1. **Что такое Prisma?**

Prisma - это современный ORM (Object-Relational Mapping) для баз данных, который позволяет разработчикам быстро и легко работать с базами данных, используя язык запросов на стороне сервера. Prisma может использоваться с различными базами данных, такими как PostgreSQL, MySQL и SQLite.

1. **Назовите отличия Prisma от других фреймворков ORM?**

Одним из основных преимуществ Prisma является автоматическая генерация типизированных API для взаимодействия с базой данных, которые позволяют сократить количество написанного кода и уменьшить вероятность ошибок в работе с базой данных.

Prisma также предлагает мощные возможности миграции, которые облегчают изменение схемы базы данных и управление ее версиями. Благодаря этому, разработчики могут легко создавать и обновлять базы данных без необходимости написания многочисленных SQL-запросов и миграций вручную.

1. **Какова роль Prisma Client в доступе к данным и управлении ими?**

Prisma Client - это клиентская библиотека, которая создается автоматически при использовании Prisma для определенной базы данных. Она предоставляет типизированный API для доступа к данным в базе данных и управления ими. С помощью Prisma Client разработчики могут выполнять операции чтения, записи и обновления данных в базе данных.

1. **Можно ли использовать Prisma с базами данных SQL и NoSQL?**

Prisma - это ORM (Object-Relational Mapping), который в настоящее время поддерживает несколько баз данных SQL, таких как PostgreSQL, MySQL, SQLite, Microsoft SQL Server и MariaDB.

Кроме того, существует экспериментальная поддержка для базы данных MongoDB, которая является документ-ориентированной NoSQL базой данных.

1. **Какие языки программирования совместимы с Prisma?**

Prisma - это инструмент для работы с базами данных, написанный на языке TypeScript. Он предоставляет типизированный API для взаимодействия с базами данных на стороне сервера. Однако, Prisma Client, который предоставляет этот API, может использоваться с любым языком программирования, который поддерживает вызовы API.

В частности, Prisma Client предоставляет официальные библиотеки для использования с языками программирования, такими как JavaScript (Node.js), TypeScript, Go, Java, Python и Ruby. Эти библиотеки автоматически генерируются на основе схемы базы данных, определенной в Prisma Schema.

Это означает, что разработчики могут использовать Prisma с любым языком программирования, который поддерживает вызовы API, чтобы легко и эффективно работать с базами данных и управлять данными в своих приложениях.

1. **Расскажите, где и как можно использовать *every*, *none* и *some*?**

Метод every может использоваться для проверки, удовлетворяют ли все элементы в коллекции (например, записи в базе данных) определенному условию.

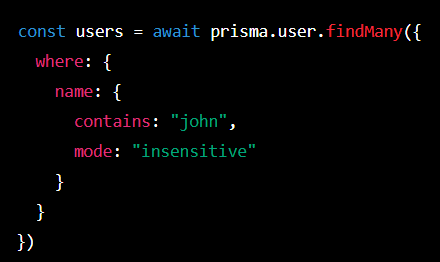
Метод none используется для проверки отсутствия элементов в коллекции, удовлетворяющих определенному условию.

Метод some используется для проверки, удовлетворяет ли хотя бы один элемент в коллекции определенному условию.

1. **Для чего нужен метод *upsert*?**

Метод upsert (от англ. "update or insert" - "обновить или вставить") в Prisma используется для обновления или создания записи в базе данных, если такая запись уже существует, либо создания новой записи, если ее еще нет.

1. **Как включить регистронезависимый поиск?**



Здесь параметр mode устанавливает режим поиска как "insensitive", что означает регистронезависимый поиск.

1. **Объясните назначение методов *$queryRaw*, *$executeRaw*.**

$queryRaw предназначен для выполнения SQL-запросов, которые возвращают данные. Метод $queryRaw возвращает результат SQL-запроса в виде массива объектов. Каждый объект в массиве соответствует одной строке возвращаемого результата.

$executeRaw используется для выполнения SQL-запросов, которые не возвращают данные, например, для вставки, обновления или удаления данных в базе данных. Метод $executeRaw возвращает количество измененных строк в результате выполнения запроса.

1. **Какие виды транзакций существуют в Prisma?**

В Prisma существует два вида транзакций:

Транзакции на уровне базы данных (database-level transactions) - используются для обеспечения целостности данных и предотвращения ошибок при выполнении нескольких запросов к базе данных. Такие транзакции выполняются в рамках одной сессии базы данных и могут включать несколько запросов, которые выполняются в одной транзакции. В Prisma транзакции на уровне базы данных могут быть выполнены с помощью метода transaction().

Транзакции на уровне приложения (application-level transactions) - используются для группировки нескольких операций в одну транзакцию на уровне приложения. Такие транзакции выполняются с помощью метода transactional(), который обеспечивает выполнение нескольких запросов в рамках одной транзакции на уровне приложения.