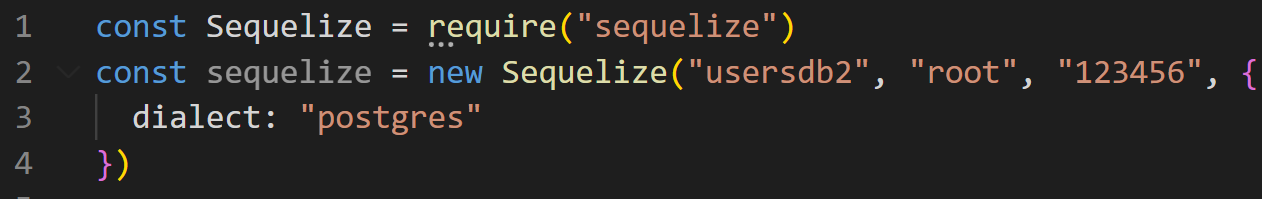


**Подключение к postgres**

Для работы с СУБД Postgres также необходимо установить пакет pg:



И при создании объекта Sequelize в параметрах указывается dialect: "postgres":

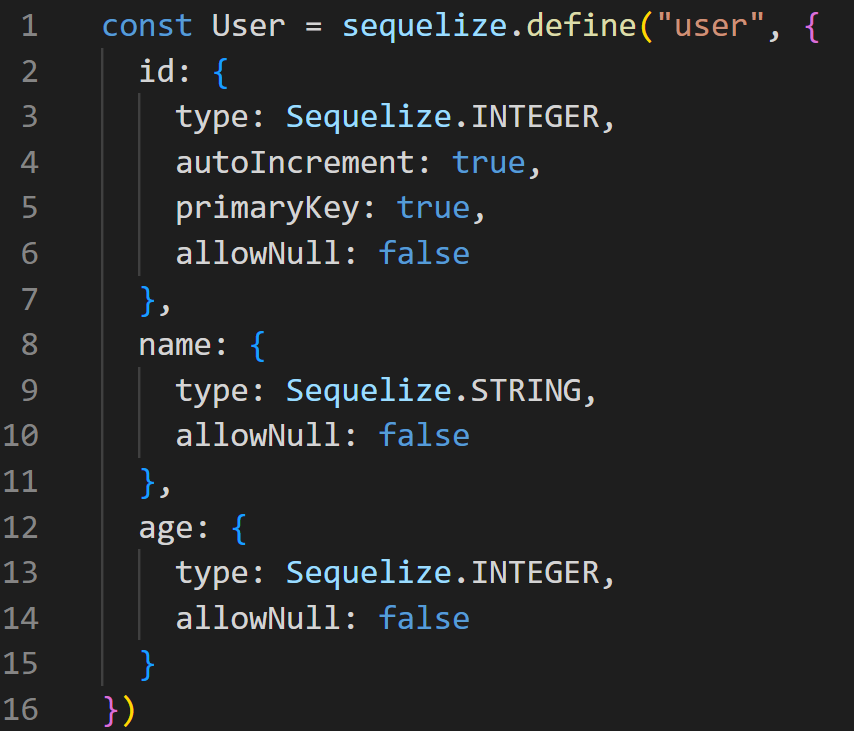


Все остальное взаимодействие с базой данных в большинстве случаев за некоторыми исключениями практически не зависит от конкретной СУБД. А это значит, что большую часть кода (если не весь код) мы можем использовать для подключения к разным СУБД.

**Определение моделей**

Ключевым компонентом в работе Sequelize с БД являются модели. Модели описывают структуру хранящихся в БД данных. Кроме того, через модели в основном идет взаимодействие с БД.

