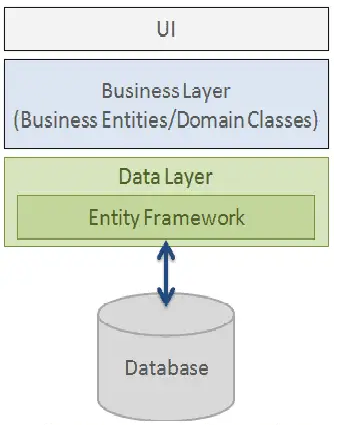
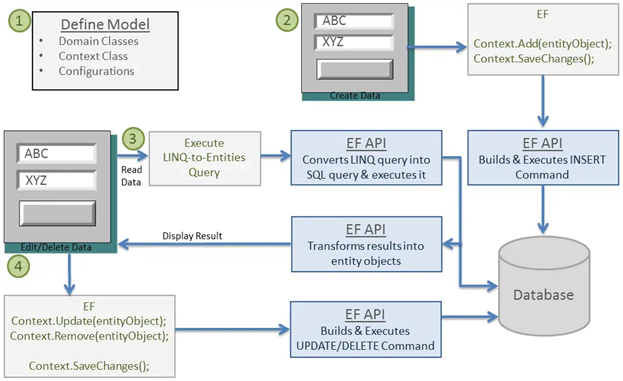
**EF**

Entity Framework is an open-source ORM framework for .NET applications supported by Microsoft. It enables developers to work with data using objects of domain specific classes without focusing on the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code compared with traditional applications.

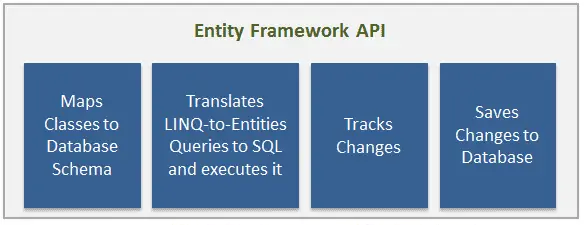


**Basic Workflow in Entity Framework**



**How Entity Framework Works?**

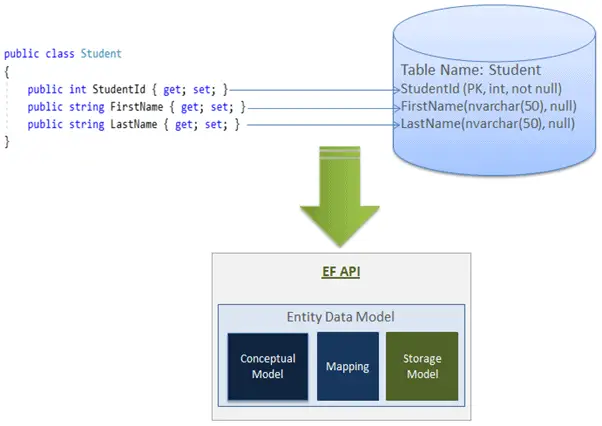
Entity Framework API (EF6 & EF Core) includes the ability to map domain (entity) classes to the database schema, translate & execute LINQ queries to SQL, track changes occurred on entities during their lifetime, and save changes to the database.



**EF API**

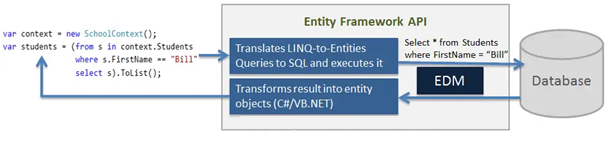
*Entity Data Model*

The very first task of EF API is to build an Entity Data Model (EDM). EDM is an in-memory representation of the entire metadata: conceptual model, storage model, and mapping between them.



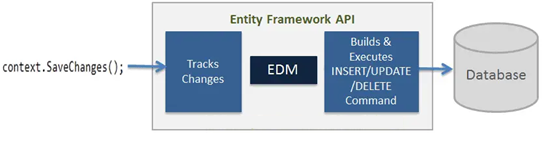
Querying

EF API translates LINQ-to-Entities queries to SQL queries for relational databases using EDM and also converts results back to entity objects.

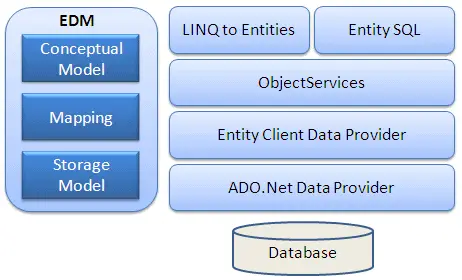


Saving

EF API infers INSERT, UPDATE, and DELETE commands based on the state of entities when the SaveChanges() method is called. The ChangeTrack keeps track of the states of each entity as and when an action is performed.



