100+ Entity framework

Interview Questions and Answers

100+ Entity Framework Interview Q&A

0-2 Exp

1. What is Entity Framework and why is it used?

Entity Framework (EF) is an ORM (Object-Relational Mapper) for .NET that enables developers to work with a database using .NET objects, eliminating the need for most of the data-access code. EF maps database tables to classes and columns to properties, allowing CRUD operations using LINQ.

Code Example:

```
var students = context.Students.ToList();
```

2. What are the different approaches in Entity Framework?

EF supports three development approaches: Database First (model from existing DB), Code First (model from classes), and Model First (model from visual designer). Code First is popular for its flexibility in coding and migrations.

Code Example:

```
public class Student { public int Id { get; set; } public string Name { get; set; } }
```

3. What is the difference between Code First, Database First, and Model First?

- Code First: Define models in code, generate DB from code.
- Database First: Generate models from existing DB.
- Model First: Create model diagram visually, generate code & DB.

  Code Example (Code First):

```
public class AppDbContext : DbContext { public DbSet<Student> Students { get; set; } }
```

4. How do you define a DbContext?

DbContext represents a session with the database. It allows querying and saving data. You define entity sets using DbSet<T>.

Code Example:

```
public class SchoolContext : DbContext { public DbSet<Student> Students { get; set; } }
```

5. What is a DbSet in Entity Framework?

DbSet<T> represents a table in the database and allows querying and saving instances of the entity. It's declared in the DbContext.

Code Example:

```
public DbSet<Student> Students { get; set; }
```

6. How do you perform CRUD operations using EF?

Use Add, Find, Update, Remove, and SaveChanges methods. EF tracks changes and applies them to the database.

Code Example:

```
context.Students.Add(new Student { Name = "John" }); context.SaveChanges();
```

7. What is Lazy Loading in EF?

Lazy Loading delays loading related data until it's explicitly accessed. It requires navigation properties to be virtual.

Code Example:

```
var student = context.Students.First(); var courses = student.Courses;
```

8. What is Eager Loading? How is it implemented?

Eager loading loads related data with the main entity using Include. It's useful to reduce round trips to DB.

Code Example:

```
var students = context.Students.Include(s => s.Courses).ToList();
```

9. What is the purpose of the virtual keyword in navigation properties?

It enables EF to create proxies for lazy loading. Without virtual, EF can't intercept navigation property access.

Code Example:

```
public virtual ICollection<Course> Courses { get; set; }
```

10. How do you configure a connection string for EF?

In appsettings.json or web.config, add the connection string and inject it via DbContextOptions.

Code Example (appsettings.json):

```
"ConnectionStrings": { "Default": "Server=.;Database=TestDb;Trusted_Connection=True;" }
```

11. What is the purpose of the OnModelCreating() method?

This method is used to configure model relationships, keys, constraints, and table mappings using Fluent API.

Code Example:

```
protected override void OnModelCreating(ModelBuilder modelBuilder) {
modelBuilder.Entity<Student>().HasKey(s => s.Id); }
```

12. What is migration in EF Code First?

Migration is a way to incrementally update the database schema from the code model. It's used to sync changes.

Code Example:

```
Add-Migration InitialCreate
Update-Database
```

13. How do you enable migrations in EF?

Use the Add-Migration command in Package Manager Console. It requires EF tools and DbContext properly set up.

Code Example:

```
Add-Migration AddStudentTable
```

14. How do you roll back a migration?

Use the Update-Database command with the target migration name to rollback to a previous migration.

Code Example:

```
Update-Database -Migration:PreviousMigrationName
```

15. How do you handle relationships (1-1, 1-many, many-many) in EF?

Define navigation properties and foreign keys in your model classes. EF will map them to correct DB relations.

Code Example:

```
public class Student { public int Id; public virtual ICollection<Course> Courses { get; set; } }
```

16. What are navigation properties?

Navigation properties are properties in an entity class that allow navigation to related entities, like foreign keys.

Code Example:

```
public virtual ICollection<Course> Courses { get; set; }
```

17. How do you update an entity in EF?

Retrieve the entity, update its properties, then call SaveChanges. EF tracks and applies the changes.

Code Example:

```
var student = context.Students.Find(1); student.Name = "Updated"; context.SaveChanges();
```

18. What is the difference between Add() and Attach()?

Add() marks the entity as new; Attach() treats it as unchanged. Attach() is used for existing records.

