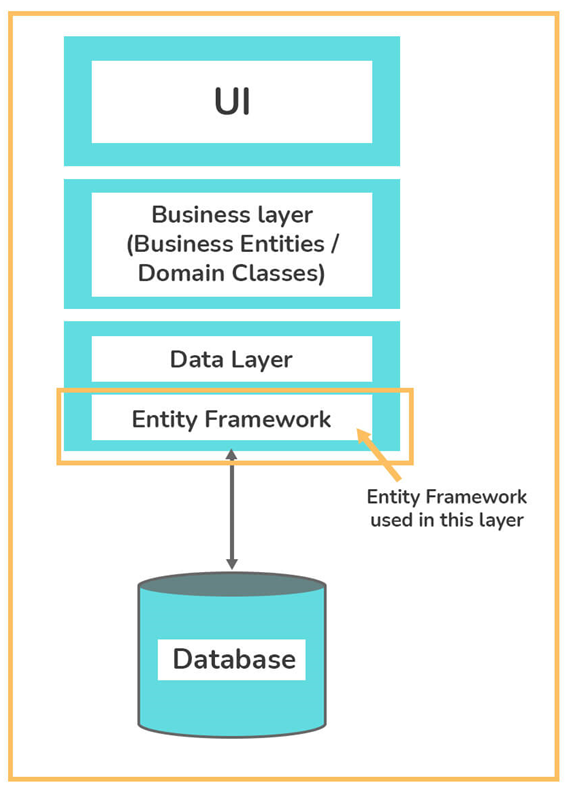
**What is Entity Framework?**

In [.NET](https://www.interviewbit.com/dot-net-interview-questions/) applications (desktop and web), ADO.NET is an old and powerful method for accessing databases and requires a lot of coding. Simply put, all of the work is done manually. Microsoft has therefore introduced Entity Framework to automate the database technique.   
  
Entity Framework (EF), a Microsoft-supported open-source ORM (Object-Relational Mapper) for .NET applications, is a set of utilities and mechanisms that works towards optimizing data-driven applications. This is considered one of the most important concepts in the [**MVC model**](https://www.interviewbit.com/mvc-interview-questions/). With the help of tables and columns, developers are able to streamline mapping between various objects in software applications. It makes use of objects of domain-specific classes but ignores the database tables and columns that are used to store the data. Designers can utilize EF in order to maintain and develop data-oriented applications with little coding and a high level of absorption when dealing with data, unlike traditional applications.



The EF fits between the business entities (domain classes) and the database, as shown in the above figure. Additionally, it retrieves data from the database and converts it to business entity objects automatically, and also saves data stored in business entity properties.

**Entity Framework Interview Questions for Freshers**

**1. Explain the advantages of the Entity Framework.**

Entity Framework has the following advantages:

* With its excellent prototypes, it is possible to write object-oriented programs.
* By allowing auto-migration, it is simple to create a database or modify it.
* It simplifies the developer's job by reducing the code length with the help of alternate commands.
* It reduces development time, development cost, and provides auto-generated code.
* A unique syntax (LINQ / Yoda) is provided for all object queries, whether they are databases or not.
* It enables the mapping of multiple conceptual models to a single storage schema.
* Business objects can be mapped easily (with drag & drop tables).

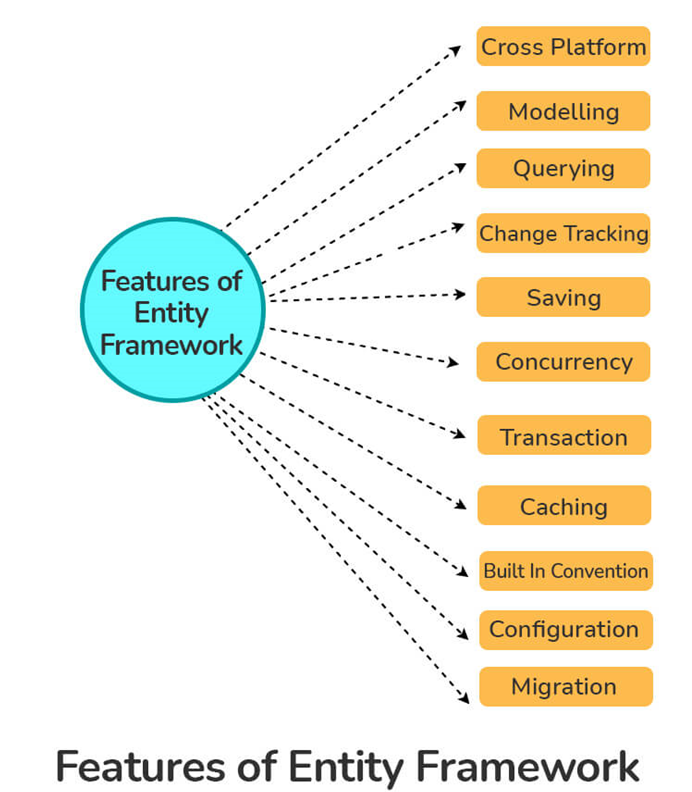
**Describe some of the disadvantages of the Entity Framework.**

Entity Framework has the following disadvantages:

* If the developer does not use raw SQL codes, things can become complicated sometimes.
* It is a slower form of the Object Relational Mapper.
* For a big domain model, it's not ideal.
* Some RDMS do not offer this feature.
* EF's main drawback is its lazy loading
* This requires a non-traditional approach to handling data that isn't available for every database.
* Since the data migration functionality is weak, it isn't fully effective in practice.

**What are the features of the Entity Framework?**

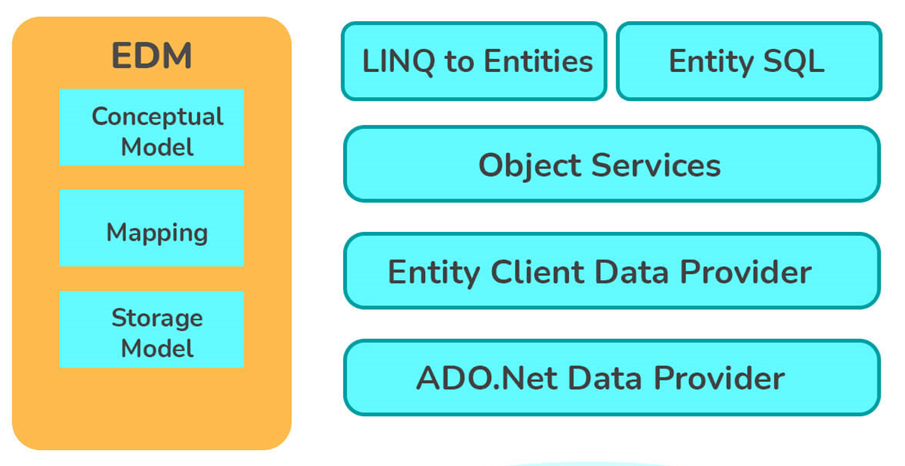
Below are some of Entity Framework's basic features:



* **Cross-Platform:** It is lightweight, extensible, open-source, and can be used on Windows, Linux, and Mac.
* **Querying:**It allows us to retrieve data from underlying databases using LINQ queries, which are then transformed into database-specific query languages.
* **Modeling:**EDMs (Entity Data Models) are typically created based on POCOs (Plain Old CLR Objects), which are entities with get/set properties of different types. This model is used when querying and saving entity data to the underlying database.
* **Change Tracking:** By using the SaveChanges method of the context, EF tracks changes to entities and their relationships and ensures the correct updates are performed on the database. Change tracking is enabled by default in EF but can be disabled by setting the AutoDetectChangesEnabled property of DbContext to false.
* **Saving:** Upon calling the "SaveChanges()" method, EF executes the INSERT, UPDATE, and DELETE commands to the database based on the changes made to entities. "SaveChangesAsync()" is another asynchronous method provided by EF.
* **Concurrency:** EF provides built-in support for Optimistic Concurrency to prevent an unknown user from overwriting data from the database.
* **Transaction:** EF's transaction management capabilities automate the querying and saving of data. Furthermore, you can customize the way that transactions are managed.
* **Caching:** First-level caching of entities is supported out of the box in the EF. Repeated queries will retrieve data from the cache rather than the database in this case.
* **Built-in Conventions:** EF conforms to the conventions of configuration programming and has a set of default settings that automatically configure the model.
* **Configuration:** By using the data annotation attribute or Fluent API, we can configure the EF model and override the default conventions.
* **Migrations:** EF provides migration commands that are executable on the command-line interface or NuGet Package Manager Console to incrementally update the database schema to keep it in sync with the application's data model.

**What are the main components of Entity Framework Architecture?**

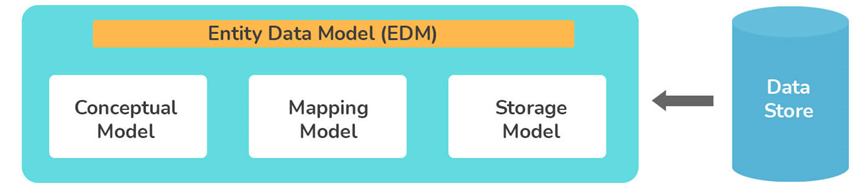
Entity Framework Architecture consists of the following components:



* **Entity Data Model (EDM):** EDMs abstract logical or relational schema and expose conceptual schema of data with a three-layered model, i.e., Conceptual (C-Space), Mapping (C-S Space), and Storage (S - Space).
* **LINQ to Entities (L2E):** L2E is basically a query language generally used to write queries against the object model. The entities defined in the conceptual model are returned by L2E.
* **Entity SQL (E-SQL):**Similar to L2E, E-SQL is another query language (for EF6 only). The developer must however learn it separately since it is more difficult than L2E. Internally, E-SQL queries are translated or converted to data store-dependent SQL queries. EF is used for converting E-SQL queries to their respective datastore queries, such as T-SQL.
* **Entity Client Data Provider:** This layer's main task is to convert E-SQL or L2E queries into SQL queries that the database understands. In turn, the ADO.Net data provider sends and retrieves data from the database.
* **Net Data Provider:**It uses standard ADO.NET to enable interaction with the database.
* **Object Service:** It is a service that facilitates access to a database, and returns data for analysis when necessary. By using it, you are able to translate data coming from entity clients into entity object structures.

**Explain different parts of the entity data model.**

The Entity Data Model consists of 3 core components that form the basis for Entity Framework. The three main components of EDM are as follows:



* **Conceptual Model:** It is also referred to as the Conceptual Data Definition Language Layer (C-Space). Typically, it consists of model classes (also known as entities) and their relationships.   Your database table design will not be affected by this. It makes sure that business objects and relationships are defined in XML files.
* **Mapping Model:**It is also referred to as the Mapping Schema Definition Language layer (C-S Space). Information about how the conceptual model is mapped to the storage model is usually included in this model. In other words, this model enables the business objects and relationships defined at the conceptual layer to be mapped to tables and relationships defined at a logical layer.
* **Storage Model:** It is also referred to as the Store Space Definition Language Layer (S-Space). Schematically, it represents the storage area in the backend. Therefore, the storage model is also known as a database design model that is composed of tables, keys, stored procedures, views, and related relationships.

**Explain what the .edmx file contains.**

First of all, a database lets you reverse engineer a model from an existing database. Entity Framework Designer is used to view and edit models stored and created in EDMX files (.edmx extensions). Using the EDMX file, you automatically generate classes that you can interact with within your application.

EDMX files represent conceptual models, storage models, and their mappings. This file contains all the mapping information between SQL tables and objects. In addition, it also includes essential information required for rendering models graphically with ADO.NET Entity Data Designer. Furthermore, it is divided into three divisions, CSDL, MSL, and SSDL.

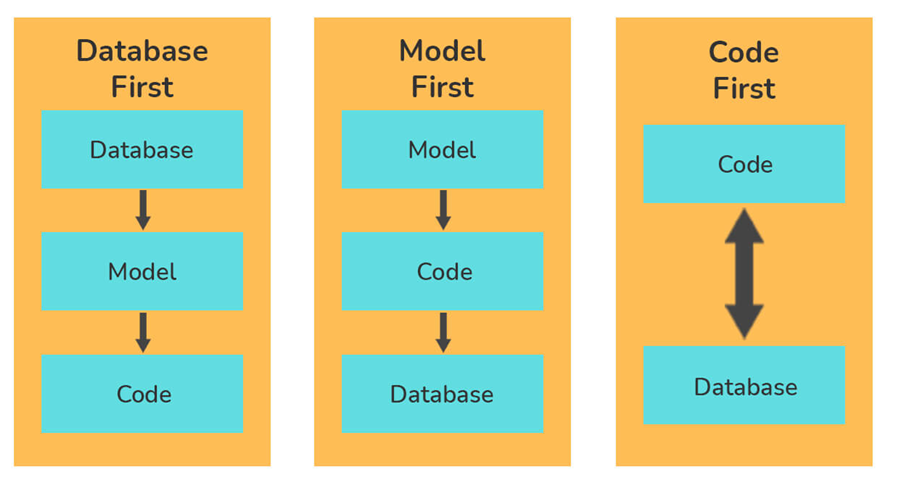
**What do you mean by migration? Write its type.**

Migration is a tool that was introduced in EF to update the database schema automatically when a model is modified without losing any data or other objects. Migrate Database To Latest Version is a new database initializer used by it. Entity Framework offers two types of migration:

* **Automated Migration:** Entity Framework 4.3 was the first to introduce automated migration so you don't have to manually migrate databases every time you alter a domain class. For example, you must also change the domain classes for each time you make a change, but with automated migration, you can simply run a command through the Package Manager Console.
* **Code-based Migration:** When you use a code-based migration, you can configure additional aspects of the migration, like setting the default value of a column, configuring a computed column, etc.

**What are different types of Entity framework approaches?**

Three different approaches to implement Entity Framework are as follows:



* **Code First Approach:**The Code First approach primarily uses classes to create the model and its relations, which are then used to create a database. This way, developers can work in an object-oriented manner without considering the database structure. By following this model, developers first write POCO classes and then use these classes to create the database. Code First is the method used by most developers using Domain-Driven Design (DDD).
* **Model First Approach:**In contrast, the Model First approach uses ORM to build model classes and their relationships. Following the successful creation of the model classes and relationships, the physical database is created using these models.
* **Database-First Approach:**In Entity Framework, Database First approach is used to build entity models based on existing databases and reduce the amount of code required. By using this approach, domain and context classes can be created based on existing classes.

**Which according to you is considered the best approach in Entity Framework?**

It is impossible to define one approach as the optimal approach when using the Entity Framework. Project requirements and the type of project determine which development approach should be used. Database First is a good approach if there is a database present. Model First is the optimal choice if no database and model classes exist. As long as the domain classes are available, the Code First method is the best choice.