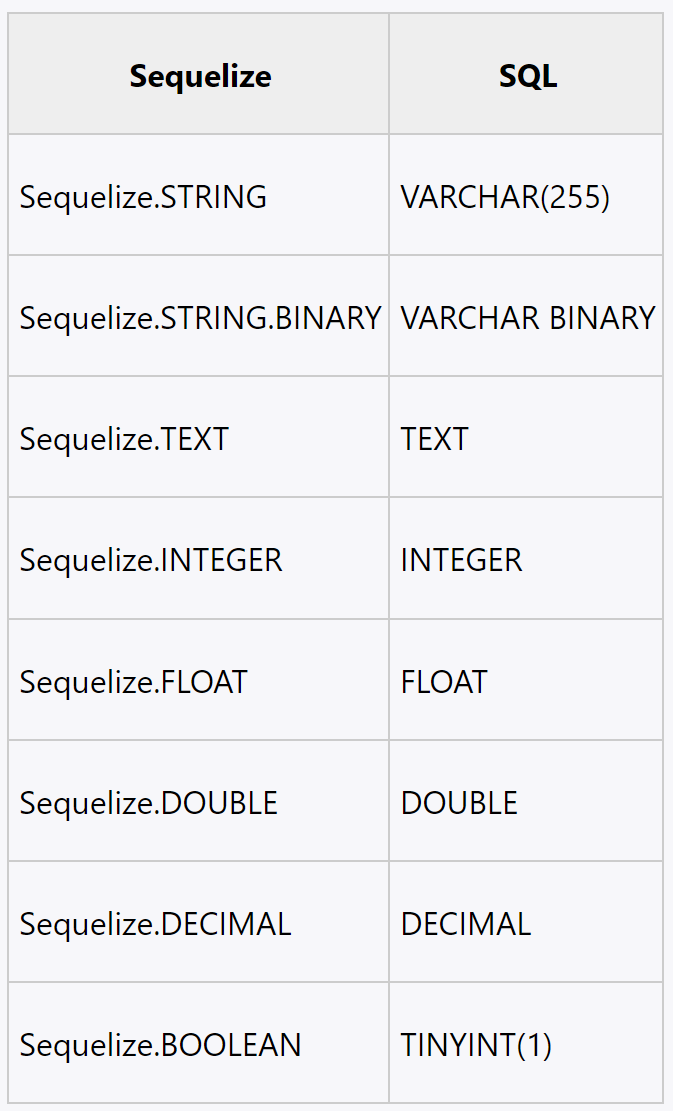
Есть два способа определения моделей. Первый способ представляет использование метода define(). Например, определим модель User, которая представляет пользователя:



Первый параметр метода - это название модели. В данном случае модель называется "user". Важно учитывать, что если в БД для этой модели нет таблицы, то она будет создаваться. Причем в качестве имени таблицы будет применяться название модели во множественном числе в соответствии с правилами английского языка. То есть в данном случае таблица будет называться "users". Это следует учитывать при наименовании моделей.

Второй параметр метода define() задает структуру модели, то есть описывает ее поля. Для каждого поля определяется ряд атрибутов. Атрибут type указывает на тип поля. В данном случае поля id и age представляют тип INTEGER, то есть целое число, а поле name - тип STRING, то есть строку.

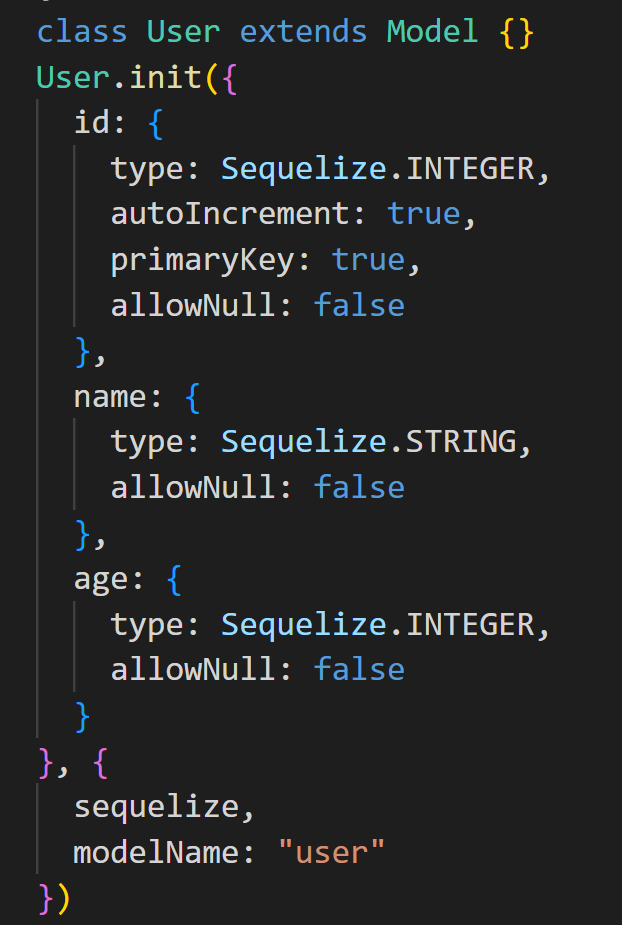
Сопоставление некоторых типов sequelize с типами SQL:



Дополнительно для поля можно задать еще ряд атрибутов. Атрибут allowNull указывает, допускает ли поле отсутствие значение. По умолчанию имеет значение true - то есть у поля может отсутствовать значение. Значение false указывает, что поле обязательно должно иметь какое-либо значение.

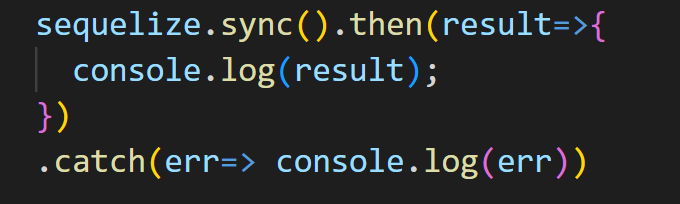
Для поля id в данном случае также задается еще пара атрибутов. Атрибут autoIncrement: true указывает, что значение столбца в таблице в БД будет автоинкрементироваться, а атрибут primaryKey: true говорит о том, что соответствующий столбец в таблице будет выполнять роль первичного ключа.

Второй способ определения модели:



**Синхронизация с базой данных**

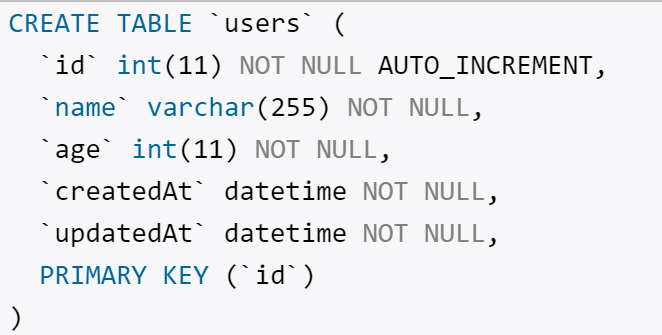
Перед началом взаимодействия с базой данных нам надо убедиться, что таблицы в базе данных соответствуют определению наших моделей. Для синхронизации выполняется метод sync:



Метод sync() синхронизирует структуру базы данных с определением моделей. Например, если для какой-то модели отсутствует соответствующая таблица в БД, то эта таблица создается.

Следует учитывать, что кроме, собственно, полей модели в таблице по умолчанию будут создаваться два дополнительных поля: createdAt и updatedAt, которые будут иметь тип datetime и будут представлять соответственно время создания и последнего обновления строки в таблице.

Например, при работе с базой данных MySQL фактически созданная таблица будет описываться следующим скриптом:



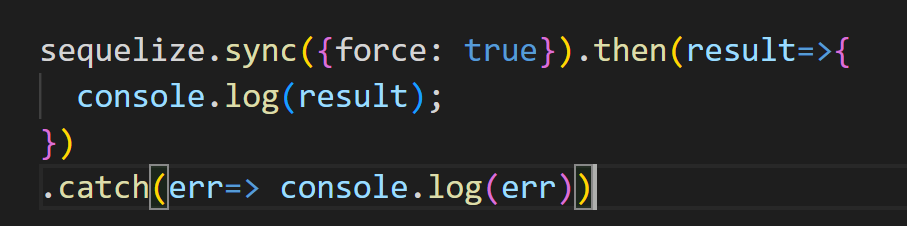
Более того, нам не обязательно определять в модели поле id, так как sequelize по умолчанию также будет создавать подобное поле, которое будет выполнять роль первичного ключа.