Code Example:

```
context.Attach(existingEntity); context.Entry(existingEntity).State = EntityState.Modified;
```

19. What is the purpose of the SaveChanges() method?

It commits all changes made to tracked entities to the database. Without it, no updates are saved.

Code Example:

```
context.SaveChanges();
```

20. How do you use LINQ with EF?

You can write LINQ queries to retrieve and manipulate data. EF translates it to SQL automatically.

Code Example:

```
var students = context.Students.Where(s => s.Name.Contains("John")).ToList();
```

21. How to retrieve related data using Include()?

Use Include() method on DbSet to load related entities (eager loading).

Code Example:

```
var students = context.Students.Include(s => s.Courses).ToList();
```

22. What are the common issues with EF performance?

Lazy loading overuse, unindexed queries, large result sets, and not using AsNoTracking() can degrade performance.

Code Example:

```
var data = context.Students.AsNoTracking().ToList();
```

23. What is shadow property in EF Core?

A shadow property is not defined in the .NET class but exists in the model and mapped to a database column.

Code Example:

```
modelBuilder.Entity<Student>().Property<DateTime>("CreatedDate");
```

24. How do you use Data Annotations for validation?

Apply attributes like [Required], [MaxLength], [Key] in model classes for validation and DB schema control.

Code Example:

```
public class Student { [Required] public string Name { get; set; } }
```

25. How can you execute raw SQL queries in EF?

Use FromSqlRaw() or ExecuteSqlRaw() to run SQL queries for reading or writing data directly.

Code Example:

```
var students = context.Students.FromSqlRaw("SELECT * FROM Students").ToList();
```

3-6 Exp

1. What are the advantages of using EF over ADO.NET?

EF simplifies database operations with LINQ, automated model generation, change tracking, and migrations. Unlike ADO.NET, EF reduces boilerplate SQL and allows strongly-typed access to entities.

Code Example:

```
var students = context.Students.Where(s => s.Age > 18).ToList();
```

2. How does EF track changes to entities?

EF tracks entity states (Added, Modified, Deleted) via ChangeTracker when entities are retrieved via DbContext. Changes are persisted using SaveChanges().

Code Example:

```
context.Entry(student).State = EntityState.Modified; context.SaveChanges();
```

3. Explain ChangeTracker and how it works.

ChangeTracker monitors entity states and helps EF know which changes to apply to the database. It can be used to access state and original values.

Code Example:

```
var entries = context.ChangeTracker.Entries();
```

4. What is the Unit of Work pattern in EF?

DbContext acts as a Unit of Work, managing all changes as a single transaction. It ensures consistency and optimizes performance.

Code Example:

```
context.Add(order); context.Add(payment); context.SaveChanges();
```

5. What are the lifecycle stages of an entity in EF?

Entities go through states: Detached, Added, Unchanged, Modified, Deleted. ChangeTracker tracks these states.

Code Example:

```
context.Entry(entity).State = EntityState.Added;
```

6. What is AsNoTracking() and when to use it?

Use AsNoTracking() for read-only queries to improve performance by disabling change tracking.

Code Example:

```
var students = context.Students.AsNoTracking().ToList();
```

7. How do you handle concurrency in EF?

Use a [Timestamp] or RowVersion column and handle DbUpdateConcurrencyException during save.

Code Example:

```
[Timestamp] public byte[] RowVersion { get; set; }
```

8. How do you configure a composite key in EF?

Use Fluent API in OnModelCreating() to define composite keys using HasKey.

Code Example:

```
modelBuilder.Entity<OrderDetail>().HasKey(od => new { od.OrderId, od.ProductId });
```

9. How do you seed initial data in EF Core?

Seed data using HasData() in OnModelCreating(). Data is inserted on Update-Database.

Code Example:

```
modelBuilder.Entity<Student>().HasData(new Student { Id = 1, Name = "John" });
```

10. What are value converters in EF Core?

They convert property values during read/write, useful for enums, custom types, or encryption.

Code Example:

```
builder.Property(p => p.Status).HasConversion<string>();
```

11. What is a shadow property and how is it useful?

Shadow properties exist in EF model but not in class. Useful for audit fields like CreatedDate.

Code Example:

```
modelBuilder.Entity<Student>().Property<DateTime>("CreatedDate");
```

12. What is Fluent API in EF and when should it be used?

Fluent API provides configuration options for EF models and overrides Data Annotations. Use it in OnModelCreating().