**What do you mean by the term navigation property in the entity framework?**

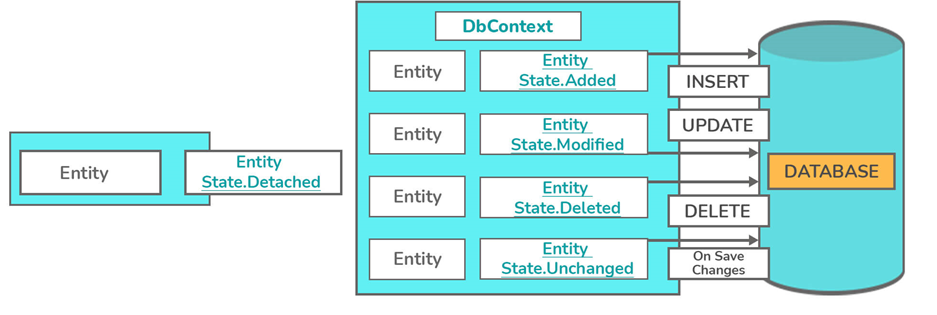
A foreign key relationship in the database is represented by the navigation property supported by the Entity Framework. It is possible to specify relationships between entities in a database using this property type. Relationships are defined in a way as to remain coherent in object-oriented code.

**What are different entity states in EF?**

There are five possible states where an entity can exist:

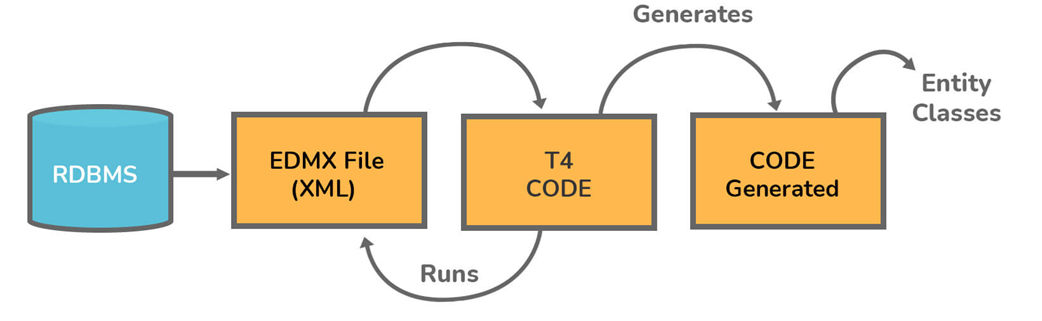
* **Added:** It is a state in which an entity exists within the context but does not exist within the database. When the user invokes the SaveChanges method, DbContext usually generates an INSERT SQL query to insert the data into the database. Upon successful completion of the SaveChanges method, the entity's state changes to unchanged.
* **Deleted:** This state indicates that the entity is marked for deletion has not been removed from the database. Also, it indicates the existence of the entity in the database. When the user invokes the SaveChanges method, DbContext usually generates a DELETE SQL query to delete or remove the entity from the database. Upon successful completion of the delete operation, DbContext removes the entity.
* **Modified:**When the entity is modified, its state becomes Modified. Also, it indicates the existence of the entity in the database. When the user invokes the SaveChanges method, DbContext usually generates an UPDATE SQL query to update the entity from the database. Upon successful completion of the SaveChanges method, the entity's state changes to unchanged.
* **Unchanged:** Since the context retrieved the entity's property values from the database, the values have not changed. This entity is ignored by SaveChanges.
* **Detached:**This state indicates that the entity is not tracked by the DbContext. If an entity was created or retrieved outside the domain of the current instance of DbContext, then its entity state will be Detached.

The following diagram represents the different entity states in Entity Framework:



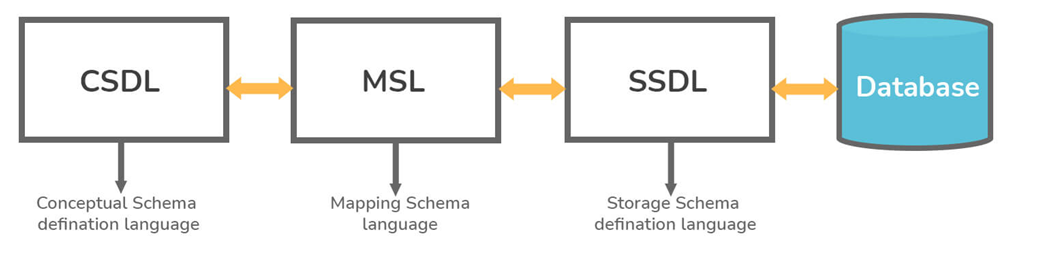
**Write the importance of the T4 entity in Entity Framework.**

In Entity Framework code generation, T4 files are crucial. EDMX XML files are read by T4 code templates, which generate C# behind code. The generated C# behind code consists only of your entity and context classes.



**Explain CSDL, SSDL, and MSL sections in an Edmx file?**

* **CSDL:** This stands for Conceptual Schema Definition Language. Basically, it's a conceptual abstraction that is exposed to the application. In this file, you will find a description of the model object.
* **SSDL:** This stands for Storage Schema Definition Language. In this section, we define the mapping to our RDBMS data structure.
* **MSL:**This stands for Mapping Schema Language. SSDL and CSDL are connected by it. It bridges the gap between the CSDL and SSDL or maps the model and the storage.



**Explain the ways to increase the performance of EF.**

Entity Framework's performance is enhanced by following these steps:

* Choose the right collection for data manipulation.
* Do not put all DB objects into one entity model.
* When the entity is no longer required, its tracking should be disabled and altered.
* Use pre-generating Views to reduce response time for the first request.
* Don't fetch all fields unless needed.
* Whenever possible, avoid using Views and Contains.
* Bind data to a grid or paging only by retrieving the number of records needed.
* Optimize and debug LINQ queries.
* Whenever possible, use compiled queries.

**Write some XML generation methods provided by the dataset object.**

DataSet objects provide the following methods for generating XML:

* **ReadXml():**This method reads an XML document into a DataSet object.
* **GetXml():** This method returns a string containing an XML document.
* **WriteXml():**This method writes XML data to disk.

**What do you mean by the migration history table in Entity Framework?**

EF6's Migration's history table (\_\_MigrationHistory) is basically a database table that is used to store data about migrations applied to a database by Code First Migrations. A table like this is created when the first migration is applied to the database. Within a given database, this table contains meta-data describing the EF Code First models' schema versions. When you used the Microsoft SQL Server database, this table was considered a system table in EF5.

**Explain how EF supports transactions.**

The SaveChanges() method in EF always wraps any operation involving inserting, updating, or deleting data into a transaction. Hence, you do not have to explicitly open the transaction scope.

**What do you mean by Deferred Execution in EF?**

Deferred Execution refers to the process of delaying the evaluation of an expression until its realized value is actually required. As a result, performance is greatly improved since unnecessary execution is avoided. Queries are deferred until the query variable or query object is iterated over a loop.

**Entity Framework Interview Questions for Experienced**

**1. Write difference between LINQ and Entity Framework.**

| **LINQ** | **Entity Framework** |
| --- | --- |
| In order to operate, LINQ relies only on SQL Server Databases. | In order to operate, the entity framework relies on several databases including SQL Server, Oracle, MYSQL, DB2, etc. |
| It generates a .dbml to maintain the relationship. | In this case, an .edmx file is generated first, then an .edmx file is maintained using three separate files- .csdl, .msl, and .ssdl. |
| DataContext enables you to query data. | ObjectContext, DbContext, and EntitySQL can all be used to query data. |
| Complex types are not supported. | Complex types are supported. |
| A database is not created from the model. | A database can be created from the model. |
| Application is developed more quickly using SQL Server. | Applications are developed more quickly using SQL Server and other databases like MYSQL, Oracle, DB2, etc. |
| It consists of a tightly coupled mechanism. | It consists of a loosely coupled mechanism. |
| Only one-to-one mappings are allowed. | One-to-one, one-to-many & many-to-many mappings are allowed. |
| It displays rapid development. | It takes longer to develop than LINQ, but it provides more capabilities. |

**Write the steps to retrieve data from database using Entity Framework in MVC.**

The following steps will show you how to retrieve data from a database in MVC (Model View Controller) using Entity Framework:

* As a first step, we must create a new project.
* The next step is to add an Entity Framework reference from the NuGet package manager.
* Then, a new class has to be created within the model inside the table structure.
* After that, we are required to add a connection string in the web.config.connection. It should be matched with the context.
* The next step is to open the Global.asax.cs class and add the new namespace of EF. We must then initialize the database.
* You will now need to right-click on the Controller folder and add a new controller, followed by a model reference in the section namespace.
* Last but not least, right-click on the controller's name and add the sections you want to retrieve.

**Explain the term dbcontext and dbset.**

**DbSet:** An entity set is represented by a DbSet class that can be used for creating, reading, updating, and deleting operations on it. Those DbSet type properties, which map to database tables and views, must be included in the context class (derived from DbContext).

**DbContext:** It is considered an essential class in EF API that bridges the gap between an entity or domain class and the database. Communication with the database is its primary responsibility.

**Difference between ADO.Net and Entity Framework.**

Below are the differences between Aadonet and Entity Framework:

* A few data layer codes are created by Ado.Net that Entity Framework doesn't create.
* Entity Framework, unlike ADO.Net, generates code for intermediate layers, data access layers, and mappings automatically. This results in a reduction in development time.
* On a performance basis, ADO.Net is more efficient and faster than Entity Framework.

**Explain the role of Pluralize and Singularize in the entity framework.**

Objects in Entity Framework are primarily assigned names using Pluralize and Singularize. This feature is available when adding a .edmx file. Entity Framework automatically assigns the Singular or Plural coding conventions when using this feature. In convention names, an additional 's' is added if there is more than one record in the object.

**What is the difference between Dapper and Entity Framework?**

.NET developers are allowed to work with relational data using domain-specific objects by object-relational mappers such as Entity Framework (EF) and Dapper. Performance-wise, Dapper is the King of Micro ORMs.

* **Dapper:**A simple micro ORM, Dapper is considered a powerful system used for data access in the .NET world. As a means to address and open-source their issues, the Stack Overflow team created Dapper. Adding this NuGet library to your .NET project allows you to perform database operations. In terms of speed, it is the king of Micro ORMs and is almost as fast as using raw ADO.NET data readers.
* **Entity Framework:**It is a set of .NET APIs used in software development for performing data access. It is [Microsoft's official tool](https://www.interviewbit.com/microsoft-interview-questions/) for accessing data.

**Comparison**

* According to NuGet downloads and performance, Dapper is the world's most popular Micro ORM. In contrast, Entity Framework is significantly slower than Dapper.
* In comparison to other ORMs, such as the Entity Framework, Dapper does not generate as much SQL, but it does an excellent job mapping from database columns to CLR properties.
* Since Dapper uses RAW SQL, it can be difficult to code, especially when multiple relationships are involved, but when a lot of data is involved and performance matters, it is worth the effort.
* Since Dapper uses IDbConnection, developers can execute SQL queries to the database directly rather than put data in other objects as they do in Entity Framework.

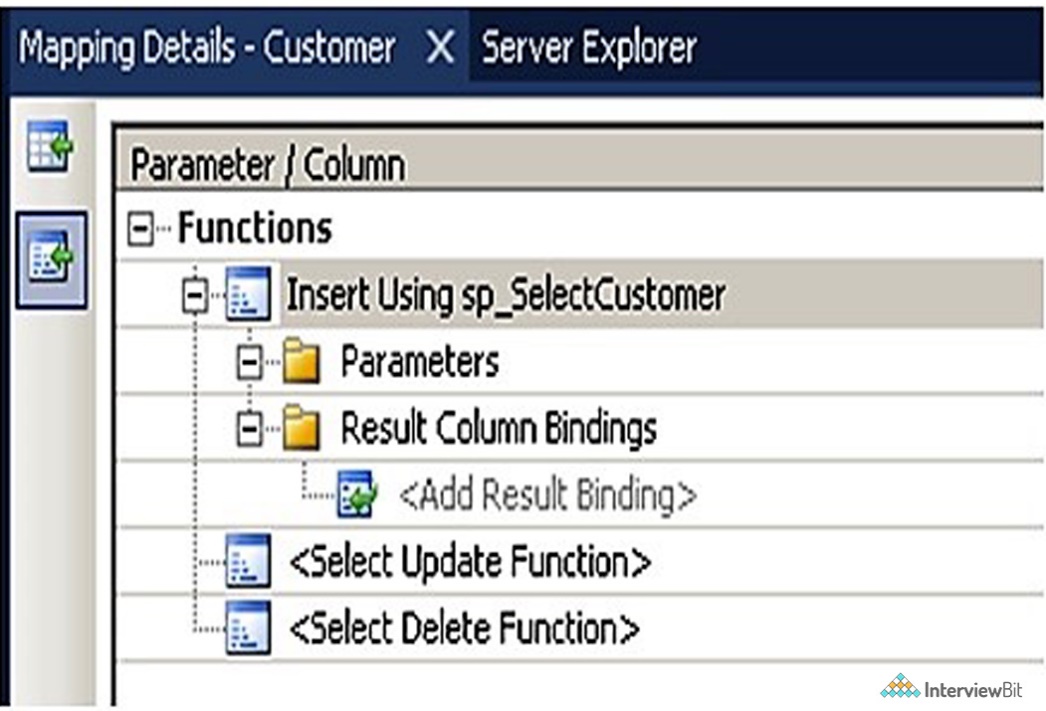
**Explain POCO Classes in EF.**

POCO stands for 'Plain Old CLR Objects'. Yet, it does not mean these classes are plain or old. A POCO class is defined as a class that contains no reference to the EF Framework or the .NET Framework at all. In EF applications, Poco entities are known as available domain objects.

POCO class is just like other normal .NET classes as these classes don't depend on any framework-specific base class, unlike the standard .NET class. Persistence-ignorant objects, or POCOs, support LINQ queries, which are supported by entities derived from the Entity Object itself. Both EF 6 and EF Core support POCO entities.

**In Entity Framework, what are the ways to use stored procedures?**

This figure shows how stored procedure mapping details can be used in EDMX:



**Explain database concurrency and the way to handle it.**

Database concurrency in EF means that multiple users can simultaneously modify the same data in one database. Concurrency controls help safeguard data consistency in situations like these.

Optimistic locking is usually used to handle database concurrency. We must first right-click on the EDMX designer and then change the concurrency mode to Fixed in order to implement locking. With this change, if there is a concurrency issue, we will receive a positive concurrency exception error.

