Contents

[Entity Framework 1](#_Toc208999605)

# Entity Framework

Question 1: Difference between code-first, database-first, and model-first approaches?

1. **Code-First** → You write **C# classes** first → EF creates database from them.
2. **Database-First** → In **Database First Approach**, the **database is already created** (tables, relationships, keys, etc.). We use Scaffold-DbContext command to generate **models and DbContext** from tables.



1. **Model-First** → You design a **visual model (EDMX)** → EF generates both classes and database. **EF Core does not support Model-First (EDMX designer)** — it only supports **Code-First** and **Database-First**. Model-First was only in **older EF (up to EF6)**.

Question 2: Explain eager loading vs lazy loading vs explicit loading?

1. **Eager Loading** → Related data is loaded **immediately together** with the main data (using Include).It uses joins.
2. **Lazy Loading** → Related data is loaded **only when you access it later** (needs EF Core Proxies and virtual keyword with navigation property). It don’t uses joins.
3. **Explicit Loading** → You **manually tell EF** to load related data (using Load). Not use joins.

(context.Entry(...).Collection(...).Load()).

1. **Which to use?** → In real projects we **prefer Eager Loading**, because it clearly fetches everything in one go and avoids surprises. **Lazy Loading** can cause many hidden queries, so it’s rarely used in big projects. **Explicit Loading** is used only when we want full manual control, but not for everyday cases.

Example- one to many (one student have many course but course will have single student)

A computer screen shot of text

AI-generated content may be incorrect.

Eager Loading (Include) - I want Student + all his Courses immediately in one query.

A black background with white text

AI-generated content may be incorrect. A black background with white text

AI-generated content may be incorrect.

Explicit Loading (Load) First give me Student, later I’ll decide whether to fetch his Courses.

A computer screen shot of a program code

AI-generated content may be incorrect. A computer screen with white text

AI-generated content may be incorrect.

Lazy Loading Don’t fetch related data until I actually try to use the navigation property. EF will **auto-fetch** when you first access student.Courses

To Enable Lazy loading follow steps:

1. Install 
2. Set this in program.cs. UseLazyLoadingProxies();

A computer screen shot of a computer code

AI-generated content may be incorrect.

1. Mark navigation properties as virtual

A computer screen shot of white text

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect. A computer screen shot of a program

AI-generated content may be incorrect.

Question 3: What's the difference between DbContext and ObjectContext?

1. DbContext → DbContext is the main class in Entity Framework Core that connects our C# code with the database. It manages the database connection, tracks changes of entities, and lets us query or save data.
2. ObjectContext → Older, heavier, low-level API (used in EF 4.x/5/6).
3. Today, we mostly use DbContext.

Question 4: How do you handle transactions in EF Core?

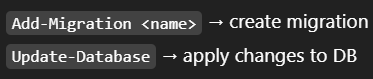
We handle transactions in EF Core using Database.BeginTransaction(). If all operations succeed, we commit; otherwise, we rollback to keep data safe. a transaction makes sure a group of database operations are all done successfully or none at all.

A screen shot of a computer program

AI-generated content may be incorrect.

Question 5: What are migrations in EF Core and how do you use them?

1. Migrations in EF Core are a way to keep the database schema in sync with our model classes. Whenever we change our C# models, migrations update the database without dropping data.
2. Common commands



Question 6: *Explain Change Tracking in EF Core*?

* Change Tracking is a feature in EF Core where the DbContext keeps track of all the changes made to entities (objects) retrieved from the database. It monitors whether an entity is **added, modified, or deleted**, so that when SaveChanges() is called, EF Core knows which SQL statements to generate and execute.
* By default, all entities retrieved via DbContext are **tracked**.
* Change tracking improves efficiency for updates but uses **more memory**.

A screenshot of a computer program

AI-generated content may be incorrect.

Question 7: *What is the difference between AsNoTracking vs Tracking queries?*

* **Tracking**: EF Core watches changes → needed for update/delete → slower for large data.
* **AsNoTracking**: EF Core does not watch → read-only queries → faster, less memory.

A black screen with white text

AI-generated content may be incorrect.

Question 8: *What is Concurrency Handling in EF Core?*

* control ensures that **two users don’t overwrite each other’s changes** when updating the same record. EF Core provides **Optimistic Concurrency**, which assumes conflicts are rare and checks **before saving**.

**How EF Core does it internally**

* EF Core uses a concurrency token (commonly a RowVersion column [Timestamp] / RowVersion).
* When SaveChanges() is called, EF Core checks whether the token in the database matches the token in your entity. EF Core generates a SQL UPDATE statement with a WHERE clause that includes the original RowVersion. And we get token when we fetch entity.
* If **no rows are affected**, EF Core throws DbUpdateConcurrencyException.

A screen shot of a computer

AI-generated content may be incorrect.

**Steps to implement concurrency control in EF Core**

* Add a RowVersion column of rowversion type ssms will auto manage in your table



* Update your entity model:

A computer screen shot of a program code

AI-generated content may be incorrect.

* **Use try-catch to handle conflicts**:
* A screen shot of a computer program

  AI-generated content may be incorrect.

Question 9: *How SQL Server (SSMS) handles concurrency in stored procedures?*

In SQL Server, concurrency can be handled using:

* **Optimistic concurrency** → check before update (like EF Core) – ( eg with RowVersion)

A screenshot of a computer program

AI-generated content may be incorrect. A screenshot of a computer program

AI-generated content may be incorrect.

* **Pessimistic concurrency** → lock the row/table while working. *UPDLOCK + ROWLOCK* → SQL Server locks the row so no one else can update it until the transaction commits.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.