Code Example:

```
modelBuilder.Entity<Student>().Property(s => s.Name).HasMaxLength(50);
```

13. Difference between DbContext and ObjectContext.

DbContext is a wrapper around ObjectContext with simpler API. ObjectContext is more verbose and less commonly used.

Code Example:

```
using(var ctx = new DbContext()) { /* easier and modern API */ }
```

14. How do you handle transactions in EF?

Use Database.BeginTransaction() to ensure atomic operations across multiple DbContext calls.

Code Example:

```
using var tx = context.Database.BeginTransaction(); context.SaveChanges(); tx.Commit();
```

15. Explain how DbContextOptions are configured in EF Core.

DbContextOptions configure the database provider, connection string, and behaviors in Startup.cs or DI registration.

Code Example:

```
services.AddDbContext<AppDbContext>(options => options.UseSqlServer(config["Connection"]));
```

16. How do you implement soft delete in EF?

Add IsDeleted property, apply a global query filter to exclude deleted records.

Code Example:

```
modelBuilder.Entity<Student>().HasQueryFilter(s => !s.IsDeleted);
```

17. What are owned entities in EF Core?

Owned entities are value objects that share the same table as the owner. Use OwnsOne().

Code Example:

```
modelBuilder.Entity<Order>().OwnsOne(o => o.Address);
```

18. How to log SQL queries generated by EF?

Use ILoggerFactory or configure logging in DbContextOptionsBuilder.

Code Example:

```
optionsBuilder.LogTo(Console.WriteLine);
```

19. Explain the difference between FirstOrDefault and SingleOrDefault in EF.

FirstOrDefault returns first match or null; SingleOrDefault expects only one match, throws if more.

Code Example:

```
var student = context.Students.FirstOrDefault(s => s.Id == 1);
```

20. How do you map a table without a primary key?

Use HasNoKey() in Fluent API for views or legacy tables without keys.

Code Example:

```
modelBuilder.Entity<Report>().HasNoKey();
```

21. How to implement TPT, TPH, and TPC inheritance in EF?

- TPH: Default, uses discriminator column.
- TPT: One table per class.
- TPC: Each class has separate table, supported from EF Core 7.

  Code Example (TPH):

```
modelBuilder.Entity<Person>().HasDiscriminator<string>("PersonType");
```

22. How do you execute stored procedures in EF?

Use FromSqlRaw() for select or ExecuteSqlRaw() for insert/update.

Code Example:

```
context.Students.FromSqlRaw("EXEC GetTopStudents").ToList();
```

23. What is global query filter in EF Core?

It's a filter applied automatically to all queries for an entity, often used for multi-tenancy or soft deletes.

Code Example:

```
modelBuilder.Entity<Student>().HasQueryFilter(s => s.TenantId == currentTenantId);
```

24. What are backing fields in EF Core?

Backing fields store data for properties and allow control over get/set logic. Use HasField() in Fluent API.

Code Example:

```
modelBuilder.Entity<Student>().Property<string>("_name").HasField("_name");
```

25. How do you configure lazy loading in EF Core?

Install Microsoft.EntityFrameworkCore.Proxies and enable it in options. Use virtual keyword on navigation properties.

Code Example:

```
services.AddDbContext<AppDbContext>(options =>
options.UseLazyLoadingProxies().UseSqlServer(...));
```

6-12 Exp

1. **How do you optimize EF queries for large datasets?**
   Use AsNoTracking(), Select() for projections, Skip/Take for paging, and avoid unnecessary joins. Apply filters early to reduce data transfer.

```
var result = context.Orders.AsNoTracking().Where(o => o.Total > 1000).Select(o => new { o.Id,
o.Total }).ToList();
```

2. **How to improve performance in EF using compiled queries?**
   Compiled queries pre-compile LINQ expressions to improve repeated query performance.

```
static readonly Func<DbContext, decimal, IEnumerable<Order>> query =
EF.CompileQuery((DbContext ctx, decimal total) => ctx.Orders.Where(o => o.Total > total));
var orders = query(context, 1000);
```

3. **What are some EF Core limitations compared to EF6?**
   EF Core lacks full support for lazy loading proxies by default, complex mapping like EDMX, and limited support for some database providers (though improving).
4. **How do you implement repository and unit of work with EF?**
   Encapsulate DbContext in a UnitOfWork class and DbSet logic in generic repository.

```
public class UnitOfWork { public IGenericRepo<Order> Orders => new
GenericRepo<Order>(_context); }
```

5. **How to prevent N+1 problems in EF?**
   Use Include() and ThenInclude() to eager load related entities in a single query.

```
var orders = context.Orders.Include(o => o.Items).ToList();
```