**What are different types of loading available to load related entities in EF?**

Entity Framework offers the following types of loading:

* Eager Loading
* Lazy Loading
* Explicit Loading

**What do you mean by lazy loading, eager loading and explicit loading?**

* **Lazy Loading:** This process delays the loading of related objects until they are needed. During lazy loading, only the objects needed by the user are returned, whereas all other related objects are only returned when needed.
* **Eager Loading:** This process occurs when you query for an object and all of its related objects are returned as well. Aside from that, all related objects will load with the parent object automatically. When the Include method is used, eager loading can be achieved in EF6.
* **Explicit Loading:** Explicit loading occurs only when lazy loading is desired, even when lazy loading is disabled. We must explicitly call the relevant load method on the related entities to process explicit loading. When the Load method is used, explicit loading can be achieved in EF6.

**What are the pros and cons of different types of loading?**

**1. Lazy Loading**

**Pros**

* When the relationships are not too high, use Eager Loading. So you can reduce further queries on the server by using Eager Loading.
* If you know that related entities will be used everywhere with the main entity, use Eager Loading.

**Cons**

* Adding the extra lines of code to implement lazy load makes the code more complicated.
* It can affect a website's search engine ranking sometimes because the unloaded content is not properly indexed.

**2. Eager Loading**

**Pros**

* Upon executing the code, the system initializes or loads the resource.
* Additionally, related entities that are referenced by a resource must be pre-loaded.
* It is advantageous when resources need to be loaded in the background.
* It saves you time by avoiding the need to execute extra SQL queries.

**Cons**

* Since everything must be loaded to begin running, starting the application takes a longer time.

**Choosing the right tool**

* When you know you will use related entities with your main entity everywhere, use Eager Loading.
* You should use Lazy Loading whenever you have one-to-many collections.
* Use lazy loading only if you are sure you won't need related entities right away.
* When you are unsure about whether or not an entity will be used, use explicit loading after you have turned off Lazy Loading.

**Write different types of inheritance supported by Entity Framework.**

In Entity Framework, inheritance is primarily divided into three types:

* **Table per Hierarchy (TPH):**The TPH inheritance representation shows one table per inheritance hierarchy class. A discriminator column also aids in distinguishing between inheritance classes. This is Entity Framework's default inheritance mapping technique.
* **Table per Type (TPT):**In this inheritance method, each domain class has its own table.
* **Table per Concrete Class (TPC):** TPC demonstrates a single table per concrete class, but does not include the abstract class. Because of this, if an abstract class is inherited by many concrete classes, then the tables in all those concrete classes will have the same properties as that of an abstract class.

**Explain Complex Type in Entity Framework.**

Complex types are defined as the non-scalar properties of entity types that assist in organizing scalar properties within entities. In addition to scalar properties, complex types may also have other complex type properties. Instances of complex types are complex objects.

**What do you mean by Micro ORM?**

Rather than creating database schemas, modifying database schemas, tracking changes, etc., Micro ORMs focus on working with database tables. EF 6.x and EF Core provide a full set of capabilities and features, making them ORMs.

**Explain EF Data access Architecture.**

There are two types of Data Access Architecture supported by the ADO.NET Framework:

* **Disconnected data access:** Disconnected data access is possible with the Data Adapter object. Datasets work independently of databases, and the data can be edited.
* **Connected data access:** A Data Reader object of a Data Provider allows you to access linked data. Data can be accessed quickly, but editing is not permitted.

**What do you mean by SQL injection attack?**

SQL injection is a method that hackers use to access sensitive information from an organization's database. This application-layer attack is the result of inappropriate coding in our applications, allowing hackers to inject SQL statements into your SQL code.

The most common cause of SQL Injection is that user input fields allow SQL statements to pass through and directly query the database. ADO.NET Data Services queries are commonly affected by SQL Injection issues.

**What is the best way to handle SQL injection attacks in Entity Framework?**

The injection-proof nature of Entity Framework lies in the fact that it generates parameterized SQL commands that help prevent our database from SQL injections.

By inserting some malicious inputs into queries and parameter names, one can generate a SQL injection attack in Entity SQL syntax. It is best to never combine user inputs with Entity SQL commands text to prevent or avoid this problem.

**Explain the ObjectSet in EF.**

ObjectSet is generally considered as a specific type of data set that is commonly used to read, update, create, and remove operations from existing entities. Only the ObjectContext instance can be used to create it. No Entity SQL method is supported by it.

**Write the namespace that is used to include .NET Data provider for SQL server in .NET code.**

NET Data Provider for SQL Server is included in .NET code by using the namespace System.Data.SqlClient.

**Explain EDM and write the process to create it.**

In Entity Framework, EDM refers to the '**Entity Data Model**'. It is considered as an entity-relationship prototype that assigns some basic prototypes for the data using various modeling procedures. Moreover, it is defined as a set of principles pertaining to the formation of data, regardless of how it is collected. Shortly, it's just a simple link or connection created between the database and the prototype. The steps for creating an Entity Data Model are as follows:

* Right-click on a project in the Solution Explorer.
* Select the Add>New Item option from the menu.
* Select the ADO.Net Entity Data Model arrangement or template.
* Please enter a name and click the 'Add' button.

**What do you mean by DbEntityEntry Class in EF?**

An important class, DbEntityEntry helps you retrieve a variety of information about an entity. DbContext offers the Entry method for retrieving an instance of DBEntityEntry of a specific entity.

**Example:**

DbEntityEntry studentEntry = dbcontext.Entry(entity);

You can access the entity state, as well as the current and original values of all properties of an entity using the DbEntityEntry. EntityState can be set using the DbEntityEntry, as shown below.

context.Entry(student).State = System.Data.Entity.EntityState.Modified;

What is ADO.NET Entity Framework?

[ADO.NET Entity Framework](https://www.scholarhat.com/tutorial/entityframework/a-brief-version-history-of-adonet-entity-framework) is an ORM framework that empowers developers to work with various relational databases like [SQL Server](https://www.scholarhat.com/tutorial/sqlserver), Oracle, DB2, MYSQL, etc. It allows developers to deal with data as objects or entities. Using the [Entity Framework](https://www.scholarhat.com/tutorial/entityframework/introduction-to-entity-framework), developers issue queries using [LINQ](https://www.scholarhat.com/tutorial/linq), then retrieve and manipulate data as strongly typed objects using C# or VB.NET Framework.

What other O/RMs you can use with .NET-based applications?

The following O/RMs, you can use with .NET-based applications:

* + Entity Framework 6.x
  + Entity Framework Core
  + Dapper
  + N Hibernate

What are Micro O/RMs?

A Micro ORM is architected to focus on the most important task of working with database tables instead of creating, modifying the database schema, tracking changes, etc. EF 6.x and EF Core both are O/RMs since they provide a full set of features.

What is Dapper?

Dapper is a simple/ micro ORM for the .NET world. Dapper was created by the StackOverflow team to address their issues and open-source it. It's a NuGet library that can be added to any .NET project for database operations.

What is an SQL injection attack?

A [SQL injection](https://www.scholarhat.com/tutorial/sqlserver/sql-injection-attacks) attack is an attack mechanism used by hackers to steal sensitive information from the database of an organization. It is the application layer (means front-end) attack that takes benefit of inappropriate coding of our applications that allows a hacker to insert SQL commands into your code that is using SQL statements.

SQL Injection arises since the fields available for user input allow SQL statements to pass through and query the database directly. SQL Injection issue is a common issue with an ADO.NET Data Services query.

How to handle SQL injection attacks in Entity Framework?

Entity Framework is injection safe since it always generates parameterized SQL commands which help to protect our database against SQL Injection.

An SQL injection attack can be made in Entity SQL syntax by providing some malicious inputs that are used in a query and in parameter names. To avoid this, you should never combine user inputs with Entity SQL command text.

What are various approaches to domain modeling in Entity Framework?

**There are three ways to approach Entity Framework:**

1. Database-first: If we start with a database, Entity Framework generates the code.

2. Model-first: If we start with a visual model, Entity Framework generates both the database and code.

3. Code-first: If we start with code, Entity Framework generates the database.

What are POCO classes?

The term POCO does not mean to imply that your classes are either plain or old. The term POCO simply specifies that the POCO classes don’t contain any reference that is specific to the entity framework or [.NET framework](https://www.scholarhat.com/tutorial/net/understanding-net-framework-45-architecture).

POCO (Plain Old CLR Object) entities are existing domain objects within your application that you use with Entity Framework.

What is the proxy object?

An object that is created from a POCO class or entities generated by the Entity Framework to support change tracking and [lazy loading](https://www.scholarhat.com/tutorial/entityframework/difference-between-lazy-loading-and-eager-loading), is known as a proxy object.

There are some rules for creating a proxy class object:

* + The class must be public and not sealed.
  + Each property must be marked as virtual.
  + Each property must have a public getter and setter.
  + Any collection navigation properties must be typed as ICollection <T>.

What are the various Entity States in EF?

Each and every entity has a state during its lifecycle which is defined by an enum (EntityState) that has the following values:

* + Added
  + Modified
  + Deleted
  + Unchanged
  + Detached

What are the different types of inheritance in Entity Framework?

[Inheritance in the Entity Framework](https://www.scholarhat.com/tutorial/entityframework/understanding-inheritance-in-entity-framework) is similar to inheritance for classes in C#. In Entity Framework, you can map an inheritance hierarchy to single or multiple database tables based on your requirements. EF supports three types of inheritances:

* + Table-per-Hierarchy (TPH)
  + Table-per-Type (TPT)
  + Table-per-Concrete-Type (TPC)

What are the various approaches in Code First for model designing?

In [Entity Framework Code First approach](https://www.scholarhat.com/tutorial/entityframework/understanding-entity-framework-code-first-migrations), our POCO classes are mapped to the database objects using a set of conventions defined in Entity Framework. If you do not want to follow these conventions while defining your POCO classes, or you want to change the way the conventions work then you can use the fluent API or data annotations to configure and to map your POCO classes to the database tables. There are two approaches, which you can use to define the model in EF Code First:

What C# Datatype is mapped with which Datatype in SQL Server?

The following table has the list of [C# Datatype](https://www.scholarhat.com/tutorial/csharp/datatype-in-csharp) mapping to the corresponding SQL Server Datatype:

C# Data Type

indexOf()

int

int

string

nvarchar(Max)

decimal

decimal(18,2)

float

real

byte[]

varbinary(Max)

datetime

datetime

bool

bit

byte

tinyint

short

smallint

long

bigint

double

float

char

No mapping

sbyte

No mapping

object

No mapping

What is Code First Migrations in Entity Framework?

**The code-first**approach allows you to define model classes as per the Domain requirements via POCOs. Hence, you have complete control over the classes being written or Implemented.

[Code First Migrations](https://www.scholarhat.com/tutorial/entityframework/entity-framework-6-code-first-migrations-with-multiple-data-contexts) allows you to create a new database or to update the existing database based on your model classes by using the Package Manager Console that exists within Visual Studio.

What is the Migrations History Table?

In EF6, the Migrations history table (\_\_MigrationHistory) is a part of the application database and is used by Code First Migrations to store details about migrations applied to a database. This table is created when you apply the first migration to the database. This table stores metadata describing the schema version history of one or more EF Code First models within a given database. In EF 5, this table was a system table when you use the Microsoft SQL Server database.

In EF 5, this table was a system table when you use the Microsoft SQL Server database.

What is automatic migration?

IEntity Framework supports automatic migration so you don't need to migrate model changes manually. So, when you will run the application, it will be handled by the EF.

What is DbSet?

DbSet is a typed entity set that is used to perform create, read, update, and delete operations on a particular entity. DbSet can only be created from a DbContext instance. DbSet does not support the Entity SQL methods.

What is ObjectSet?

ObjectSet is a typed entity set that is used to perform create, read, update, and delete operations on a particular entity. ObjectSet is can only be created from an ObjectContext instance. ObjectSet does not support the Entity SQL methods.

How to execute plain SQL in EF6?

EF6 allows us to execute raw SQL queries to query the database. The following methods are used to execute raw SQL queries:

* + DbSet.SqlQuery()
  + DbContext.Database.SqlQuery()
  + DbContext.Database.ExecuteSqlCommand()

How does EF support Transaction?

**I**n EF, whenever you execute SaveChanges() to insert, update, or delete data into the database, it wraps that operation in a transaction. So, you don’t need to open a transaction scope explicitly.

What are the features of Entity Framework?

Querying, Cross-Platform, Modeling, Saving, Change tracking, Concurrency, Caching, Transaction, Configuration, Built-in conventions, and Migrations. These are the features of Entity Framework.

What is the purpose of the conceptual model?

The conceptual model is also called the Conceptual Data Definition Language Layer or C-Space. It contains the entities/model classes and also their relationships. This all thing doesn't affect our database table design. Because It ensures that business objects and relationships are demarked in XML files.

What is the purpose of the mapping model?

The mapping model is also called the Mapping Schema Definition Language layer or C-S Space. It includes information about the mapping of the conceptual model to the storage model. You can say that this model maps the business objects and their relationships at the conceptual layer to tables and relationships at the logical layers.

What is the purpose of the storage model?

The storage model is also called the Store Space Definition Language Layer or S-Space. The storage area in the backend is always represented by this model. That's why it's also called the database design model composing stored procedures, keys, views, tables, and related relationships.

What is meant by migration in Entity Framework?

The Entity Framework introduced the migration tool for automatically updating the database schema. There are two types of migration provided by Entity Framework automated migration and code-based migration.

What does the .edmx file contain?

By using the EDMX files, Classes can be automatically generated to interact within the application. EDMX file represents the conceptual models, storage models, and the mappings. All information about SQL objects and tables is contained in it. The crucial information needed to render models graphically with ADO.NET is also contained. Its 3 types are MSL, CSDL, and SSDL.

Mention some XML generation methods that the dataset object provides.

**The below methods are provided by Dataset objects to generate XML:**

1.GetXml(): A string containing an XML document is provided by this method.

2.ReadXml(): an XML document is read into a Dataset object by this method.

3. Write XML (): The XML data is written to disk by this method.

Why is the T4 entity important in the Entity Framework?

[T4 files](https://www.scholarhat.com/tutorial/entityframework/using-t4-templates-in-entity-framework) are crucial in the code generation of Entity Framework. T4 code templates are used to read the EDMX XML files. The C# behind the code is then generated by the T4 files. Just entity and the context classes are contained in the generated C# behind the code.

What is meant by the navigation property in Entity Framework?

It represents a foreign key relationship in Entity Framework's database. The relationships between the entities in a database can be specified with the help of this property type. In object-oriented programming, the relationships are defined such that they remain coherent.

What is meant by deferred execution in Entity Framework?

We can wait for the evaluation of any expression till the time we want its realized value to appear. Hence we can improve the performance significantly by avoiding unnecessary execution. Until the query object or query variable iterates over a loop, queries get deferred.

What do you mean by database concurrency?

