

1. ORM (Object-Relational Mapping) - это технология, которая позволяет связывать объектно-ориентированный код с реляционной базой данных. Она предоставляет программистам возможность работать с данными в виде объектов, а не в виде SQL-запросов.

2. Модель - это абстрактное представление данных в приложении. Она определяет структуру данных, их типы и отношения между ними. Модель может использоваться для создания объектов данных и выполнения операций CRUD (создание, чтение, обновление, удаление).

3. Пул соединений - это механизм, который позволяет приложению устанавливать и использовать несколько соединений с базой данных одновременно. Это увеличивает производительность приложения, поскольку каждый запрос может быть выполнен параллельно.

4. Sequelize и Prisma - это два популярных фреймворка ORM для Node.js. Sequelize поддерживает несколько баз данных, включая MySQL, PostgreSQL и SQLite. Prisma же сфокусирован на работе с базами данных, поддерживающими язык запросов GraphQL.

5. Scope в Sequelize - это механизм, который позволяет определить набор условий для выборки данных. Эти условия могут быть повторно использованы в различных запросах, что упрощает их написание и поддержку.

6. Хуки (hooks) в Sequelize - это функции, которые вызываются автоматически во время выполнения операций CRUD. Они могут быть использованы для выполнения дополнительных действий перед или после операции.

7. Функция Модель.hasMany в Sequelize используется для определения отношений между моделями. Она указывает, что одна модель имеет много связанных записей в другой модели.

8. Raw query в Sequelize - это запрос, написанный на языке SQL, который выполняется непосредственно на базе данных. Он может быть использован в случаях, когда Sequelize не предоставляет достаточно гибких возможностей для выполнения нужных запросов.

9. Prisma отличается от других фреймворков ORM тем, что он предоставляет интеграцию с языком запросов GraphQL. Это позволяет разработчикам использовать привычный синтаксис запросов GraphQL для доступа к данным в базе данных.

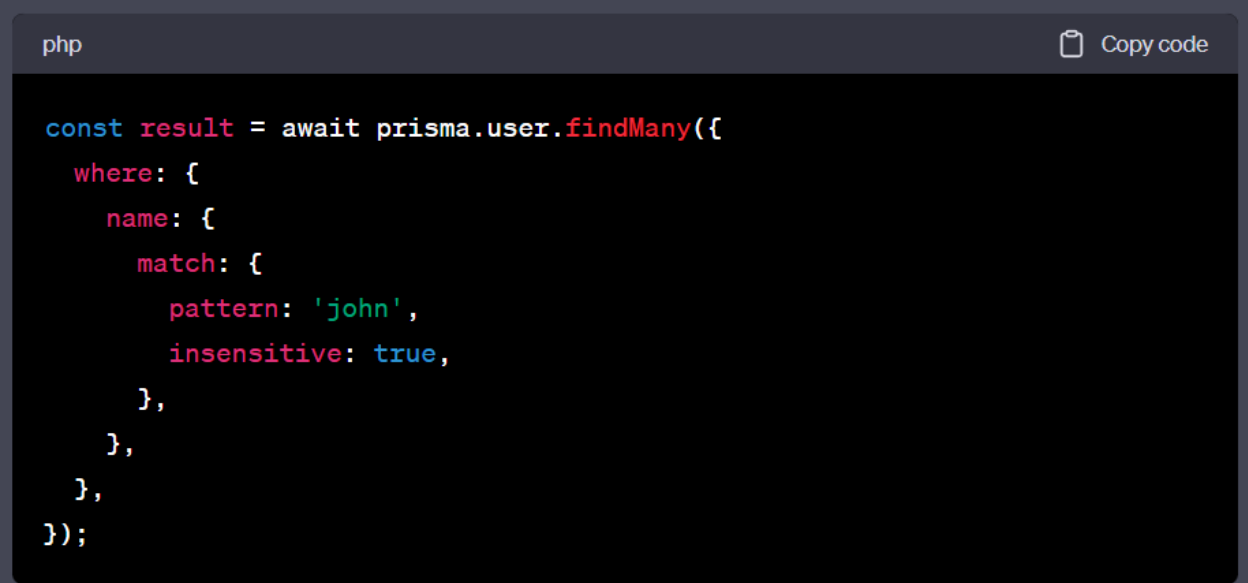
10. Prisma может использоваться с базами данных SQL и NoSQL, но только в том случае, если они поддерживают язык запросов GraphQL. Например, Prisma поддерживает базы данных PostgreSQL, MySQL, SQLite и MongoDB.

11. Prisma может использоваться с языками программирования, поддерживающими работу с Node.js. Это включает JavaScript, TypeScript, Flow и Reason.

12. Методы `every`, `none` и `some` в Prisma используются для выполнения сложных запросов, которые требуют использования операторов AND, OR и NOT. Метод `every` возвращает записи, которые соответствуют всем условиям запроса. Метод `none` возвращает записи, которые не соответствуют ни одному из условий. Метод `some` возвращает записи, которые соответствуют хотя бы одному из условий.

13. Метод `upsert` в Prisma используется для выполнения операции создания или обновления записи в базе данных. Если запись с заданными параметрами уже существует, то она будет обновлена. В противном случае будет создана новая запись.

14. Для включения регистронезависимого поиска в Prisma можно использовать модификатор `i` внутри функции `match`. Например, следующий код выполнит поиск по полю `name`, игнорируя регистр:



```
php                                                                    Copy code

const result = await prisma.user.findMany({
  where: {
    name: {
      match: {
        pattern: 'john',
        insensitive: true,
      },
    },
  },
});
```

15. Методы `$queryRaw` и `$executeRaw` в Prisma позволяют выполнить произвольный SQL-запрос к базе данных. Метод `$queryRaw` возвращает результат запроса в виде массива объектов. Метод `$executeRaw` возвращает количество измененных записей. Оба метода могут быть использованы для выполнения сложных операций, которые не могут быть выполнены с помощью стандартных методов Prisma.