Также можно вручную создать таблицу на сервере и потом к ней подключаться без необходимости синхронизации.

Но, возможно, нам не нужны эти дополнительные столбцы createdAt и updatedAt. В этом случае при определении объекта Sequelize нам надо задать параметр define: {timestamps: false}:



Если в базе данных есть подобная таблица, но она не соответствует определению модели, то мы можем использовать параметр {force: true}, чтобы удалить таблицы и создать их заново, но уже с нужной нам структурой:



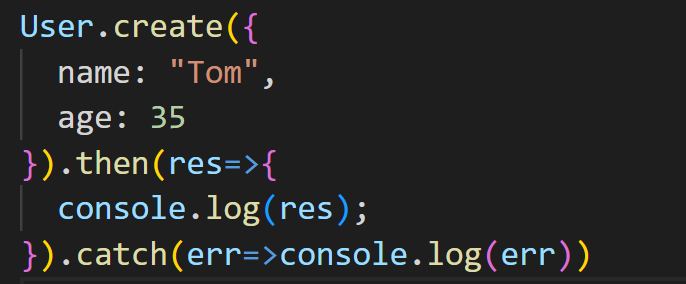
**Запросы к базе данных**

Рассмотрим базовые операции с данными на примере прошлой модели:



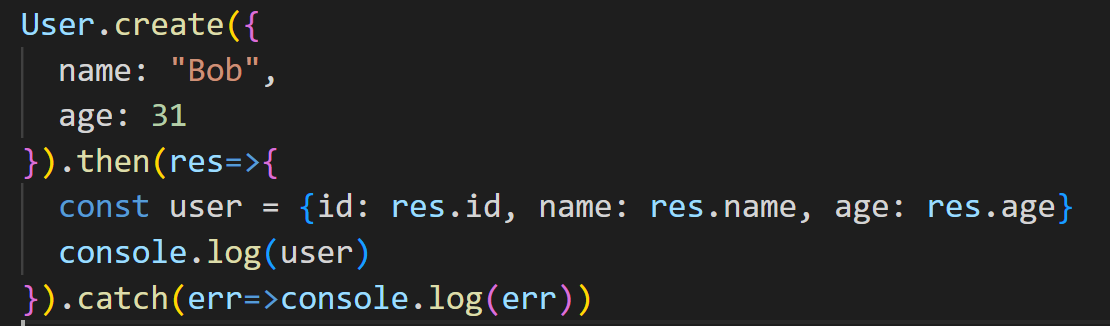
**Добавление данных**

Для добавления данных в базу данных у модели вызывается метод create(), в который передается добавляемый объект:



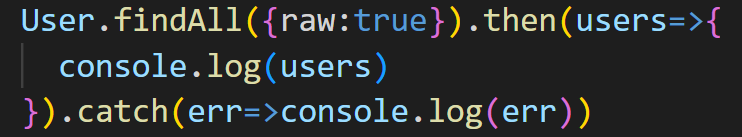
Как мы видим, свойства передаваемого в метод create объекта соответствуют свойствам модели User за исключением свойства id, которое генерируется базой данных.

После операции добавления мы можем получить добавленный объект, в том числе его id, сгенерированный базой данных:

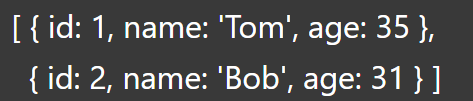


**Получение данных**

Для извлечения всех данных применяется метод findAll():

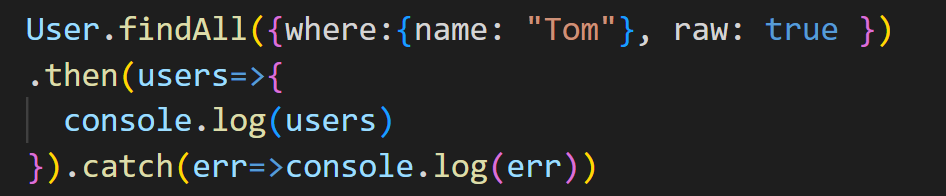


В данном случае в метод передается необязательный объект {raw:true}, который позволяет получить непосредственно данны из БД в вид объектов без дополнительных метаданных. Например, консольный вывод в моем случае выглядел следующим образом:



**Простейшая фильтрация**

Для применения фильтрации при получении данных (а также при обновлении и удалении) применяется оператор where, который указывает на критерий фильтрации. Например, выберем из БД всех пользователей, у которых name="Tom":

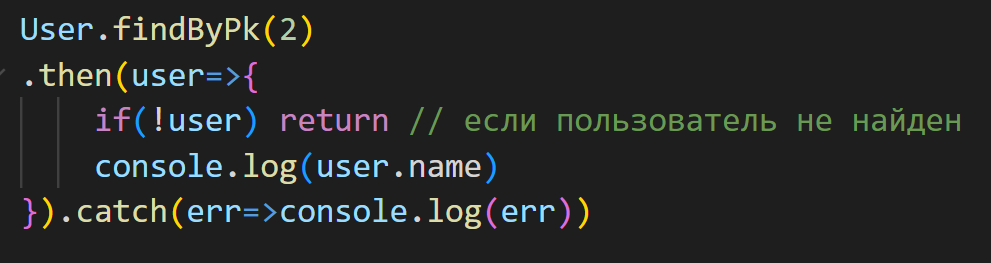


Консольный вывод в данном случае:

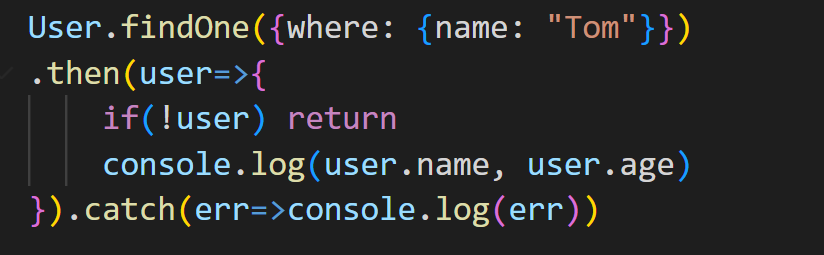


**Получение одного объекта**

Нередко необходимо получить лишь один объект из БД. В этом случае мы можем использовать такие методы как findByPk() (получает объект по первичному ключу) и findOne() (получает один объект по определенному критерию). Например, получим пользователя с id=2:



Или получим одного пользователя с именем Tom:



**Обновление**

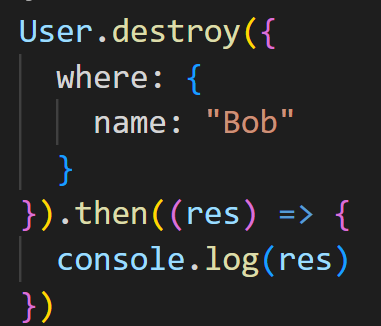
Для обновления применяется метод update(), в который передается объект с новыми значениями и объект-критерий выборки обновляемых объектов:



В данном случае первый объект указывает, что все обновляемые объекты будут получать для поля age значение 36. Второй объект указывает на критерий выборки - обновляются только объекты, у которых name="Bob".

**Удаление**

Для удаления используется метод destroy(), в который передается объект-критерий выборки удаляемых объектов:



В данном случае удаляются объекты, у которых name="Bob".

**Этапы выполнения:**

1. ***Установите пакет sequelize;***
2. ***Создайте объект Sequelize;***
3. ***Установите пакет mysql2;***
4. ***Установите подключение к mysql;***
5. ***Установите пакет pg;***
6. ***Установите подключение к postgres;***
7. ***Определите модель User, которая представляет пользователя, двумя способами;***
8. ***Синхронизируйте структуру базы данных с определением моделей;***
9. ***Добавьте данные в базу данных у модели;***
10. ***Извлеките данные из базы данных;***
11. ***Примените фильтрацию при получении данных;***
12. ***Получите ОДИН объект из базы данных;***
13. ***Обновите данные;***
14. ***Удалите данные.***