It arises when multiple users are accessing and modifying the same data simultaneously in the same database. Concurrency controls keep the consistencyof data protected in such situations.

How can you handle database concurrency?

You can implement optimist locking to handle the database concurrency. You can implement the locking by right-clicking on the EDMX designer and setting the concurrency model to Fixed. If any case any concurrency issue exists, a positive concurrency exception error arises.

Explain SSDL, CSDL, and MSL divisions in an EDMX file.

1. **SSDL**stands for Storage Schema Definition Language. Mapping to RDBMS data structure is defined in this division.

2. **CSDL**stands for Conceptual Schema Definition Language. It is an app that exposes conceptual abstraction. The model object's description can be obtained in this division.

3. **MSL**stands for Mapping Schema Language. MSL connects CSDL and SSDL. Or we can say that it maps the model to the storage.

Explain the terms dbset and dbcontext.

**1. dbset**-operations can be created, updated, read, and deleted on any entity set in a dbset class. We must include the dbset type properties mapping to the database tables and views in the context class (from dbcontext).

2. **dbcontext**- It is an important class in Entity Framework API. This is used to connect a domain class or entity and the database. Its main responsibility is to communicate with the database.

What is meant by the Object Set in Entity Framework?

An object set is a specific type of entity set that can be used to read, update, create, and remove operations from any existing entity. It can only be created by using an Object Context instance. It does not support any kind of Entity SQL method.

Explain the eager loading, lazy loading, and explicit loading in detail.

1. Eager loading**:**In this loading, all the related objects are also returned along with the object's query. All related objects get loaded automatically along with the parent object. You can achieve eager loading in EF 6 by using the Include method.

2. Explicit loading**:**This loading occurs only when we desire lazy loading. The relevant load method should be explicitly called for processing explicit loading. We can achieve explicit loading in EF 6 by using the load method.

3. [Lazy loading](https://www.scholarhat.com/tutorial/entityframework/difference-between-lazy-loading-and-eager-loading): This loading process of related objects is delayed till the time we need them. Only objects you need are returned in the lazy loading. Simultaneously, the other related objects get returned only when we need them.

Explain the role of singularizing and pluralizing in the Entity Framework.

The important responsibility of singularizing and pluralizing in Entity Framework is assigning the objects meaningful naming conventions. This feature can be processed through the .edmx file. The coding conventions will assigned to the singular and plural while this feature is used. The convention names have an extra 's' if there are two or more than two records within an object.

What is meant by micro O/RMs in Entity Framework?

The micro ORM is not made to create database schemas, track changes, modify database schemas, etc. Instead, it is primary function is to work with database tables. Because Entity Framework Core and Entity Framework 6 have a complete set of functionalities and features, they are referred to as O/RMs.

What is meant by Optimistic Locking?

It is the process of reading a record and noting a version number, timestamp, date, or hash. Then, you can check that the record hasn't changed before writing the code back. While writing the record back, we filter the update on the version, ensuring that it's atomic. Then the version is updated in a single hit.

How does EF support transactions?

In Entity Framework, whenever we perform Save Changes () to put in, modernize, or delete data into the database, it wraps that process in a transaction.

Which namespace is used for the inclusion of the .NET data provider for the SQL server in your .NET code?

We use namespace as - System.Data.SqlClient for the inclusion of a .NET data provider for SQL server in our .NET code.

When should modeling entry approaches be used?

The Code First approach is better when we have the domain classes already with us. On the other hand, the Database First approach is the right fit when we have a database. And the Model First approach is used when we don't have any database or model classes.

Mention the primary functions of the Entity Framework.

EF enables the mapping of domain classes to the database schema. EFAlterations in the entities are kept on track.EF enables the execution of LINQ queries to SQL. In EF the changes in stats are stored in the database.

What are the advantages of the Model First Approach?

Model First Approach provides flexibility for designing the Entity Models separately and offers options to improve them in further stages. This approachdoes not use many databases because we can create model classes by drawing them using the EDMX designer.

Which, according to you, is the best approach in the Entity Framework?

There is no special approach that can be referred to as the best approach in Entity Framework. The selection of the approaches primarily relies on the project requirements and the project's types. If there is the database's existence, then it is good to use the Database First approach. If there is no database and the model classes, then the model-first approach is the best selection. If there is the availability of the domain classes, the Code-First approach is the most suitable choice.

What do you understand by LINQ to Entities?

It is defined as one of the popular query languages in Entity Framework. It mainly helps write queries against the objects to retrieve entities based on the conceptual model's definitions.

What do you understand by the Entity SQL?

It is an alternate query language that is similar to a LINQ for Entities. However, it is complicated than LINQ to Entities. Programmers who want to use this language will have to learn it separately.

How would you handle large volumes of data using Entity Framework?

Entity Framework can handle large data volumes through various strategies. One approach is to use the As NoTracking method, which prevents  Entity Framework from tracking changes in entities, reducing memory usage and improving performance. The second strategy involves using stored procedures for complex queries or operations, as they are faster than LINQ queries.

What do you know about ComplexType in Entity Framework?

ComplexType is a non-scalar property of entity types. This type helps users to assign scalar relationships between entities.

**What is Entity Framework (EF)?**

* + Entity Framework is an ORM (Object-Relational Mapping) framework provided by Microsoft. It enables developers to work with databases using .NET objects and eliminates the need for most of the data-access code that developers usually need to write.

**Explain the different approaches to working with Entity Framework.**

* + Entity Framework supports three different approaches:
    - **Database-First:** Creates entity classes based on an existing database schema.
    - **Model-First:** Allows creating entity classes and relationships visually in the EDMX designer.
    - **Code-First:** Enables developers to define entities, relationships, and configurations using code.

**What are the key components of Entity Framework Core?**

* + Entity Framework Core includes:
    - **DbContext:** Represents a session with the database and a gateway to perform CRUD operations.
    - **DbSet:** Represents a collection of entities of a particular type within the context.
    - **Entity:** Represents a class that corresponds to a table in the database.
    - **LINQ to Entities:** Allows querying entities using LINQ.

**Explain Lazy Loading and Eager Loading in Entity Framework.**

* + **Lazy Loading:** Lazy loading is the technique where related entities are loaded from the database only when they're accessed for the first time. It's the default behavior in Entity Framework Core, and it helps in avoiding unnecessary database calls.
  + **Eager Loading:** Eager loading is the technique where related entities are loaded along with the main entity using the Include method or using projection. It reduces the number of database round trips but might fetch unnecessary data.

**What is the difference between Add, Attach, and Update methods in Entity Framework Core?**

* + **Add:** Used to add a new entity to the context. It changes the state of the entity to Added.
  + **Attach:** Used to re-attach an existing entity to the context. It changes the state of the entity to Unchanged.
  + **Update:** Used to update an existing entity in the context. It changes the state of the entity to Modified.

**What is the purpose of migrations in Entity Framework Core?**

* + Migrations in Entity Framework Core allow developers to evolve the database schema over time in a structured way. They enable creating, updating, and reverting database schema changes, keeping the database schema in sync with the application's data model.

**How does Entity Framework handle relationships between entities (one-to-one, one-to-many, many-to-many)?**

* + Entity Framework supports various types of relationships:
    - **One-to-One:** One entity instance is related to exactly one instance of another entity.
    - **One-to-Many:** One entity instance is related to multiple instances of another entity.
    - **Many-to-Many:** Multiple instances of one entity are related to multiple instances of another entity using a joining table.

**Explain the advantages and disadvantages of using Entity Framework.**

* + **Advantages:** Rapid development, reduced amount of boilerplate code, easy to maintain, supports various database providers, LINQ support for querying data.
  + **Disadvantages:** Performance overhead, potential for generating inefficient queries, complexity in some advanced scenarios, learning curve.

**What is the purpose of the DbContext class in Entity Framework?**

* + The DbContext class in Entity Framework represents the session with the database and provides a set of APIs to perform CRUD (Create, Read, Update, Delete) operations on entities. It also manages the entity objects during their lifecycle.

**Explain how Entity Framework handles transactions.**

* + Entity Framework Core supports transactions through the SaveChanges method. Multiple changes within a single SaveChanges call are made within a transaction. Additionally, you can explicitly manage transactions using Database.BeginTransaction and Transaction.Commit/Transaction.Rollback methods.

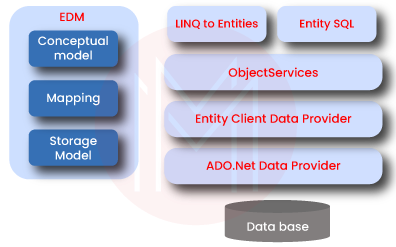
**List down the features of Entity Framework.**

* Querying
* Cross-Platform
* Modeling
* Saving
* Change tracking
* Concurrency
* Caching
* Transaction
* Configuration
* Built-in conventions
* Migrations

**What are the three core components of the entity data model?**

The three main components of the entity data model forming the basis of the Entity Framework are given below:

* Conceptual model
* Mapping model
* Storage model



**Mention the main parts of Entity Framework Architecture?**

The following are the main components of Entity Framework architecture:

* LINQ to Entities (L2E)
* Entity Data Model (EDM)
* Entity SQL (E-SQL)
* Net data provider
* Entity client data provider
* Object service

|  |
| --- |
| ***If you want to enrich your career and become a professional in .Net Programming Language, then visit Mindmajix - a global online training platform: "***[***.Net Training***](https://mindmajix.com/net-training)***" This course will help you to achieve excellence in this domain.*** |

**What is the purpose of the conceptual model?**

It is also called the Conceptual Data Definition Language Layer or C-Space. It contains the entities/model classes as well as their relationships. This doesn't affect our database table design. It ensures that business objects and relationships are demarked in XML files.

**What is the purpose of the mapping model?**

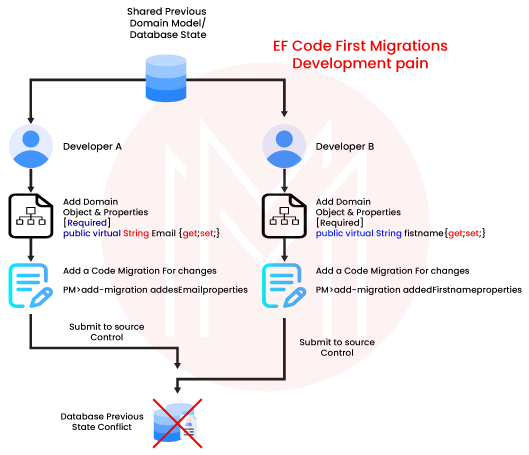
It is also called the Mapping Schema Definition Language layer or C-S Space. This model includes the information about the mapping of the conceptual model to the storage model. We can say that this model maps the business objects and their relationships at the conceptual layer to tables and relationships at the logical layer.

**What is the purpose of the storage model?**

It is also called the Store Space Definition Language Layer or S-Space. The storage area in the backend is represented by this model. That's why it's also referred to as the database design model composing stored procedures, keys, views, tables, and related relationships.

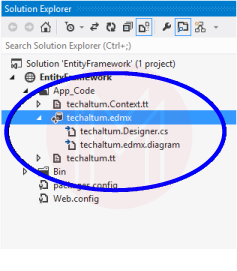
**What is meant by migration in Entity Framework?**

The Entity Framework introduced the migration tool for automatically updating the database schema whenever a model modifies without losing data. The two types of migration provided by Entity Framework are automated migration and code-based migration.



**What does the .edmx file contain?**

Classes can be automatically generated to interact within our application using the EDMX files. The conceptual models, storage models, and the mappings are represented by EDMX files. All the information about SQL objects and tables is contained in it. The essential information needed to render models graphically with ADO.NET is also contained. Its 3 divisions are MSL, CSDL, and SSDL.



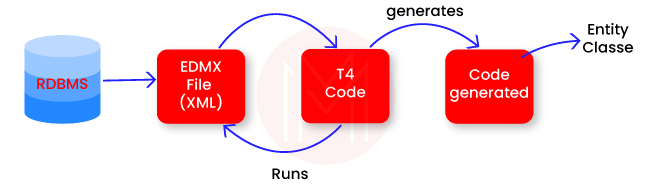
**Mention some XML generation methods that the dataset object provides.**

The following methods are provided by Dataset objects to generate XML:

**GetXml():** A string containing an XML document is provided by this method.c  
**ReadXml():** an XML document is read into a Dataset object by this method.  
**WriteXml():** The XML data is written to disk by this method.

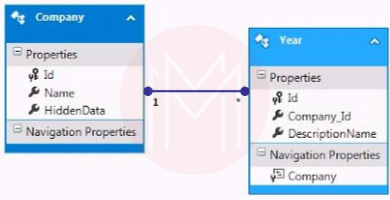
**Why is the T4 entity important in the Entity Framework?**

T4 files are very important in the code generation of Entity Framework. The T4 code templates read the EDMX XML files. The C# behind the code is then generated by the T4. Just our entity and the context classes are contained in the generated C# behind code.



**What is meant by the navigation property in Entity Framework?**

The navigation property represents a foreign key relationship in Entity Framework's database. The relationships among the entities in a database can be specified with the help of this property type. In the object-oriented code, the relationships are defined such that they remain coherent.



**What is meant by deferred execution in Entity Framework?**

We can delay the evaluation of any expression till the time we want its realized value to appear. Therefore we can improve the performance significantly by avoiding unnecessary execution. Until the query object or query variable iterates over a loop, queries get deferred.

**What do you mean by database concurrency?**

Database concurrency arises when multiple users are accessing and modifying the same data simultaneously in the same database. And the concurrency controls are the systems by which the consistency of data is protected in such situations.

**How can you handle database concurrency?**

We can implement optimist locking to handle the database concurrency. We can implement the locking by right-clicking on the EDMX designer and setting the concurrency model to Fixed. If any concurrency issue exists, a positive concurrency exception error arises.

**What are various types of properties supported in Entity Framework?**

Entity Framework supports mainly 3 types of properties. And they are given below:

* Complex property
* Navigational property
* Scalar property
* **Entity Framework Interview Questions for Experienced**

**Explain SSDL, CSDL, and MSL divisions in an EDMX file.**

* **SSDL-** Its full form is Storage Schema Definition Language. Mapping to the RDBMS data structure is defined in this division.
* **CSDL-**Its full form is Conceptual Schema Definition Language. It is an application that exposes conceptual abstraction. The model object's description can be found in this division.
* **MSL-** Its full form is Mapping Schema Language. It connects CSDL and SSDL. Or we can say that it maps the model to the storage.



**How can the Entity Framework's performance be increased?**

Its performance can be increased by:

* Not pulling all databases into a single entity model.
* Choosing the right data collection for manipulation.
* Using pre-generating views to reduce the first request's response time.
* Disabling or altering the tracking of entity when no longer required.
* Not fetching all fields without use.
* Optimizing and debugging LINQ queries.
* Using compiled queries as much as possible.

