1. ORM (Object-Relational Mapping) - это техника программирования, которая позволяет связывать объекты в приложении с таблицами в реляционных базах данных. ORM предоставляет абстракцию базы данных, которая позволяет разработчикам работать с объектами и их отношениями, а не с низкоуровневыми операциями базы данных, такими как SQL-запросы.
2. Для работы Sequelize с Microsoft SQL Server необходимо установить дополнительный пакет tedious, который является драйвером для подключения к SQL Server. Вы можете установить его с помощью npm команды:

npm install tedious

1. В контексте Sequelize, модель представляет собой JavaScript-класс, который отображает таблицу базы данных. Он определяет структуру и поведение данных, которые будут храниться в базе данных. Модель определяет название таблицы, имена столбцов, их типы данных, связи с другими моделями и другие метаданные.
2. Для выполнения SELECT-запроса в Sequelize используется метод Model.findAll(). Например:

Model.findAll({ where: { /\* условие выборки \*/ } });

1. Для выполнения INSERT-запроса в Sequelize используется метод Model.create(). Например:

Model.create({ /\* данные для вставки \*/ });

1. Для выполнения UPDATE-запроса в Sequelize используется метод Model.update(). Например:

Model.update({ /\* обновляемые данные \*/ }, { where: { /\* условие выборки \*/ } });

1. Для выполнения DELETE-запроса в Sequelize используется метод Model.destroy(). Например:

Model.destroy({ where: { /\* условие выборки \*/ } });

1. В Sequelize "хуки" (hooks) позволяют подписываться на различные события жизненного цикла моделей и выполнять код до или после этих событий. Хуки могут быть использованы для валидации данных, преобразования значений, выполнения дополнительных действий и других задач, связанных с моделью. Они предоставляют возможность влиять на поведение моделей в различных точках жизненного цикла, таких как создание, обновление или удаление записей.
2. Функция Model.hasMany в Sequelize используется для определения отношения "один-ко-многим" между двумя моделями. Она указывает, что модель имеет несколько связанных записей в другой модели. Например, если у вас есть модель "Пользователь" и модель "Заказ", и каждый пользователь может иметь несколько заказов, то вы можете определить связь между ними с помощью User.hasMany(Order). Это позволяет вам легко получать все заказы, связанные с определенным пользователем.
3. "Raw query" (сырой запрос) в контексте базы данных означает SQL-запрос, написанный вручную без использования ORM или других абстракций. В Sequelize можно выполнять сырые запросы с помощью функции sequelize.query(). Это позволяет вам написать и выполнить собственные SQL-запросы непосредственно в вашем приложении.
4. Транзакция (transaction) в базах данных представляет собой логическую единицу работы, состоящую из нескольких операций базы данных, которые должны быть выполнены как одна атомарная единица. Транзакция обеспечивает целостность данных и гарантирует, что либо все операции внутри транзакции будут успешно выполнены и зафиксированы в базе данных, либо ни одна из них не будет выполнена и изменения будут отменены (откачены).

В Sequelize, транзакции могут быть созданы с помощью метода sequelize.transaction(). Вы можете выполнить несколько операций базы данных, таких как создание, обновление или удаление записей, в рамках созданной транзакции. После выполнения всех операций, вы можете зафиксировать транзакцию вызовом метода commit(), чтобы сохранить изменения, или вызвать метод rollback(), чтобы откатить все изменения и вернуть базу данных в исходное состояние.