**Entity Framework Architecture**



**Context Class in Entity Framework**

The context class is a most important class while working with EF 6 or EF Core. It represent a session with the underlying database using which you can perform CRUD (Create, Read, Update, Delete) operations.

using System.Data.Entity;

public class SchoolContext : DbContext

{

// Entities

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

public DbSet<StudentAddress> StudentAddresses { get; set; }

public DbSet<Grade> Grades { get; set; }

}

**Entity in Entity Framework?**

An entity in Entity Framework is a class that maps to a database table. This class must be included as a DbSet<TEntity> type property in the DbContext class. EF API maps each entity to a table and each property of an entity to a column in the database.

For example, the following Student, and Grade are domain classes in the school application.

public class Student

{

public int StudentID { get; set; }

public string StudentName { get; set; }

public DateTime DateOfBirth { get; set; }

public Grade Grade { get; set; }

}

public class Grade

{

public int GradeId { get; set; }

public string GradeName { get; set; }

public string Section { get; set; }

public ICollection<Student> Students { get; set; }

}

In the above context class, Students, and Grades properties of type DbSet<TEntity> are called entity sets. The Student, and Grade are entities. EF API will create the Students and Grades tables in the database.

**Types of Entities in Entity Framework**

There are two types of Entities in Entity Framework: POCO Entities and Dynamic Proxy Entities.

POCO Entities (Plain Old CLR Object)

A POCO entity is a class that doesn't depend on any framework-specific base class. It is like any other normal .NET CLR class, which is why it is called "Plain Old CLR Objects".

Dynamic Proxy Entities (POCO Proxy)

Dynamic Proxy is a runtime proxy class which wraps POCO entity. Dynamic proxy entities allow **lazy loading**.

**Note:** Dynamic proxy entities are only supported in EF 6. EF Core 2.0 does not support them yet.

A POCO entity should meet the following requirements to become a POCO proxy:

1. A POCO class must be declared with public access.
2. A POCO class must not be sealed (NotInheritable in Visual Basic).
3. A POCO class must not be abstract (MustInherit in Visual Basic).
4. Each navigation property must be declared as public, virtual.
5. Each collection property must be ICollection<T>.
6. The ProxyCreationEnabled option must **NOT** be false (default is true) in context class.

**EntityState in Entity Framework**

EF API maintains the state of each entity during its lifetime. Each entity has a state based on the operation performed on it via the context class. The entity state represented by an enum System.Data.Entity.EntityState in EF 6 and Microsoft.EntityFrameworkCore.EntityState in EF Core with the following values:

* Added
* Modified
* Deleted
* Unchanged
* Detached

**Properties in Entities**

An Entity can include two types of properties: Scalar Properties and Navigation Properties.

Scalar Property

The primitive type properties are called scalar properties. Each scalar property maps to a column in the database table which stores an actual data.

Navigation Property

The navigation property represents a relationship to another entity.

There are two types of navigation properties: Reference Navigation and Collection Navigation

Reference Navigation Property

If an entity includes a property of another entity type, it is called a Reference Navigation Property. It points to a single entity and represents multiplicity of one (1) in the entity relationships.

EF API will create a ForeignKey column in the table for the navigation properties that points to a PrimaryKey of another table in the database.

Collection Navigation Property

If an entity includes a property of generic collection of an entity type, it is called a collection navigation property. It represents multiplicity of many (\*).

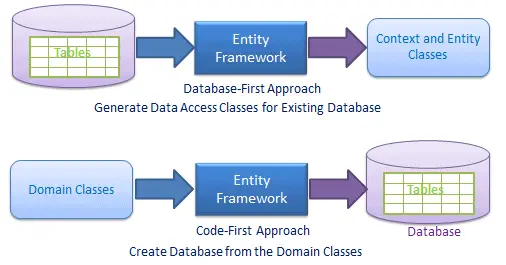
EF API does not create any column for the collection navigation property in the related table of an entity, but it creates a column in the table of an entity of generic collection.

**Entity Framework Core**

Entity Framework Core is the new version of Entity Framework after EF 6.x. It is open-source, lightweight, extensible and a cross-platform version of Entity Framework data access technology.

**EF Core Development Approaches**

EF Core supports two development approaches 1) Code-First 2) Database-First. EF Core mainly targets the code-first approach and provides little support for the database-first approach



EF Core features

1. DbContext & DbSet
2. Data Model
3. Querying using Linq-to-Entities
4. Change Tracking
5. SaveChanges
6. Migrations

**EF Core Database Providers**

Database NuGet Package

SQL Server Microsoft.EntityFrameworkCore.SqlServer

MySQL MySql.Data.EntityFrameworkCore

PostgreSQL Npgsql.EntityFrameworkCore.PostgreSQL

SQLite Microsoft.EntityFrameworkCore.SQLite

SQL Compact EntityFrameworkCore.SqlServerCompact40

In-memory Microsoft.EntityFrameworkCore.InMemory

**Install NuGet packages for the following two things to use EF Core in the application:**

EF Core DB provider 🡺 Microsoft.EntityFrameworkCore.SqlServer

EF Core tools 🡺 Microsoft.EntityFrameworkCore.Tools [Microsoft.EntityFrameworkCore.Tools.Dotnet (CLI deprecated)]