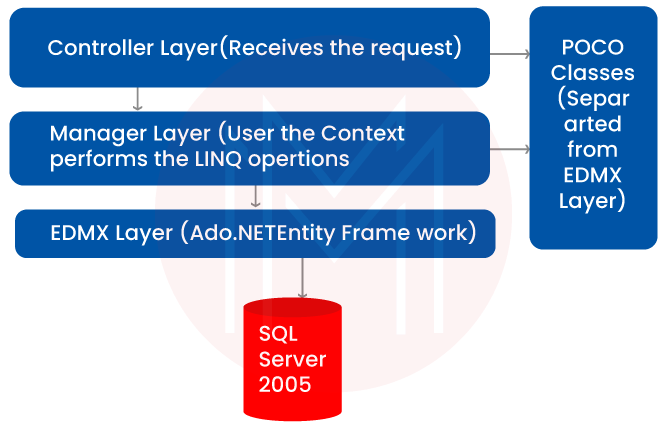
**Explain the terms dbset and dbcontext.**

**Dbset-** operations can be created, updated, read, and deleted on any entity set in a dbset class. We must include the dbset type properties mapping to the database tables and views in the context class (from dbcontext).  
**Dbcontext-**It is an important class in Entity Framework API. This is used to connect a domain class or entity and the database. Its main responsibility is to communicate with the database.

**What are POCO classes in Entity Framework?**

POCO means 'Plain Old CLR Objects'. Still, it doesn't imply the oldness and plainness of these classes. A POCO class doesn't contain any reference to the .NET framework or Entity Framework. The POCO entities are the same as available domain objects in Entity Framework objects.

These classes are not dependent on any framework-specific base class. This is unlike the general .NET class. The entity object-derived LINQ queries are supported by POCO. And the POCO entities are supported by both EF core and EF 6.



**Mention the various types of loading that are available for loading related entities in EF.**

The given loading types are offered by Entity Framework:

1. Lazy loading
2. Eager loading
3. Explicit loading

**Explain the eager loading, lazy loading, and explicit loading in detail.**

**Lazy loading:** The loading process of related objects is delayed till the time we need them. Only the objects we need are returned in the lazy loading. At the same time, the other related objects get returned only when we need them.  
**Eager loading:** in this loading, all the related objects are also returned along with the object's query. All the related objects get loaded automatically along with the parent object. We can achieve eager loading in EF 6 by using the Include method.  
**Explicit loading:** This occurs only when we desire lazy loading. The relevant load method should be explicitly called for processing explicit loading. We can achieve explicit loading in EF 6 by using the load method.

**What are different types of inheritance supported by the Entity Framework?**

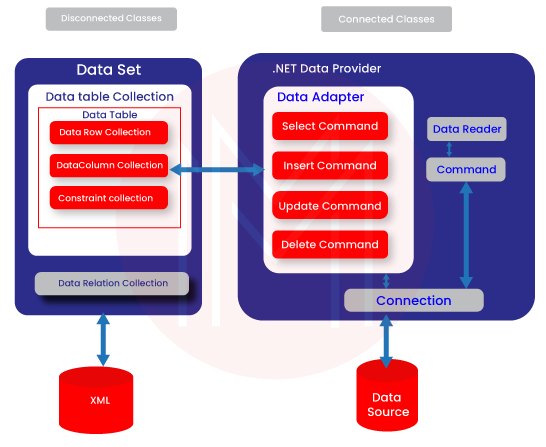
The following three are primary types of inheritance in Entity Framework:

* Table per hierarchy (TPH)
* Table per type (TPT)
* Table per concrete class (TPC)

**Explain the data access architecture available in Entity Framework.**

ADO.NET supports the given two types of data access architecture:

* **Connected data access:** We can access the linked data by the data provider's data reader object. And we can collect the data quickly, but we can't edit it.
* **Disconnected data access:** The data adapter object enables disconnected data access. Datasets work independently of databases, and we can edit data also.



**Which methods can be followed to execute plain SQL in the Entity Framework?**

We can follow the given three methods to execute plain SQL in Entity Framework:

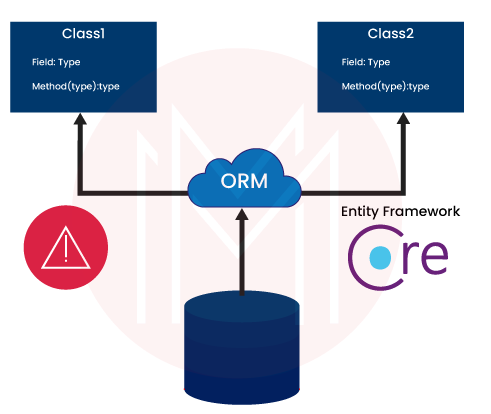
1. SqlQuery()
2. Database.SqlQuery()
3. Database.ExecuteSqlCommand()

**Explain the role of singularizing and pluralizing in the Entity Framework.**

The main responsibility of singularizing and pluralizing in Entity Framework is assigning the objects meaningful naming conventions. This feature can be accessed through the .edmx file. The coding conventions will be assigned to the singular and plural while this feature is used. The convention names have an extra 's' if there are two or more records within an object.

**What is meant by micro O/RMs in Entity Framework?**

A micro ORM is not made to create database schemas, track changes, modify database schemas, etc. Instead, its primary function is to work with database tables. Because the Entity Framework Core and Entity Framework 6 have a complete set of functionalities and features, they are referred to as O/RMs.



**Name some O/RMs that you can use with applications based on .NET.**

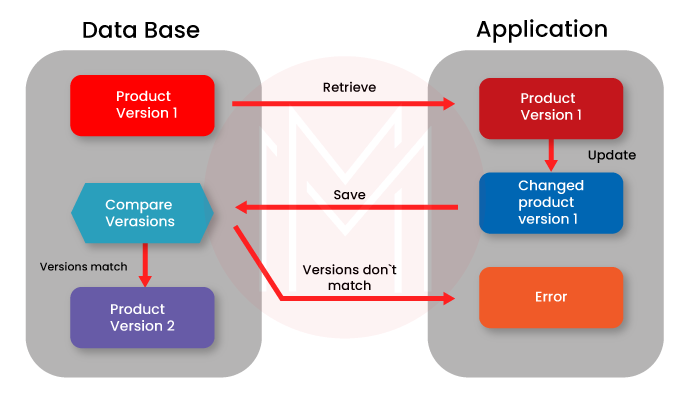
Some of the O/RMs that we can use with applications based on .NET are listed below:

1. Entity Framework Core
2. Entity Framework 6
3. Dapper
4. N hibernate

**What is meant by Optimistic Locking?**

Optimistic Locking is the process of reading a record and noting a version number, timestamp, date, or hash. And then, we check that the record hasn't changed before writing the code back. While writing the record back, we filter the update on the version, ensuring that it's atomic. And then the version is updated in a single hit.

We can abort the transaction and then re-start it if the record is found dirty.



**What is meant by SQL injection attack?**

The hackers use this method of SQL injection to access sensitive information from the databases of organizations. This happens because of improper coding in the applications. This enables the hackers to inject SQL statements into our SQL code.

The main reason behind SQL injection is that the SQL statements are allowed by the user input fields to pass through and query the database directly.

**Which namespace is used for the inclusion of the .NET data provider for the SQL server in your .NET code?**

We use namespace- System.Data.SqlClient for the inclusion of a .NET data provider for SQL server in our .NET code.

**Frequently Asked Interview Questions on Entity Framework:**

**Name some companies that use Entity Framework in their tech stacks.**

Some of the companies that use Entity Framework in their tech stack are given below:

* RoofStacks
* PFB
* Mews
* Insoft
* Tray Corp.
* product development
* Evodeck Digital
* OPLOG Dev Tech.

**List some popular tools with which the Entity Framework is integrated.**

PostgreSQL, MySQL, SQLite, Microsoft SQL Server, .NET, Oracle, Firebird, and IBM DB2 are some of the popular tools with which Entity Framework is integrated.

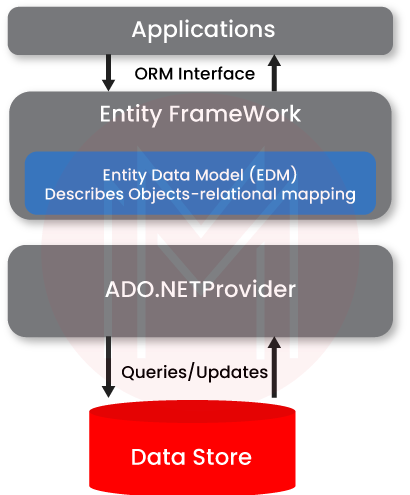
**List some alternatives to Entity Framework.**

Some of the alternatives to Entity Framework are listed below:

* Hibernate
* NHibernate
* Entity Framework Core
* SQLAlchemy
* Sequelize

**Differentiate between Entity Framework and MVC.**

MVC is a framework that focuses primarily on the way of delivering a webpage to the client from the server. While the Entity Framework enables you to access various abstract kinds of databases (MySQL, MSSQL, etc.) and query objects. It is an object-relational mapper, and it doesn't have SQL strings in the projects.



**Write down the advantages of using the Entity Framework?**

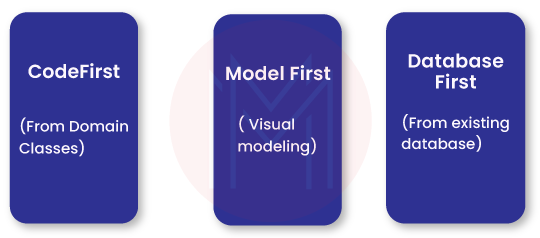
The advantages of Entity Framework are:

1. We can write object-oriented codes with excellent prototypes
2. The developer's job gets simplified as the alternate commands reduce the code length significantly.
3. Creating and modifying a database becomes simple because of auto-migration.
4. All the object queries have a unique syntax (Yoda/LINQ) irrespective of them being the databases.
5. The auto-generated code reduces development costs as well as development time.
6. The drag and drop tables help in mapping the business objects easily.
7. Multiple conceptual models can be mapped to a single storage scheme.

**What are three types of approaches for modeling entries in Entity Framework?**

The following three types of Entity Framework approaches are there:

1. Code First
2. Model First
3. Database First



**When should the different approaches be used?**

The Code First approach suits best when we have the domain classes already with us. On the other hand, the Database First approach is best when we have a database. And the Model First approach is used when we don't have a database or the model classes. The Model First approach provides us with the visual entity tool for designing.

**Why should the Entity Framework be used?**

It becomes tough to write and manage ADO.NET codes at times. And the Microsoft developed Entity Framework solves this problem of managing the tedious tasks. Entity Framework provides domain-specific objects as relational data. Hence, the code-based tasks get reduced significantly.

**Differentiate between Entity Framework and ADO.NET.**

**Entity Framework-** It is slower than ADO.NET. No data layer codes are created by it. The data access layer codes, intermediate codes, and mapping codes are automatically created by it. This is the reason why they work and time of the developers get cut short.

**ADO.NET-** It is relatively faster. Several data layer codes are created by it. Codes are not created for the intermediate layers, data access layers, and mapping codes by ADO.NET.

**Mention the primary functions of the Entity Framework.**

The primary functions of Entity Framework are given below:

* It enables the mapping of domain classes to the database schema.
* Alterations in the entities are kept on track.
* It enables the execution of LINQ queries to SQL.
* The changes stats are stored in the database.

**What is meant by Entity Framework?**

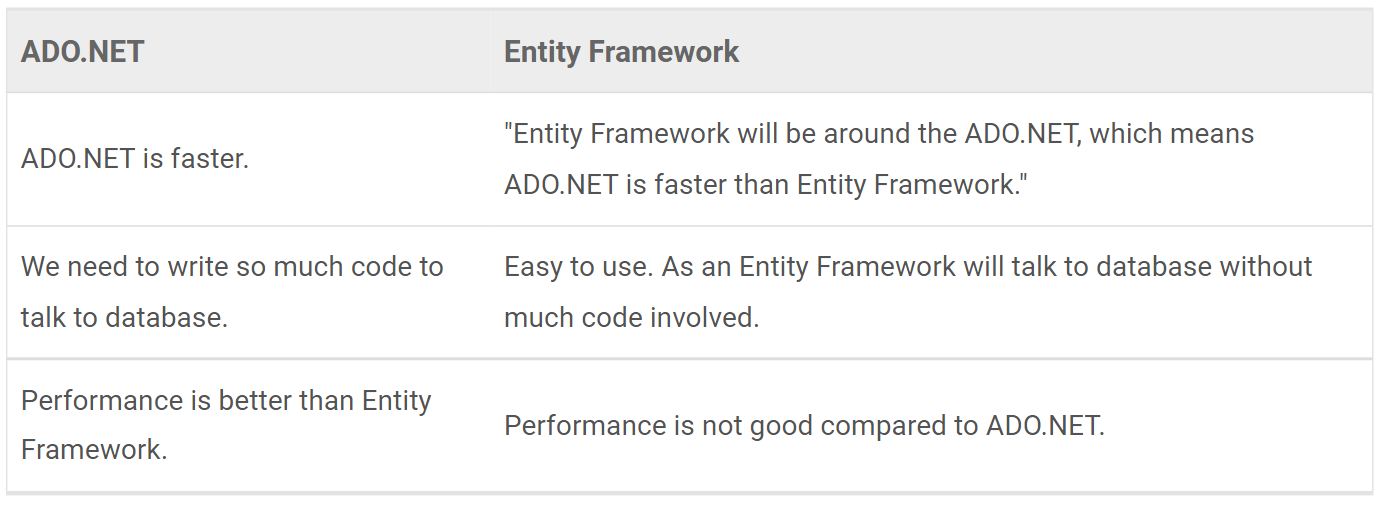
**Ans:**ADO.NET Entity Framework is an ORM (object-relational mapping) that extends ADO.NET components to produce a higher abstract object model. Instead of working with datasets, datatables, commands, and connections as illustrated in the code below, you interact with higher-level domain objects such as customers, suppliers, and so on.

**What are the features of Entity Framework?**

 Entity Framework is an ORM framework for.NET offer features like data modeling using classes, LINQ integration for querying, support for different database providers, automatic code generation, change tracking for simple persistence, and choices for lazy or eager loading of related data.

**What is the difference between ADO.NET and Entity Framework?**

**Ans:**The following are the differences between ADO.NET and Entity Framework:



**What is Code First approach in Entity Framework?**

**Ans:**We avoid using Entity Framework's Visual Designer when using the Code First technique. To put it another way, the EDMX file is not included in the solution. As a result, you now have complete control over both the context and entity classes.

**What is a Conceptual Model?**

**Ans:**The model classes that include the relationships are known as conceptual models. These aren't affected by the database's design.

