**Вопросы к лабораторной работе №9**

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.

**Использование ORM экономит много времени, потому что:**

Суть этого достаточно сложного определения заключается в том, что создается некая абстракция - “**виртуальная объектная база**”, запросы к которой, преобразуются в **SQL команды,** т. е. больше не нужно писать **SQL-запросы к базе данных вручную** (хотя в некоторых, достаточно нетривиальных случаях, писать запрос вручную все-таки придется, благо таких случаев не так много).

**Использование библиотеки ORM более гибко, потому что:**

* Это соответствует вашему естественному способу кодирования.
* Он абстрагирует систему БД, поэтому вы можете изменить ее, когда захотите.
* Модель слабо связана с остальной частью приложения, поэтому вы можете изменить ее или использовать где-либо еще.
* Это позволяет вам использовать ООП, как наследование данных без головной боли.

**Недостатки:**

* Вы должны изучить это, и библиотеки ORM не являются легкими инструментами;
* Вы должны настроить это. Та же проблема.
* Производительность нормальная для обычных запросов, но мастер SQL всегда будет лучше с собственным SQL для больших проектов.
* Абстрагирует БД. Хотя это нормально, если вы знаете, что происходит за кулисами, это ловушка для новых программистов, которые могут писать очень жадные операторы, как тяжелый удар в цикле for.

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.

Существует множество технологий для работы с базами данных, и каждая из них имеет свои особенности. Ниже приведены некоторые отличия Entity Framework (EF) от других технологий работы с базами данных:

1. EF представляет базу данных в виде объектов и позволяет работать с ней с помощью языка программирования C#. Это позволяет разработчикам использовать объектно-ориентированный подход к работе с базами данных и упрощает написание кода.
2. EF поддерживает множество различных провайдеров баз данных, включая Microsoft SQL Server, Oracle, MySQL и другие. Это позволяет использовать EF для работы с различными базами данных без необходимости изучения новых технологий.
3. EF предоставляет механизмы автоматического создания таблиц базы данных на основе модели данных, что упрощает и ускоряет процесс разработки приложений.
4. EF позволяет использовать LINQ для написания запросов к базе данных, что упрощает и ускоряет написание кода и сокращает количество ошибок.
5. EF предоставляет механизмы отслеживания изменений в объектах модели данных и автоматического сохранения этих изменений в базу данных. Это упрощает написание кода и сокращает количество ошибок.
6. EF поддерживает механизмы оптимистической блокировки, которые позволяют избежать конфликтов при изменении одних и тех же данных несколькими пользователями одновременно.
7. EF поддерживает механизмы миграции базы данных, которые позволяют автоматически обновлять структуру базы данных при изменении модели данных. Это упрощает процесс разработки и обновления приложений.

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

Алгоритм выполнения CRUD-запросов в EF выглядит следующим образом:

1. Создание объекта контекста базы данных.

using (var context = new MyDbContext()) { // ... }

* Добавление объекта в коллекцию DbSet объекта контекста.

context.MyObjects.Add(myObject);

* Получение объекта из базы данных с помощью метода Find.

var myObject = context.MyObjects.Find(id);

* Обновление свойств объекта.

myObject.Name = "New Name";

* Удаление объекта из коллекции DbSet объекта контекста.

context.MyObjects.Remove(myObject);

1. Сохранение изменений в базе данных.

context.SaveChanges();