### **What is ORM?**

Object-Relational Mapping (ORM) is a programming technique that allows developers to interact with a relational database using an object-oriented programming language, such as C#. Instead of writing raw SQL queries, you work with objects and classes that represent the data in your database. The ORM acts as a translator, converting your object-oriented code into SQL commands that the database can understand and vice versa.

At its core, ORM simplifies data access by creating a "virtual object database" that you can work with from within your application. This abstraction layer handles the complexities of database interactions, making your code cleaner, more readable, and easier to maintain.

#### **How ORM Maps C# Classes to Database Tables**

The magic of ORM lies in its ability to map the properties of your C# classes to the columns of your database tables. Let's consider a simple example:

**C# Class:**

C#

public class Product  
{  
 public int ProductId { get; set; }  
 public string Name { get; set; }  
 public decimal Price { get; set; }  
}

**SQL Server Table:**

SQL

CREATE TABLE Products (  
 ProductId INT PRIMARY KEY,  
 Name NVARCHAR(100),  
 Price DECIMAL(18, 2)  
);

The ORM, in this case, Entity Framework Core, would be configured to understand that the Product class corresponds to the Products table. It would also map the ProductId, Name, and Price properties of the Product class to the respective columns in the Products table. When you want to retrieve a product from the database, you would use your programming language to query for a Product object, and the ORM would generate the necessary SELECT statement. Similarly, when you create a new Product object and save it, the ORM generates the corresponding INSERT statement.

#### **Benefits of ORM**

Using an ORM like Entity Framework Core offers several significant advantages:

* **Productivity:** ORMs significantly reduce the amount of boilerplate code you need to write for data access. You don't have to manually create SqlConnection and SqlCommand objects or write repetitive INSERT, UPDATE, and DELETE statements. This allows you to focus on the business logic of your application.
* **Maintainability:** By abstracting away the database-specific code, ORMs make your application easier to maintain and refactor. If you need to change a column name in your database, you only need to update the corresponding property in your C# class, and the ORM will handle the rest.
* **Abstraction from SQL:** ORMs allow you to write database queries using the syntax of your programming language (e.g., LINQ in C#). This is often more intuitive and less error-prone than writing raw SQL. It also provides a level of database independence; in some cases, you can switch the underlying database with minimal changes to your application code.

### **EF Core vs EF Framework**

Entity Framework (EF) is Microsoft's object-relational mapping framework for .NET. Over the years, it has evolved, leading to two major versions: the original Entity Framework (often referred to as EF6) and the more modern Entity Framework Core (EF Core).

|  |  |  |
| --- | --- | --- |
| Feature | EF Core | EF Framework (EF6) |
| **Platform** | Cross-platform (Windows, macOS, Linux) | Windows-only |
| **Performance** | Generally faster and more lightweight | Can be slower and more memory-intensive |
| **Modern Features** | Supports LINQ, async queries, compiled queries, and has a more extensible architecture | Has a mature feature set but lacks some of the modern performance optimizations of EF Core |
| **Flexibility** | More flexible and modular, allowing you to only include the features you need | More monolithic in its design |

In essence, **EF Core** is the recommended choice for new .NET applications due to its cross-platform nature, superior performance, and ongoing development with modern features. **EF Framework (EF6)** is still a viable option for existing applications that are tightly coupled to the Windows ecosystem and have a long history with the framework.

### **EF Core 8.0 Features**

EF Core is continuously evolving, and version 8.0 introduced several exciting new features and improvements. Some of the key highlights include:

* **JSON Column Mapping:** This allows you to map a property of your C# class to a JSON column in your database. This is particularly useful for storing and querying semi-structured data without needing to create a separate table.
* **Improved Performance with Compiled Models:** EF Core 8.0 introduces AOT (Ahead-of-Time) compiled models, which can significantly reduce the startup time of your application. This is especially beneficial for serverless and cloud-native scenarios.
* **Interceptors and Better Bulk Operations:** Interceptors provide a way to intercept and modify database operations before they are sent to the database. EF Core 8.0 enhances this feature and also improves the performance of bulk operations like **BulkUpdate and BulkDelete.**

Program.cs

using RetailInventory.Data;

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

// Ensure DB is created

using (var context = new AppDbContext())

{

context.Database.EnsureCreated();

if (!context.Categories.Any())

{

// Seed categories and products

var electronics = new Category { Name = "Electronics" };

var groceries = new Category { Name = "Groceries" };

var products = new List<Product>

{

new Product { Name = "Laptop", StockLevel = 10, Category = electronics },

new Product { Name = "Smartphone", StockLevel = 15, Category = electronics },

new Product { Name = "Bread", StockLevel = 30, Category = groceries },

new Product { Name = "Milk", StockLevel = 20, Category = groceries }

};

context.Categories.AddRange(electronics, groceries);

context.Products.AddRange(products);

context.SaveChanges();

}

var inventory = context.Products.Include(p => p.Category).ToList();

Console.WriteLine("Retail Inventory:");

foreach (var product in inventory)

{

Console.WriteLine($"- {product.Name} ({product.Category.Name}) - Stock: {product.StockLevel}");

}

}

**Output:**