**What is the primary benefit of using Entity Framework or EF?**

**Ans:**The main advantage of using Entity Framework (EF) is that it automatically generates code for the Model (Middle Layer), Mapping code, and Data Access Layer. It saves a significant amount of time during the development process.

**What are the disadvantages of Entity Framework?**

**Ans:**There are certain disadvantages to Entity Framework, including as performance overhead because of its abstraction layer, the potential for created SQL inefficiencies, a lack of control over complex queries, and a learning curve for advanced functionality. If not used carefully, it could also result in ineffective database architectures.

**How can you use Entity Framework in MVC to retrieve data from a database?**

**Ans:**The following are the steps to retrieve data from a database in Model View Controller using Entity Framework:

* Create a new project
* Add the Entity Framework reference to the NuGet package manager
* Within the table structure, create a new class in the model
* Add the connection string to the name of the web.config.connection string so that it matches the context
* Open Global.asax.cs, add the Entity Framework extra namespace and then initialize the database
* Right-click the Controller folder, then go to the Namespace section and add a new controller and model reference
* Add the sections you want to see by right-clicking on the controller's name

**What is a Storage Model?**

**Ans:**Our database design models are called Storage Models, and they include database tables, views, stored procedures, and keys with relationships.

**What are Micro ORMs?**

**Ans:** Instead of generating, altering, or tracking changes, a Micro ORM is designed to focus on the most crucial task of working with database tables. Both EF 6.x and EF Core are O/RMs since they offer comprehensive functionalities.

**What additional O/RMs are there that you can utilize with.NET applications?**

**Ans:**You can utilize the following O/RMs with.NET applications:

* Dapper
* Entity Framework Core 6.x
* Entity Framework 6.x
* N Hibernate

**Define Dapper.**

**Ans:**For the.NET world, Dapper is a simple/micro ORM. The StackOverflow team designed Dapper to address their difficulties and made it open source. It's a NuGet library for database operations that can be included in any .NET project.

**In Entity Framework, what is migration?**

**Ans:**When your model changes, Entity Framework introduced a migration tool that adjusts the database schema without losing any existing data or database objects.

There are two kinds of Migration:

* Code-based Migration
* Automated Migration

**What is a DBContext class and what does it do?**

**Ans:** DBContext represents the database connection and a set of tables, while DBSet represents the tables themselves. The DBContext lets you use a connection string to connect your model properties to your database (probably using Entity Framework).

Later on, you'll use the DBContext to refer to a database in your controller to handle data.

**What are the benefits of the Model First Approach?**

**Ans:**Model first approach provides the flexibility to design Entity Models independently and allows for later stage improvement.

Model classes can be created by drawing them in the edmx designer, so there is no need for a large database.

**What exactly is in an .edmx file?**

**Ans:**An XML file with the extension .edmx declares a conceptual model, a storage model, and the mapping between these models. This file also contains the data that ADO.NET entity data model designer uses to render a model graphically. It contains all of the mapping details for how an object maps to SQL tables. It is classified into three types: SSDL, CSDL, and MSL.

**Explain the significance of the T4 entity in the Entity Framework.**

**Ans:**T4 entities are critical in Entity Framework because they are at the heart of entity framework code generation. It reads the EDMX XML file and generates C# code behind the scenes.

**How can you load related entities in EF (Entity Framework)?**

**Ans:**There are three ways to load related entities or data into EF.

* Loading with zeal
* Explicit Loading
* Lazy Loading

**What are the advantages and disadvantages of the Database First Approach?**

**Ans:**The advantages and disadvantages of Database First Approach are following:

**1. Advantages:**

If an existing database exists, it is simple to create entity models.

For data-intensive applications, this is the preferred method.

**2. Disadvantages:**

A massive amount of code is generated when we create a .edmx file from an existing database.

We must extend the models if we want to add additional functionality to the models generated.

**How does Entity Framework handle concurrency?**

**Ans:**Concurrency is handled in EF by using optimistic locking. To use optimistic locking, right-click on the EDMX designer and select Fixed as the concurrency mode. You should now receive an OptimisticConcurrencyException error whenever we have a concurrency problem**.**

**What are Entity Framework POCO classes?**

**Ans:**POCO is an abbreviation for Plain Old C# Object. EDMX generated classes that are cluttered with entity tags. We frequently want to use simple.NET classes and integrate them with the Entity Framework.

**What are the Entity Framework Architecture components?**

**Ans:**The following are the Entity Framework components:

* Entity SQL
* Object Service
* Entity Data Model (EDM)
* LINQ to Entities
* ADO.Net
* Data Provider
* Entity Client Data Provider

**What is an Entity Data Model?**

**Ans:**An entity data model (EDM), which is independent of the underlying database schema, is a conceptual representation of the data in a database. EDM is represented in Entity Framework by a collection of classes that provide the organization of entities (objects) and connections between them.

It acts as a connection point between the application and the database, enabling programmers to handle data in an object-oriented way. Entities, affiliations, and mappings between entities and the database tables are all included in this model.

**Explain the different parts of the Entity Data Model?**

**Ans:**There are three primary components to the entity data model (EDM):

* **Conceptual Model:** It illustrates how the application views the data conceptually. It describes entities (things), their attributes, and their connections with one another. In an object-oriented programming language like C# or VB.NET, it is often specified via classes
* **Storage Model:**It shows how the underlying database is organized. It consists of tables, views, keys, and other components unique to databases. By using the database schema, it is represented
* **Mapping:**These show how the conceptual model and the storage model relate to one another. It details the mapping of entities' properties to database columns and tables. This step makes sure that data is correctly stored in the database and retrieved when needed

**What is LINQ to entities?**

**Ans:**LINQ to Entities is query language which we use to write queries against the object models and the resultant query will return the entities defined in the Conceptual Model.

**What is the EF Data Access Architecture?**

**Ans:**The ADO.NET Framework supports two Data Access Architecture models:

* Interested in making connections
* Disconnected

EF employs the Disconnected model (no connections are left open). Because you work with data and make desired changes before performing the SaveChanges. A context in an N-tier application either creates or saves entities, but not both. It generates entities that are serialized and dispose of the context. Entities that return from the client application are deserialized and reattached to a new context instance, where their changes are saved.

**What is the DataAdapter class in ADO.NET?**

**Ans:**The ADO.NET data-adapter class retrieves data from a database, stores it in a dataset, and then return changes to the database. Data-adapter serves as an intermediary for all types of communication. Data-adapter fills data into a Data-table using the Fill() method.

**What is the difference between connected and disconnected data access in ADO.NET?**

**Ans:**Connected data access: You can have connected data access by using the DataReader objects of the data provider. It allows for quick access to data but does not allow for editing.

Disconnected data access: Disconnected data access is accomplished using the DataAdapter object. The dataset is editable and works independently of the database.

**What exactly is Optimistic Locking?**

**Ans:** Optimistic locking is a strategy that involves reading a record, noting a version number (other methods include dates, timestamps, or checksums/hashes), and ensuring that the version hasn't changed before writing the record back. When you write the record back, you filter the version update to ensure it's atomic. (that is, it hasn't been updated between the time you check the version and write the record to the disc) and update the version in a single hit.

If the record is dirty (i.e., a different version than yours), the transaction is aborted and the user can restart it. This strategy is best suited for high-volume systems and three-tier architectures where a connection is not always maintained.

**What are the key functionalities offered by the Entity Framework?**

**Ans:** The key functionalities offered by the Entity framework are Object-relational mapping, LINQ integration, change tracking, CRUD operations, Lazy loading, transactions, and database-first approaches.

**Discover the various types of inheritance that Entity Framework supports.**

**Ans:** There are three types of inheritance:

* **Table Par Hierarchy(TPH):** In this, all classes in an inheritance hierarchy are mapped to a single table. It puts all the fields for each type into a single table. After that, it uses the discriminator field to differentiate the types
* **Table Per Type(TPT):** It is similar to the TPH, but in this, each class in the inheritance has its own table
* **Table Per Concrete Class:**It creates a separate table for each concrete class and stores its fields in that separate table

**Explain the concept of database concurrency and discover effective strategies for managing it.**

**Ans:** Database concurrency is the ability to handle multiple concurrent transactions and it modify the data simultaneously. Below are some strategies:

* **Optimistic concurrency:** It assumes that conflicts are rare and each transaction reads the data without locking it. If no conflicts occur, then the changes are committed. Otherwise, it aborts the transaction
* **Pessimistic concurrency:** It assumes that conflicts are likely between concurrent transactions. This method ensures access to the data but can lead to the degradation of performance
* **Row versioning:** This strategy includes that each row in the database has a version number, and it gets updated when a transaction modifies a row

**Explain the ways to increase performance of Entity Framework?**

Ans: Make use of eager loading, compiled queries, and stay away from the N+1 query problem to improve Entity Framework speed. When performing bulk updates, disable auto-detect changes and use AsNoTracking() for read-only data. Use batch updates for efficiency and stored processes for complex operations; avoid loading unneeded data. Design and index your database properly to maximize it. Reduce the number of lengthy transactions, and think about utilizing AsQueryable() with LINQ-to-Objects.

**What is meant by the Entity SQL?**

**Ans:** Entity SQL is a query language that allows developers to query conceptual models in the entity framework. Conceptual models represent the data as relationships, and Entity SQL allows users to query those relationships.

**What is meant by the SQL injection attack?**

**Ans:** SQL injection is a type of security attack. It inserts malicious SQL code into user inputs or backend databases, which can manipulate the application’s SQL statements and potentially gain unauthorized access. This can lead to data breaches and unauthorized data exposure.

#### **What is Entity Framework Core?**

Entity Framework Core (EF Core) is an open-source, lightweight, extensible, and cross-platform version of Entity Framework, which is Microsoft’s object-relational mapper (ORM) for .NET. It enables .NET developers to work with a database using .NET objects, eliminating the need for most of the data-access code that developers usually need to write.

#### **How does EF Core work?**

EF Core works by mapping the classes in your application to database tables and the instances of those classes (objects) to rows in those tables. It tracks changes to objects in your application and applies them to the database when you call the SaveChanges() method. It also automatically generates SQL queries for reading and writing data in the database.

#### **What are the main components of EF Core?**

The main components of EF Core include:

* **DbContext:** A way to configure and use EF Core, acting as a bridge between your domain or entity classes and the database.
* **DbSet:** Represents a collection of entities of a specific type that you can query and save.
* **Model:** Represents the mapping between your .NET classes and the database schema.
* **Querying:** Allows data to be retrieved from the database using LINQ.
* **Change Tracking:** Keeps track of changes made to the entities so that it can update the database accordingly.
* **Migrations:** Provides a way to update the database schema to match the data model of your application.

#### **What are Migrations in EF Core, and why are they useful?**

Migrations in EF Core are a way to update the database schema to match the current state of the application’s data model. They are useful because they allow developers to apply incremental updates to the database schema without losing data or manually writing SQL scripts. EF Core can automatically generate migrations and can be applied programmatically or via command-line tools.

#### **Can you explain the Code-First approach in EF Core?**

The Code-First approach in EF Core allows developers to define their database schema using C# classes. The framework then generates the database and tables based on these classes. This approach is useful for developers who prefer to work within the code rather than dealing with database design tools. It also allows for version control of the database schema through migrations.

#### **What is the difference between Entity Framework and EF Core?**

Entity Framework (EF) is the original ORM for .NET, which is designed to work with .NET Framework. EF Core is a complete rewrite of Entity Framework designed to work with .NET Core, making it cross-platform, lightweight, and more performant. EF Core also introduces new features and improvements over EF, such as shadow properties, global query filters, and better support for asynchronous operations.

#### **What are shadow properties in EF Core?**

Shadow properties are properties that are not defined in your .NET entity class but are defined in the EF Core model. They are useful for storing data that is not part of the domain model but is needed for database relationships or other EF Core functionalities. Shadow properties can be accessed and modified using the EF Core API but do not appear in the class code.

#### **How does EF Core handle transactions?**

EF Core supports transactions, allowing multiple database operations to be executed as one atomic operation. By default, EF Core automatically wraps a series of operations performed during a single call to SaveChanges() in a transaction. Developers can also manually begin, commit, or roll back transactions using the Database.BeginTransaction(), Commit(), and Rollback() methods, respectively.

#### **What is lazy loading, and how is it implemented in EF Core?**

Lazy loading is a pattern where the loading of related data is delayed until it is specifically requested. In EF Core, lazy loading can be implemented by installing Microsoft.EntityFrameworkCore.Proxies package and enabling it in the DbContext configuration. This involves creating virtual navigation properties in your entity classes, which EF Core will then override to add lazy loading capabilities.

#### **How can you improve the performance of applications using EF Core?**

To improve the performance of applications using EF Core, consider the following strategies:

* Use eager loading, explicit loading, or lazy loading judiciously to load related data efficiently.
* Minimize the use of tracking for entities when not needed by using .AsNoTracking().
* Optimize query performance by selecting only the needed columns and avoiding N+1 queries.
* Use batching for large updates or inserts to reduce database round-trips.
* Consider using raw SQL queries or stored procedures for complex queries or operations for better performance.

#### **How do you define relationships in EF Core?**

In EF Core, relationships between entities (tables) can be defined using navigation properties and the Fluent API or data annotations in your entity classes. There are three main types of relationships: one-to-one, one-to-many, and many-to-many.

* **One-to-One:** A primary key in one entity is also a foreign key in another entity. This can be configured using the HasOne and WithOne methods in the Fluent API.
* **One-to-Many:** A single entity on one side is related to multiple entities on the other side. This is represented by a collection navigation property on the one side and a foreign key on the many side. The Fluent API is configured using HasMany and WithOne.
* **Many-to-Many:** Entities on both sides can relate to multiple entities on the other side. EF Core handles many-to-many relationships by implicitly creating a join table (though you can explicitly define it if needed).

#### **What is a DbContext?**

A DbContext in EF Core represents a session with the database, allowing you to query and save instances of your entities. It is a combination of the Unit Of Work and Repository patterns and provides APIs to perform CRUD operations, manage transactions, and track changes to objects. The DbContext manages the database connections and is configured with connection strings and other database-related settings.

#### **What are Entities in EF Core?**

Entities in EF Core represent the data that you want to map to the database tables. They are CLR (Common Language Runtime) classes that are mapped to database tables. An entity includes properties that map to the columns of a database table. EF Core tracks changes made to these entities and applies these changes to the database based on the type of operations performed (Create, Read, Update, Delete).

#### **How do you handle database connections in EF Core?**

Database connections in EF Core are managed by the DbContext class. You typically specify the connection string and other database provider options in the OnConfiguring method of your DbContext class or in the AddDbContext method in the Startup.cs (or a similar configuration class) of your application. EF Core opens and closes connections to the database automatically when performing operations like querying or saving data. However, you can manually control the connection using the DbContext.Database.GetDbConnection() method if you need to perform custom database operations.

