1. Что такое ORM?

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

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

ORM (Object-Relational Mapping) - это технология, которая позволяет связать объектно-ориентированный код с базой данных. ORM-решения используются для упрощения работы с базами данных, так как они предоставляют удобный интерфейс для доступа к данным и автоматически генерируют SQL-запросы.

**Достоинства ORM-решений:**

1. Упрощение работы с базами данных. ORM-решения позволяют работать с базами данных на более высоком уровне абстракции, что делает код более читаемым и понятным.

2. Повышение безопасности. ORM-решения обеспечивают защиту от SQL-инъекций и других атак на базу данных.

3. Ускорение разработки. ORM-решения позволяют быстро создавать и изменять таблицы в базе данных.

4. Поддержка различных СУБД. ORM-решения поддерживают работу с различными СУБД, что позволяет использовать один и тот же код для работы с разными базами данных.

**Недостатки ORM-решений:**

1. Сложность настройки. Настройка ORM-решений может быть сложной и требует определенных знаний в области баз данных.

2. Ограничения в производительности. ORM-решения могут работать медленнее, чем непосредственная работа с базой данных через SQL-запросы.

3. Ограничения в функциональности. ORM-решения могут не поддерживать все возможности, которые предоставляются СУБД, что может привести к ограничениям в функциональности приложения.

4. Необходимость в обучении. Использование ORM-решений требует знаний в области объектно-ориентированного программирования и баз данных, что может быть сложно для новичков.

1. Что такое Entity Framework?

**Entity Framework** представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

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

Модель EDM - это набор основных понятий, которые описывают структуру данных независимо от формы хранения.

1. Уровень хранилища Удаленный описывается Store Schema Definition Language (SSDL)
2. Уровень сопоставления Связующий Mapping Schema Language (MSL)
3. Концептуальный уровень Рабочий описывается Conceptual Schema Definition Language (CSDL)

5. Перечислите способы создания моделей. Опишите их действия.

1. **Database-First** создание базы данных -> генерация EDMX-модели

2. **Model-First** создание графической модели EDMX -> генерация базы данных

3. **Code-First** настройка классов C# объектной модели 1) генерация сущностных классов из существующей базы данных 2) создание базы данных из созданной вручную модели объектов C

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

1. Поддержка LINQ. EF поддерживает язык запросов LINQ, что позволяет писать запросы к базе данных на языке C# или VB.NET, что упрощает работу с данными.

2. Кроссплатформенность. EF поддерживает работу с различными СУБД, включая Microsoft SQL Server, Oracle, MySQL, PostgreSQL и др., что делает его кроссплатформенным решением.

3. Code First подход. EF позволяет создавать базу данных из объектной модели приложения, что упрощает процесс разработки и обновления базы данных.

4. Поддержка множественных отношений. EF поддерживает работу с множественными отношениями между таблицами в базе данных, что делает его более гибким решением.

5. Поддержка хранимых процедур и функций. EF позволяет вызывать хранимые процедуры и функции из C#-кода, что может повысить производительность приложения.

6. Поддержка миграций. EF позволяет автоматически создавать миграции для обновления базы данных при изменении объектной модели приложения.

7. Для чего и как используется DbContext?

Экземпляр **DbContext** представляет сеанс с базой данных и может **использоваться** для запроса и сохранения экземпляров сущностей. **DbContext** — это сочетание шаблонов единиц работы и репозитория.

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

Для настройки сопоставления моделей и таблиц с помощью атрибутов

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

набор методов, которые определяются сопоставление между классами и их свойствами и таблицами и их столбцами

Если мы используем подход Code First, то классы моделей сопоставляются с таблицами с помощью ряда правил в Entity Framework. Но иногда необходимо изменить и переопределить логику этих правил. Для этого используется Fluent API и аннотации данных. Fluent API по большому счету представляет набор методов, которые определяются сопоставление между классами и их свойствами и таблицами и их столбцами. Как правило, функционал Fluent API задействуется при переопределении метода **OnModelCreating**:

10.Какие существуют способы загрузки связанных данных?

Способы получения связанных данных:

►"жадная загрузка" или eager loading

**Eager** **load** (**жадная** **загрузка**) — это процесс, при котором запрос для одного типа объекта также загружает связанные объекты в рамках одного запроса к базе данных.

►"ленивая загрузка" или lazy loading

**Lazy** **loading** («**ленивая**» **загрузка**) — это метод, который откладывает загрузку некоторых элементов страницы, показывая их лишь тогда, когда пользователь доскроллит страницу до их расположения.

►explicit loading("явная загрузка")

**Явная** **загрузка**, как и отложенная **загрузка**, не приводит к **загрузке** всех связанных данных в первом запросе. Но при этом, в отличие от отложенной **загрузки**, при вызове навигационного свойства связанного класса, эта **загрузка** не приводит к автоматическому извлечению связанных данных, вы должны явно вызвать метод **Load**(), если хотите загрузить связанные данные.

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

**Database-First** создание базы данных -> генерация EDMX-модель

**Code-First** настройка классов C# объектной модели 1) генерация сущностных классов из существующей базы данных 2) создание базы данных из созданной вручную модели объектов C

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

**Миграция** — это средство поддержания структуры БД в актуальном состоянии.

После того, как вы добавили или удалили поле, добавили класс (и прописали его в DbContext), изменили параметры полей в entity configuration, вы должны создать новую миграцию с помощью команды Add-Migration, которую надо запустить из **Project Management Console**.

Entity Framework автоматически сгенерирует для вас код миграции, то есть вызовы, создающие поля или таблицы. Иногда вам нужно подкорректировать этот код, например, если речь идёт о переименовании поля.

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