6. **What are best practices for managing DbContext lifetimes?**
   Use Scoped lifetime in web apps to ensure one context per request. Avoid Singleton for DbContext.

```
services.AddDbContext<AppDbContext>(options => options.UseSqlServer(...),
ServiceLifetime.Scoped);
```

7. **How do you deal with disconnected entities in EF?**
   Attach entity and set the state manually before updating.

```
context.Attach(order); context.Entry(order).State = EntityState.Modified;
```

8. **What's the impact of tracking vs no-tracking on memory and performance?**
   Tracking increases memory usage and overhead, suitable for updates. No-tracking is faster and ideal for read-only operations.

```
var data = context.Orders.AsNoTracking().ToList();
```

9. **Explain query caching in EF Core.**
   EF Core caches the query plan (not data) when using compiled queries. EF doesn't cache result data by default.
10. **What is batching in EF Core and how does it improve performance?**
    EF Core batches insert/update/delete operations into fewer SQL commands to reduce round-trips.

```
context.AddRange(orders); context.SaveChanges();
```

11. **What is the difference between AddRange() and Add()?**
    Add() adds a single entity, AddRange() adds multiple. AddRange() reduces overhead.

```
context.AddRange(orderList); context.SaveChanges();
```

12. **How to handle model changes in production with minimal downtime?**
    Use migration scripts with careful versioning, run dotnet ef migrations script, apply during maintenance.
13. **Explain how migrations can be customized for DevOps pipelines.**
    Use dotnet ef migrations script to generate SQL files, check into source control, apply using deployment tools.
14. **How do you use interceptors in EF Core?**
    Create a class implementing IDbCommandInterceptor or ISaveChangesInterceptor.

```
public class LoggingInterceptor : IDbCommandInterceptor { /* log SQL here */ }
```

15. **How do you audit changes to entities in EF?**
    Use ChangeTracker.Entries() to inspect entity states and log changes before SaveChanges().

```
var changes = context.ChangeTracker.Entries().Where(e => e.State == EntityState.Modified);
```

16. **How do you configure schema names and table mappings dynamically?**
    Use ModelBuilder.HasDefaultSchema() or ToTable() inside OnModelCreating().

```
modelBuilder.HasDefaultSchema("tenant1");
```

17. **Can you implement multitenancy using EF Core?**
    Yes, with global query filters, separate schemas or databases, and injecting tenant ID.

```
modelBuilder.Entity<Order>().HasQueryFilter(o => o.TenantId == currentTenant);
```

18. **What are strategies for managing large models in EF?**
    Split models into separate DbContexts (bounded contexts), modularize configurations via IEntityTypeConfiguration.
19. **How do you implement CQRS with EF Core?**
    Use separate read and write DbContexts, project read models via LINQ, write via commands.

```
public class ReadDbContext : DbContext { /* Only queries */ }
```

20. **How do you deal with performance issues in multi-table joins?**
    Avoid unnecessary includes, select only required fields, and use DTOs to flatten data.

21. **Can you implement EF in a microservices architecture?**
    Yes, each microservice should have its own DbContext and database schema. Avoid shared models.
22. **What are alternate keys and how do you configure them?**
    Alternate keys are unique constraints other than PK. Use HasAlternateKey().

```
modelBuilder.Entity<User>().HasAlternateKey(u => u.Email);
```

23. **How does EF Core support spatial data types?**
    Install NetTopologySuite, and configure in UseSqlServer() or similar.

```
options.UseSqlServer(connStr, x => x.UseNetTopologySuite());
```

24. **What's the best way to implement pagination in EF?**
    Use Skip() and Take() on ordered queries.

```
var pageData = context.Products.Skip(20).Take(10).ToList();
```

25. **Explain concurrency tokens and their usage.**
    Concurrency tokens like RowVersion are used to detect concurrent edits and throw exceptions.

```
[Timestamp] public byte[] RowVersion { get; set; }
```

12-18 Exp

1. **What are the architectural considerations for using EF in enterprise systems?**
    Separate layers (DAL, BLL), avoid DbContext leakage, follow SOLID, and use repositories and unit of work patterns.

```
public interface IOrderRepository { Task<Order> GetByIdAsync(int id); }
```

2. **How do you scale EF in high-load systems?**
    Use read replicas, connection pooling, async operations, no-tracking queries, and distributed caching.

```
services.AddDbContextPool<AppDbContext>();
```