#### **What is LINQ, and how does it integrate with EF Core?**

LINQ (Language Integrated Query) is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages using a syntax reminiscent of SQL but integrated into the programming language. In EF Core, LINQ queries are used to interact with the database in a strongly typed manner. You can write LINQ queries against the DbSet<TEntity> properties in your DbContext, and EF Core translates these LINQ queries into the corresponding SQL queries for the database.

EF Core’s integration with LINQ allows you to write complex queries with conditions, joins, grouping, and ordering directly in C# (or another .NET language), abstracting away the specifics of the underlying database. This integration also provides the benefits of compile-time syntax checking and IntelliSense support in IDEs like Visual Studio.

#### **How do you use Fluent API in EF Core?**

The Fluent API in EF Core is used to configure the database schema in a detailed and fluent manner, allowing for configuration beyond what data annotations can achieve. It is used inside the OnModelCreating method of your DbContext class. The Fluent API provides methods to configure aspects of your model, such as

### ****Intermediate EF Core Interview Questions and Answers****

#### **What is the purpose of the DbContext class in EF Core?**

The DbContext class in EF Core serves as a bridge between your domain or entity classes and the database. It is responsible for managing the database connections, tracking changes to objects, querying the database, and saving changes back to the database.

#### **Explain the role of DbSet in EF Core.**

DbSet represents a collection of entities of a specific type that you can query, add, update, or remove. It provides the methods necessary for these operations and is typically used as a property within your DbContext class to access the entities of a given type.

#### **How do you configure a one-to-many relationship using Fluent API in EF Core?**

A one-to-many relationship can be configured using the Fluent API using the HasMany() and WithOne() methods. For example, if you have a Blog entity that can have many Post entities, you would configure it like this:

modelBuilder.Entity**<**Blog**>()**

.HasMany**(**b =**>** b.Posts**)**

.WithOne**(**p =**>** p.Blog**)**

.HasForeignKey**(**p =**>** p.BlogId**)**;

This configures the relationship where a Blog has many Posts, and each Post is associated with one Blog through the BlogId foreign key.

#### **What is Lazy Loading, and how do you enable it in EF Core?**

Lazy Loading is a pattern where the loading of related data is deferred until it is explicitly requested. In EF Core, lazy loading can be enabled by installing Microsoft.EntityFrameworkCore.Proxies package and enabling it with the UseLazyLoadingProxies() method in the DbContext options builder. Additionally, navigation properties must be declared as virtual to enable them to load lazily.

#### **Describe how to handle transactions in EF Core.**

In EF Core, transactions are used to execute a set of operations as a single unit of work. EF Core supports transactions implicitly through the SaveChanges() method, which wraps the operations in a transaction. For explicit transaction control, you can use the Database.BeginTransaction(), commit the transaction with Commit(), or roll it back with Rollback() methods.

#### **Explain the concept of Shadow Properties in EF Core.**

Shadow Properties are fields in the database that do not have corresponding properties in the entity class. They are useful for storing data that should not be exposed through the entity’s API, such as audit information (e.g., CreatedAt, CreatedBy). Shadow properties are configured using the Fluent API.

#### **How do you perform a raw SQL query in EF Core?**

In EF Core, you can perform a raw SQL query using the FromSqlRaw method to query data. For example:

**var blogs = context.Blogs.FromSqlRaw(“SELECT \* FROM Blogs”).ToList();**

To execute commands that do not return entities, you can use the ExecuteSqlRaw method.

#### **What are Entity States in EF Core? Describe them.**

Entity States in EF Core represent the state of an entity with respect to its tracking by the DbContext. The possible states are:

* **Added**: The entity is new and should be inserted into the database on SaveChanges().
* **Unchanged**: The entity has not been modified since it was retrieved from the database.
* **Modified**: The entity has been modified, and the changes should be saved in the database.
* **Deleted**: The entity has been marked for deletion from the database.
* **Detached**: The entity is not being tracked by the DbContext.

#### **How can you improve query performance in EF Core?**

Improving query performance in EF Core can be achieved by:

* Using AsNoTracking() for read-only queries to reduce overhead.
* Selecting only the necessary columns instead of retrieving entire entities.
* Avoid N+1 queries by eagerly loading related data when necessary.
* Using raw SQL queries for complex queries for better efficiency.
* Enabling query caching where appropriate.

#### **What is the difference between Add(), Attach(), and Update() methods in EF Core?**

* Add(): Marks an entity as new. EF Core will insert it into the database on SaveChanges().
* Attach(): Attaches an entity to the context without marking it as Added or Modified. This is useful for scenarios where you want to mark specific properties as modified.
* Update(): Marks an entity and all its properties as Modified. EF Core will update the existing record in the database on SaveChanges(), regardless of whether the entity was previously tracked.

#### **Explain the concept of lazy loading in EF Core.**

Lazy loading is a pattern used to delay the loading of related data until it is specifically requested. In EF Core, this means that when you query for an entity, its related entities (e.g., navigation properties) are not loaded from the database immediately. Instead, they are loaded the first time they are accessed in code. To enable lazy loading in EF Core, you must install a specific NuGet package (e.g., Microsoft.EntityFrameworkCore.Proxies) and enable it in your DbContext with UseLazyLoadingProxies() method. Lazy loading can improve the performance of the initial query but can lead to the N+1 query problem if not used carefully.

#### **What is eager loading, and how is it different from lazy loading?**

Eager loading is a pattern where related data is loaded together with the main entity in a single query. In EF Core, this is achieved using the Include() method to specify related entities to be loaded. Eager loading differs from lazy loading in that it proactively fetches related data upfront, avoiding the N+1 query problem but potentially fetching more data than is necessary, which can impact performance.

#### **How do you track changes in EF Core?**

EF Core tracks changes to entities retrieved from the database through its change-tracking mechanism. When you query the database for an entity, EF Core keeps a snapshot of its original values. As you modify the entity’s properties, EF Core compares the current values to the original snapshot to determine what has changed when SaveChanges() is called. This mechanism enables EF Core to generate the appropriate SQL statements to update the database based on the changes made to the entity.

#### **Discuss the AsNoTracking() method.**

The AsNoTracking() method is used to tell EF Core not to track any changes for the returned entities. When entities are queried with AsNoTracking(), EF Core returns the entities without keeping a snapshot of their original values, which means it cannot track changes or save these entities back to the database unless they are attached to a tracking context. Using AsNoTracking() can significantly improve performance for read-only operations because it reduces the overhead associated with change tracking.

#### **How do you perform CRUD operations in EF Core?**

CRUD operations in EF Core are performed using instances of your DbContext and its DbSet properties:

* **Create:** Add new entities to the context using the Add(), AddRange(), or similar methods, and then save these changes to the database with SaveChanges().
* **Read:** Use LINQ queries against DbSet properties to retrieve entities from the database.
* **Update:** Retrieve an entity, modify its properties, and use SaveChanges() to apply the changes to the database. EF Core’s change tracking automatically detects changes.
* **Delete:** Remove entities from the context using the Remove(), RemoveRange(), or similar methods, and then use SaveChanges() to reflect the changes in the database.

#### **Explain the role of DbSet in EF Core.**

DbSet<T> represents a collection of entities of a specific type that EF Core manages. It provides the main entry point for querying and interacting with entities of a given type in the database. Through DbSet, you can perform operations like adding, removing, or querying entities. Each DbSet corresponds to a table or view in the database, and the operations performed on the DbSet are translated into SQL commands that EF Core executes against the database.

#### **What is a composite key, and how can you configure it in EF Core?**

A composite key is a key that consists of two or more columns that together uniquely identify a row in a table. In EF Core, you can configure a composite key using the Fluent API by overriding the OnModelCreating method in your DbContext class. Within this method, use the HasKey() method on the EntityTypeBuilder object to specify the properties that make up the composite key.

#### **How do you use Fluent API in EF Core?**

The Fluent API in EF Core is used to configure the model in your DbContext class. It provides a detailed, code-based method to configure your EF Core models, allowing for configurations that are not possible with data annotations alone. Here’s how you can use Fluent API in EF Core:

* Override the OnModelCreating Method: The Fluent API configurations are placed inside the OnModelCreating method in your DbContext class. This method is called by the framework when your model is being created.
* Configure Entity Properties and Relationships: You use the modelBuilder object provided as a parameter to the OnModelCreating method to configure entities, their properties, relationships, indexes, and more.

#### **Discuss concurrency control in EF Core.**

Concurrency control in EF Core is about ensuring that two or more database operations do not conflict with each other, particularly in scenarios where multiple processes or users attempt to modify the same data simultaneously. EF Core handles concurrency by default through optimistic concurrency control. Here’s how it works and how you can implement it:

* **Optimistic Concurrency Control:** This approach assumes that conflicts are rare and doesn’t lock the data; instead, it checks for conflicts before completing the transaction. If a conflict is detected, an exception is thrown, and the application must resolve the conflict.
* **Concurrency Tokens:** EF Core uses concurrency tokens to implement optimistic concurrency control. A property in your entity class can be configured as a concurrency token, which EF Core will check before performing an update or delete operation. If the token value has changed since the entity was fetched, EF Core will abort the transaction because another operation has modified the data.

### ****Advanced EF Core Interview Questions and Answers****

#### **Explain TPH, TPT, and TPC inheritance in EF Core.**

* **TPH (Table Per Hierarchy):** This strategy involves using a single database table to store the data for all types in an inheritance hierarchy. Discriminator columns are used to differentiate between the types. This approach is typically more performant for querying because it avoids joins but can lead to sparse tables if there are many nullable properties only relevant to certain subclasses.
* **TPT (Table Per Type):** In this strategy, each class in the inheritance hierarchy is mapped to its own database table. These tables are related through foreign keys. TPT can lead to more normalized database schemas but may suffer from performance issues due to the required joins when querying across types.
* **TPC (Table Per Concrete class):** Each concrete class in the hierarchy is mapped to its own table, and these tables are not related to the database. Each table contains columns for all of the class’s properties, including those inherited. This approach can eliminate the need for joins and discriminator columns but can lead to data duplication across tables.

#### **How do you handle transactions in EF Core?**

In EF Core, transactions are used to ensure data integrity by grouping multiple operations into a single atomic operation. You can manage transactions manually using the DbContext.Database.BeginTransaction() method, or you can take advantage of EF Core’s automatic transaction management by executing operations within a SaveChanges() or SaveChangesAsync() call. Additionally, the TransactionScope class can be used to create a transaction that spans across multiple contexts.

#### **What is an intercepting SQL Query Feature?**

The SQL query interception feature in EF Core allows developers to intercept and modify SQL queries before they are executed against the database. This can be useful for logging, auditing, or applying custom logic such as dynamic filtering or data modification. You can implement this feature by overriding the OnModelCreating method in your DbContext and using the DbCommandInterceptor class to define the interception logic.

#### **How can you optimize performance in EF Core?**

Performance in EF Core can be optimized through various strategies:

* **Eager Loading:** Use the Include method to preload related data within the same query to avoid the N+1 query problem.
* **Projection:** Use Select to load only the necessary data fields rather than entire entities.
* **Batching:** Take advantage of EF Core’s ability to batch operations to reduce round trips to the database.
* **AsNoTracking:** Use AsNoTracking for read-only operations to improve query performance.
* **Indexing:** Ensure proper indexing in your database to speed up query execution.
* **Pooling:** Use DbContextPooling to reduce the overhead of instantiating DbContext instances.

#### **Explain how to use stored procedures with EF Core.**

In EF Core, stored procedures can be executed using the DbContext.Database.ExecuteSqlRaw or ExecuteSqlRawAsync methods for scenarios that don’t return entities. For retrieving entities, you can use the FromSqlRaw or FromSqlInterpolated methods on a DbSet to map the results of a stored procedure to entity types. Parameters can be passed to stored procedures safely to avoid SQL injection attacks.

#### **How do you handle database versioning in EF Core migrations?**

EF Core Migrations allow you to version your database schema by capturing changes in your data model. Each migration consists of a set of up (apply migration) and down (revert migration) methods. You can manage migrations using the EF Core tools (e.g., Add-Migration, Update-Database, Remove-Migration) to apply, revert, or script migrations. This approach helps manage schema changes over time, facilitating database version control.

#### **What is the purpose of the EF Core In-Memory Database Provider?**

The EF Core In-Memory Database Provider is intended for testing purposes. It allows you to run unit tests against an in-memory database that mimics the behavior of a real database system without the overhead of actual database operations. This facilitates rapid development and testing cycles by enabling the testing of data access layers without requiring an actual database connection or affecting a live database.

#### **How do you customize code-first conventions in EF Core?**

Customizing code-first conventions in EF Core involves defining your own conventions that EF Core will apply during the model-building process. You can customize conventions by overriding the OnModelCreating method in your DbContext class. Here, you can use the Fluent API to configure aspects of your model, such as property types, relationships, table names, and more, according to your specific requirements. This provides a way to apply custom configuration rules and mappings that go beyond the default conventions.

#### **What is the DbContext Pooling in EF Core, and when should it be used?**

DbContext pooling is a performance optimization feature that reuses instances of DbContext classes instead of creating new ones for each request. This reduces the overhead of initialization and disposal of DbContext instances, improving application performance. It should be used in scenarios where the application handles many short-lived DbContext instances, such as web applications.

#### **How do Global Query Filters work in EF Core? Can you provide an example?**

Global Query Filters are LINQ query predicates applied globally to all queries involving a particular entity type. They are useful for implementing soft delete functionality, multi-tenancy, or filtering data based on some global criteria. For example, you can apply a global query filter only to include active records by default:

**modelBuilder.Entity<MyEntity>().HasQueryFilter(e => e.IsActive);**

This ensures that queries against MyEntity will automatically filter out records where IsActive is false.

#### **Explain the difference between Entity Framework Core’s tracking and no-tracking queries.**

Tracking queries in EF Core keeps track of the changes to the returned entity instances in the context. This is useful when you intend to update the entities. No-tracking queries, on the other hand, do not keep track of changes to the entities, leading to better performance. No-tracking is useful for read-only operations. You can opt into no-tracking with .AsNoTracking() or by configuring the query at the context level.

#### **How do you handle concurrency in EF Core?**

Concurrency in EF Core is handled using concurrency tokens. A property in the entity is marked as a concurrency token using the [ConcurrencyCheck] attribute or by using Fluent API. EF Core uses this token to ensure that the entity has not been modified by another process between the time it was read and the time it was updated or deleted. If a concurrency conflict occurs, EF Core throws a DbUpdateConcurrencyException, which can be handled to implement custom conflict resolution logic.

