**1. ЧТО ТАКОЕ ORM?**

ORM (Object-Relational Mapping) — это технология программирования, которая позволяет связывать объектно-ориентированный код с реляционными базами данных.

**2.Зачем используются ORM-решения при работе с базами данных? Опишите их достоинства и недостатки.**

ORM-решения используются для упрощения работы с базами данных. Они позволяют программистам работать с базами данных на более высоком уровне абстракции, используя объектно-ориентированный подход. ORM-решения автоматически генерируют SQL-запросы на основе объектов, что упрощает написание кода и снижает вероятность ошибок. Они также обеспечивают более безопасную работу с базами данных.

**Достоинства ORM-решений:**

Упрощение работы с базами данных. ORM-решения позволяют программистам работать с базами данных, используя объектно-ориентированный подход, что упрощает процесс работы с данными.

Увеличение производительности. ORM-решения могут оптимизировать SQL-запросы, что позволяет увеличить производительность приложения.

Уменьшение количества ошибок. ORM-решения предоставляют удобный интерфейс для работы с базами данных, что уменьшает количество ошибок, связанных с написанием SQL-запросов.

Переносимость. ORM-решения позволяют переносить приложения между различными СУБД, что упрощает разработку и поддержку приложений.

**Недостатки ORM-решений:**

Сложность настройки. ORM-решения могут быть сложны в настройке и требуют дополнительных знаний для работы с ними.

Ограничения. ORM-решения могут иметь ограничения в функциональности, которые могут ограничить возможности приложения.

Производительность. ORM-решения могут иметь некоторые накладные расходы, которые могут снизить производительность приложения.

Сложность отладки. ORM-решения могут усложнить процесс отладки приложения, так как могут быть скрыты некоторые детали работы с базой данных.

**3.Что такое Entity Framework?**

Entity Framework - это технология объектно-ориентированного доступа к данным (ORM), которая позволяет разработчикам работать с данными в виде объектов и свойств, а не в виде SQL-запросов и таблиц базы данных.

**4.Что такое EDM в Entity Framework и из каких уровней она состоит?**

EDM (Entity Data Model) в Entity Framework - это концептуальная модель данных, которая описывает структуру и отношения между сущностями в базе данных**. Она состоит из трех уровней:**

Концептуальный уровень (CSDL) - описывает сущности и их отношения в виде объектов, которые могут быть использованы в приложении.

Хранилищный уровень (SSDL) - описывает структуру базы данных и ее таблицы, столбцы, ключи и отношения.

Уровень сопоставления (MSL) - связывает концептуальный и хранилищный уровни, определяя, как сущности и их свойства отображаются на таблицы и столбцы в базе данных.

**5. Перечислите способы создания моделей. Опишите их действия.**

Ручное создание моделей — это процесс создания моделей вручную с помощью языка программирования C#.

Использование генераторов кода — это процесс создания моделей с помощью специальных инструментов, которые генерируют код на основе заданных параметров.

Использование сторонних библиотек — это процесс создания моделей с помощью сторонних библиотек, которые предоставляют готовые решения для определенных задач.

Использование инструментов визуального моделирования — это процесс создания моделей с помощью специальных инструментов, которые позволяют создавать модели визуально.

**6. Назовите основные отличия EF от других технологий работы с базами данных.**

**Основные отличия EF от других технологий работы с базами данных:**

EF позволяет работать с базами данных, используя объектно-ориентированный подход.

EF автоматически генерирует SQL-запросы на основе LINQ-выражений, что упрощает написание кода и уменьшает количество ошибок.

EF позволяет использовать различные провайдеры баз данных, включая SQL Server, MySQL, Oracle и другие.

EF поддерживает работу с хранимыми процедурами и функциями баз данных.

EF позволяет использовать различные подходы к созданию модели данных, включая Code First, Database First и Model First.

EF поддерживает работу с транзакциями баз данных, что обеспечивает целостность данных и безопасность приложений.

**7. Для чего и как используется DbContext?**

DbContext - это класс, который предоставляет доступ к базе данных и позволяет выполнять операции CRUD (Create, Read, Update, Delete) с объектами в базе данных.

DbContext используется для создания и управления объектами контекста базы данных.