3. **How to isolate EF logic using Clean Architecture or Onion Architecture?**
    Keep EF in Infrastructure layer. Use interfaces in core layer and inject via DI.

```
public class OrderService(IOrderRepository repo) { }
```

4. **What are advanced concurrency conflict resolution strategies?**
    Use RowVersion, catch DbUpdateConcurrencyException, merge or prompt user for conflict resolution.

```
context.Entry(order).OriginalValues["RowVersion"] = clientVersion;
```

5. **What role does EF play in Domain-Driven Design?**
    EF is infrastructure, not domain. Use aggregate roots, map domain entities to EF models.

```
modelBuilder.Entity<Order>().OwnsMany(o => o.Items);
```

6. **How do you abstract EF behind application services?**
    Expose interfaces in application layer, implement using EF in infrastructure.

```
public interface IUnitOfWork { Task CommitAsync(); }
```

7. **What strategies do you follow for database versioning and backward compatibility?**
    Use semantic migration scripts, avoid destructive changes, ensure zero-downtime deployments.
8. **How do you handle schema evolution in distributed teams using EF?**
    Use shared migration projects, enforce CI/CD validations, branch-based migration isolation.
9. **Explain database sharding support with EF (or workarounds).**
    EF doesn't support native sharding. Use sharding libraries or route DbContext to tenant-specific databases.

```
options.UseSqlServer(GetConnectionForTenant(tenantId));
```

10. **How do you benchmark EF vs Dapper for performance?**
    Use BenchmarkDotNet or custom Stopwatch code, profile memory and speed.

```
var sw = Stopwatch.StartNew(); var data = context.Users.ToList(); sw.Stop();
```

11. **When is EF not a good choice and why?**
    When raw performance is critical, you need fine-grained SQL, or legacy schema support is needed.
12. **How do you build resilient, retry-enabled data access with EF?**
    Use EnableRetryOnFailure() in SQL provider, wrap calls in Polly policies.

```
options.UseSqlServer(connStr, opt => opt.EnableRetryOnFailure());
```

13. **What is the future of EF in cloud-native architectures?**
    EF continues evolving for cloud via performance, diagnostics, and distributed patterns compatibility.
14. **How do you combine EF with caching strategies like Redis or MemoryCache?**
    Cache query results manually with key-based lookups.

```
var users = cache.GetOrCreate("users", () => context.Users.ToList());
```

15. **How to implement async streaming with EF for real-time applications?**
    Use IAsyncEnumerable<T> and await foreach for large read pipelines.

```
await foreach (var user in context.Users.AsAsyncEnumerable()) { }
```

16. **What are considerations when using EF in Azure Functions or AWS Lambdas?**
    Use transient context, avoid sync code, ensure thread safety.

```
using var context = new AppDbContext();
```

17. **How can EF Core be used in multitenant SaaS systems securely?**
    Inject tenant ID, use schema-per-tenant or database-per-tenant strategy with query filters.

```
modelBuilder.Entity<Order>().HasQueryFilter(o => o.TenantId == tenantId);
```

18. **How do you design EF models to be forward-compatible?**
    Avoid removing columns, version entities, use nullables and feature flags.
19. **What is the impact of EF on memory usage in containerized environments?**
    Track context size, use AsNoTracking, limit query scope, and avoid context reuse.
20. **How do you validate data and business rules with EF and FluentValidation?**
    Use FluentValidation in services layer before calling EF SaveChanges().

```
var result = validator.Validate(order); if (!result.IsValid) return;
```

21. **How do you enforce domain rules using aggregate roots and EF?**
    Aggregate roots should encapsulate rules; EF only persists state.

```
public void AddItem(OrderItem item) { if (item.Qty > 0) Items.Add(item); }
```

22. **What are your EF Core upgrade strategies during major migrations?**
    Use preview/testing envs, script-based migration, check for data loss, automate rollback paths.
23. **How do you secure EF queries against injection or data exposure?**
    Use parameterized queries, avoid exposing DbContext directly, and whitelist fields in projection.
24. **How do you audit data changes across microservices using EF?**
    Use Outbox pattern, ChangeTracker, and save events alongside entities.

```
context.OutboxEvents.Add(new EntityChangedEvent(...));
```

25. **How do you ensure EF doesn't become the bottleneck in large-scale systems?**
    Optimize queries, index DB properly, scale reads, use Dapper where needed, monitor EF logs.

```
context.ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;
```


Best Wishes
for the Interview!
Be confident and believe
in yourself.