#### **What are Owned Entities in EF Core, and how are they used?**

Owned entities in EF Core are entities that are conceptually part of another entity. They are defined using the .OwnsOne() or .OwnsMany() methods in the Fluent API. Owned entities are stored in the same table as the owning entity by default, although they can be configured to use a separate table. They are useful for encapsulating complex value types within an entity.

#### **Explain EF Core support for Value Conversions and provide an example.**

Value conversions in EF Core allow values to be converted between the database representation and the entity property type when reading from or writing to the database. This is useful for handling enumeration types, encrypting data, or working with custom types. For example, you can use value conversions to store an enumeration as a string in the database:

modelBuilder.Entity**<**MyEntity**>()**

.Property**(**e =**>** e.MyEnumProperty**)**

.HasConversion**(**

v =**>** v.ToString**()**,

v =**>** **(**MyEnum**)**Enum.Parse**(**typeof**(**MyEnum**)**, v**))**;

EF Core implements optimistic concurrency control using concurrency tokens. A special property (e.g., a row version) in the entity is used as a concurrency token. When an entity is updated or deleted, EF Core includes the concurrency token in the WHERE clause of the SQL command. If the command affects zero rows, it means the entity was modified after it was fetched, indicating a concurrency conflict.

#### **Can you perform raw SQL queries with EF Core? Provide an example.**

Yes, EF Core allows performing raw SQL queries using the FromSqlRaw or ExecuteSqlRaw methods for querying and command execution, respectively. This is useful for executing queries or commands that cannot be expressed with LINQ. For example, to perform a raw SQL query:

**var blogs = context.Blogs.FromSqlRaw(“SELECT \* FROM Blogs WHERE Name = {0}”, blogName).ToList();**

#### **What are Interceptors in EF Core, and how can they be used?**

Interceptors in EF Core allow you to intercept and modify operations such as command execution, transactions, and connection opening. They can be used for logging, auditing, changing command text, or implementing retry logic. You can implement an interceptor by deriving from one of the interceptor interfaces (e.g., IDbCommandInterceptor) and registering it with the DbContext.

### ****Practical and Best Practices EF Core Interview Questions and Answers****

#### **How do you handle model changes in EF Core without losing data?**

To handle model changes without losing data, you use EF Core migrations. Migrations track changes to the model and generate code to update the database schema incrementally. To apply migrations without losing data, you carefully plan model changes, generate migrations using the Add-Migration command, review the migration code for accuracy, and apply the migration to the database using the Update-Database command, ensuring data is preserved or appropriately transformed.

#### **What is the best way to manage database connections in EF Core?**

The best way to manage database connections in EF Core is to let the context handle connections automatically. EF Core opens and closes connections as needed for operations like querying or saving data. For more control, you can manually open and close connections using the DbContext.Database property. However, always ensure connections are properly closed after use, preferably with a using statement or try-finally blocks to avoid connection leaks.

#### **When should you use eager loading, and when should you use explicit loading in EF Core?**

* **Eager Loading:** Use eager loading when you know you will need related data for every entity being queried. It is done using the Include method, which loads related data as part of the initial query. This approach minimizes the number of queries but can result in large data retrieval if not used carefully.
* **Explicit Loading:** Use explicit loading when you need related data conditionally or for specific entities after loading them. This is done using the Load method on the navigation property. It allows finer control over when and how much-related data is loaded.

#### **How can you ensure thread safety when using EF Core?**

To ensure thread safety with EF Core, avoid sharing DbContext instances across threads. Each thread should use its own instance of DbContext. This is crucial because DbContext is not thread-safe, and using it from multiple threads can lead to unpredictable behavior and data corruption. For web applications, this typically means using dependency injection to configure the DbContext with a scoped lifetime, ensuring a new instance per request.

#### **What are the best practices for handling exceptions in EF Core?**

Best practices for handling exceptions in EF Core include:

* Catching specific exceptions like DbUpdateConcurrencyException and DbUpdateException to handle concurrency conflicts and update errors, respectively.
* Using try-catch blocks around database operations to manage exceptions gracefully.
* Logging exceptions for debugging and monitoring purposes.
* Implementing retry logic for transient faults, possibly using the built-in support for it in EF Core.

#### **How can you optimize bulk insert operations in EF Core?**

To optimize bulk insert operations in EF Core:

* Consider using third-party libraries like EF Core.BulkExtensions, which offer optimized bulk operations.
* Reduce change tracking overhead by calling ChangeTracker.AutoDetectChangesEnabled = false before adding entities to the context.
* Batch insert operations should be done in reasonable chunks to avoid overwhelming the database.
* For maximum efficiency, use raw SQL for bulk inserts when dealing with a very large number of records.

#### **What strategies can you employ to manage large models in EF Core?**

Strategies for managing large models in EF Core include:

* Splitting the model into multiple DbContext classes, each handling a subset of the model, to improve maintainability and performance.
* Using bounded contexts to separate different areas of the application logically.
* Implementing a repository and unit of work patterns to abstract and manage data operations, making the data access layer more manageable.

#### **How do you implement soft delete functionality in EF Core?**

Soft delete can be implemented in EF Core by adding a flag (e.g., IsDeleted) to entities to mark them as deleted. You can then use Global Query Filters to filter out entities marked as deleted from all queries automatically. When deleting an entity, instead of removing it, set the flag to true and update the entity.

#### **What are some common performance pitfalls in EF Core, and how can you avoid them?**

Common performance pitfalls in EF Core include:

* Overusing eager loading leads to unnecessarily large data retrieval. Use it judiciously and prefer explicit or lazy loading when appropriate.
* Not using .AsNoTracking() for read-only queries, resulting in unnecessary change tracking overhead.
* Performing bulk updates and inserts without optimization. Use bulk operation techniques or third-party libraries.
* Avoid these pitfalls by profiling and monitoring your application, understanding the data access patterns, and choosing the most efficient data loading and modification strategies.

#### **How do you migrate from EF 6 to EF Core?**

Migrating from EF 6 to EF Core involves:

* Assessing compatibility since EF Core may not support all EF 6 features. Review the EF Core documentation for feature compatibility.
* Updating the data model to align with EF Core conventions and features.
* Rewrite the data access code to use EF Core APIs and adjust for any differences in behavior or functionality.
* Generating a new migration in EF Core to create or update the database schema.
* Testing extensively to ensure the application behaves as expected with EF Core.

#### **Discuss strategies for handling large datasets in EF Core.**

* One strategy is to use pagination, where you fetch data in chunks rather than all at once, using methods like Skip() and Take().
* Another approach is to optimize your queries by using appropriate indexing, including covering indexes, to speed up data retrieval.
* Utilizing asynchronous methods (ToListAsync() instead of ToList()) can also improve performance by allowing parallel execution of database queries.

#### **How do you manage complex queries using EF Core?**

* Break down complex queries into smaller, manageable parts, possibly using method chaining or query composition techniques.
* Utilize raw SQL queries or stored procedures for queries that are too complex to be efficiently expressed using LINQ.
* Consider using database views or materialized views to simplify complex queries and improve performance.

#### **Explain how to implement repository and unit of work patterns with EF Core.**

* Implement a generic repository interface for common CRUD operations, abstracting away EF Core specifics.
* Implement a unit of work class to coordinate multiple repository operations within a single transaction.
* Inject these repositories and unit of work instances into your application services to keep your business logic decoupled from data access concerns.

#### **What are some common pitfalls in EF Core, and how can you avoid them?**

* N+1 query problem: Avoid it by using eager loading (Include()), explicit loading (Load()), or projection (Select()).
* Ignorance of database interactions: Understand how LINQ queries translate into SQL to write efficient queries and optimize performance.
* Failure to handle concurrency: Use appropriate concurrency control mechanisms like optimistic concurrency or row versioning.

#### **How do you use EF Core in a multi-threaded environment?**

* Use a separate instance of DbContext per thread or per logical unit of work.
* Avoid sharing DbContext instances across threads, as they are not thread-safe.
* Use async/await patterns for asynchronous database operations to prevent blocking threads.

#### **What are the best practices for securing data in EF Core?**

* Implement proper authentication and authorization mechanisms in your application to control access to data.
* Use parameterized queries or stored procedures to prevent SQL injection attacks.
* Encrypt sensitive data at rest and during transmission using appropriate encryption algorithms.

#### **How do you handle multi-tenancy in EF Core?**

* Add a TenantId column to your tables to associate each record with a specific tenant.
* Filter queries based on the current tenant, either explicitly in each query or through middleware.
* Use a separate database schema or database per tenant, if necessary, to completely isolate tenant data.

#### **Discuss the use of Value Converters in EF Core.**

* Value converters allow mapping between types in the database and types in the application.
* They can be used to handle scenarios like mapping enum values to strings, storing JSON data in a text column, or converting between different date/time representations.
* Implement custom value converters when the built-in converters don’t meet your requirements.

#### **What tools and extensions are available for EF Core development?**

* Visual Studio provides built-in support for EF Core development, including tools for model creation, migrations, and database management.
* dotnet-ef command-line tool offers commands for managing database migrations, generating database scripts, and more.
* EF Core Power Tools extension for Visual Studio provides additional features such as reverse engineering an existing database into EF Core models.

### ****Under the Hood EF Core Interview Questions and Answers****

#### **How does EF Core generate SQL from LINQ queries?**

EF Core uses a query provider that translates LINQ expressions into SQL queries. The process involves parsing the LINQ expression tree, translating it into a model-specific query expression tree that represents the query in terms of the underlying database model, and then generating the corresponding SQL statement. This process allows EF Core to support a wide range of database systems by abstracting the specifics of SQL generation for each database provider.

#### **What is the Change Tracker in EF Core, and how does it work?**

The Change Tracker in EF Core is responsible for keeping track of all the changes made to the entities since they were loaded from the database. It does this by taking a snapshot of entity states when they are first retrieved and then comparing the current state to this snapshot upon calling SaveChanges(). Based on this comparison, EF Core determines which entities must be inserted, updated, or deleted in the database.

#### **Can you explain the concept of Shadow State in EF Core?**

Shadow State in EF Core refers to properties that are not defined in the entity class itself but are nevertheless part of the model and managed by EF Core. These properties exist only in the EF Core model and are stored in the database, making them useful for fields like foreign keys or audit columns that don’t need to be exposed to the entity class. Shadow properties are accessed and modified using the EF Core API, such as Entry(entity).Property(“ShadowPropertyName”).CurrentValue.

#### **How does EF Core handle database migrations and schema updates?**

EF Core handles database migrations, and schema updates through a feature called Migrations. Developers define changes to the model, and EF Core generates code that can apply these changes to the database schema. Migrations can be added, removed, or modified manually if needed. Migrations can be applied programmatically via the DbContext.Database.Migrate() method or through command-line tools, allowing for controlled schema evolution across different environments.

#### **What mechanisms does EF Core use to Optimize Query Performance?**

EF Core optimizes query performance through several mechanisms, including:

* **Query Caching:** EF Core caches compiled query plans to avoid re-compiling the same query.
* **Batching:** EF Core batches multiple operations into a single database command when possible, reducing round trips to the database.
* **No-Tracking Queries:** For read-only scenarios, queries can be executed without tracking changes to the entities, reducing overhead.
* **Eager Loading:** Allows specifying related data to be loaded upfront, reducing the number of separate queries required.

#### **Explain how EF Core uses both Lazy Loading and Eager Loading.**

EF Core supports both lazy loading and eager loading for loading related data:

* **Lazy Loading:** Related data is loaded on-demand when the navigation property is accessed for the first time. This is achieved by making navigation properties virtual and using proxy entities or interceptors. Lazy loading can lead to the N+1 query problem if not used carefully.
* **Eager Loading:** Related data is loaded upfront in the initial query using the Include method. This approach is efficient when you know you will need the related data for every entity being loaded, as it reduces the number of queries made to the database.

#### **What are compiled queries in EF Core, and when would you use them?**

Compiled queries in EF Core are queries that have been compiled into a delegate for reuse. This feature is useful for queries that are executed frequently with different parameters. By compiling a query once and reusing it, you avoid the cost of query compilation on subsequent executions, improving performance. Compiled queries are defined using the EF.CompileQuery and EF.CompileAsyncQuery methods.

#### **How does EF Core manage transactions across multiple database operations?**

EF Core manages transactions by wrapping multiple database operations in a single transaction scope. By default, EF Core uses a transaction for each SaveChanges() call. For more control, developers can use the BeginTransaction() method to start a transaction explicitly, execute multiple operations, and then commit or roll back the transaction using the Commit() or Rollback() methods, ensuring atomicity of the operations.

#### **Discuss how EF Core supports multiple database providers.**

EF Core supports multiple database providers through an extensible provider model. Each provider implements the interfaces defined by EF Core to translate operations, queries, and updates into database-specific SQL and functionality. This design allows EF Core to interact with different types of databases, such as SQL Server, SQLite, PostgreSQL, and others, by simply changing the provider in the application configuration without altering the application code.

#### **What are the limitations of EF Core?**

* EF Core lacks some features compared to Entity Framework 6, such as no support for lazy loading of related data by default, no built-in support for many-to-many relationships without explicit join entities, and limited support for complex types.
* EF Core may have performance issues with complex queries compared to raw SQL queries or stored procedures.
* Some database-specific features may not be fully supported or optimized in EF Core.

#### **Discuss how EF Core works with Asynchronous Programming.**

* EF Core provides asynchronous versions of its database operations, such as ToListAsync() or SaveChangesAsync().
* Asynchronous programming allows non-blocking execution of database operations, improving the scalability and responsiveness of applications.
* When using asynchronous methods, EF Core utilizes the underlying database provider’s asynchronous APIs to perform I/O operations asynchronously.

#### **What is the Entity Framework Core Query Pipeline?**

* The EF Core query pipeline is the internal mechanism responsible for translating LINQ queries into SQL queries that can be executed against the database.
* It includes components for query composition, translation of LINQ expressions to SQL, materialization of query results, and execution of SQL commands.
* Developers can extend or customize the query pipeline through interceptors or query filters to modify query behavior or add additional processing.

#### **How does EF Core handle caching?**

* EF Core performs caching at various levels to improve performance.
* Query results, entity instances, and metadata are cached to reduce database round-trips and improve application responsiveness.
* EF Core supports query result caching, which allows the caching of query results for a specified duration or until the underlying data changes.
* Developers can control caching behavior through configuration options or explicit caching mechanisms.

#### **Explain how transactions work in EF Core.**

* EF Core supports transactions to ensure data consistency and integrity when performing multiple database operations.
* Transactions can be managed explicitly using TransactionScope or BeginTransaction() methods on the DbContext or DbTransaction objects.
* EF Core automatically enlists ambient transactions when running within a transaction scope, allowing multiple database operations to participate in the same transaction.
* Transactions can be committed or rolled back using Commit() or Rollback() methods, respectively.