Для использования DbContext необходимо создать класс, который наследуется от класса DbContext и определить свойства DbSet для каждой таблицы в базе данных. Затем можно использовать объекты DbSet для выполнения операций CRUD с объектами в базе данных.

**8. Как и для чего используются аннотации при настройке конфигураций Code First?**

Аннотации используются для указания дополнительной информации о модели данных. Они могут быть использованы для указания ограничений на свойства, настройки индексов, настройки отношений между таблицами.

**9. Как и для чего используются Fluent API при настройке конфигураций Code First?**

С помощью Fluent API можно настраивать свойства сущностей, отношения между сущностями, индексы, ограничения и другие аспекты модели данных.

**10. Какие существуют способы загрузки связанных данных?**

Существует несколько способов загрузки связанных данных:

Жадная загрузка (Eager loading) - это загрузка всех связанных данных вместе с основными данными в одном запросе к базе данных. Это позволяет избежать дополнительных запросов к базе данных при обращении к связанным данным.

Ленивая загрузка (Lazy loading) - это загрузка связанных данных только при обращении к ним. Это позволяет уменьшить количество запросов к базе данных, но может привести к задержкам при обращении к связанным данным.

Загрузка по требованию (Load on demand) - это загрузка связанных данных только при определенных условиях, например, при прокрутке страницы или при нажатии на кнопку. Это позволяет уменьшить количество запросов к базе данных и ускорить загрузку страницы.

Загрузка в фоновом режиме (Background loading) - это загрузка связанных данных в фоновом режиме, чтобы не прерывать работу пользователя. Это может быть полезно при загрузке больших объемов данных или при работе с медиа-файлами.

**11. В чем разница между концепциями Code First и Database First?**

Code First - это подход, при котором разработчик создает модели данных в коде и затем использует инструменты Entity Framework для создания базы данных на основе этих моделей. Это означает, что база данных создается автоматически на основе кода, который разработчик написал.

Database First - это подход, при котором разработчик создает базу данных вручную, используя инструменты управления базами данных, такие как SQL Server Management Studio. Затем разработчик использует инструменты Entity Framework для создания моделей данных на основе этой базы данных.

Основная разница между этими двумя подходами заключается в том, что Code First начинается с кода, а Database First начинается с базы данных. Code First позволяет разработчикам быстро создавать базы данных, не тратя время на создание схемы базы данных вручную. Database First позволяет разработчикам создавать базы данных с большей точностью и контролем, но может занять больше времени на создание схемы базы данных вручную.

**12. Зачем EF использует миграции? Как создать новую миграцию?**

EF использует миграции для управления изменениями в базе данных. Миграции позволяют создавать и применять изменения в базе данных, такие как добавление новых таблиц, изменение существующих таблиц или добавление новых столбцов.

Чтобы создать новую миграцию, необходимо выполнить следующие шаги:

Открыть консоль диспетчера пакетов (Package Manager Console) в Visual Studio.

Выбрать проект, в котором необходимо создать миграцию.

Ввести команду Add-Migration <название миграции>, где <название миграции> - это уникальное название для новой миграции.

Нажать Enter, чтобы выполнить команду.

После выполнения команды EF создаст новую миграцию, которая будет содержать изменения в базе данных. Эту миграцию можно применить к базе данных, используя команду Update-Database.

**13. Расскажите алгоритм выполнения CRUD-запросов при работе с БД через библиотеку Entity Framework.**

CRUD-запросы (Create, Read, Update, Delete) в Entity Framework выполняются с помощью объекта контекста DbContext. Алгоритм выполнения CRUD-запросов в Entity Framework выглядит следующим образом:

Создание объекта контекста DbContext.

Создание объекта модели данных (Entity) и заполнение его свойств.

Выполнение запроса на создание записи в БД с помощью метода Add() объекта DbSet.

Выполнение запроса на чтение данных из БД с помощью метода Find() объекта DbSet.

Выполнение запроса на обновление данных в БД с помощью метода Update() объекта DbSet.

Выполнение запроса на удаление данных из БД с помощью метода Remove() объекта DbSet.

Сохранение изменений в БД с помощью метода SaveChanges() объекта контекста DbContext.