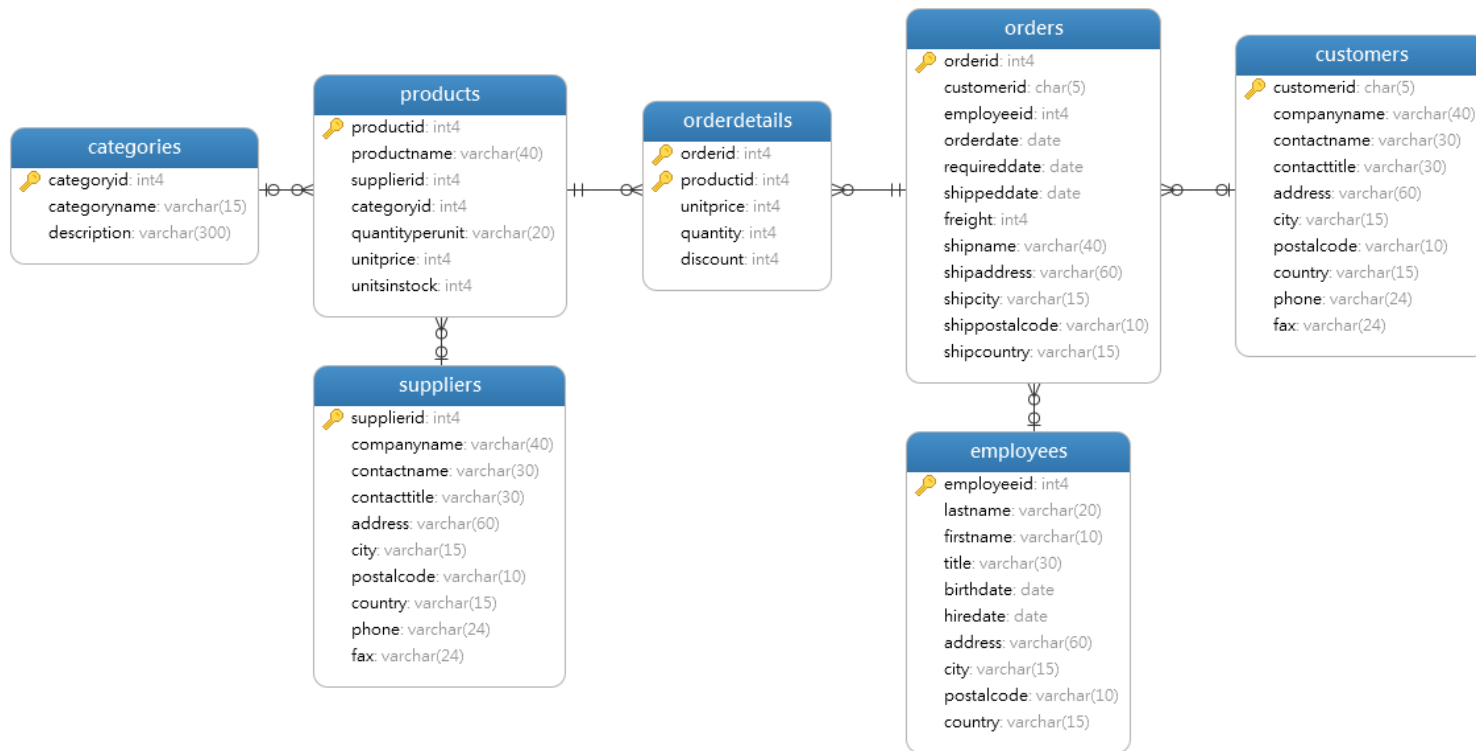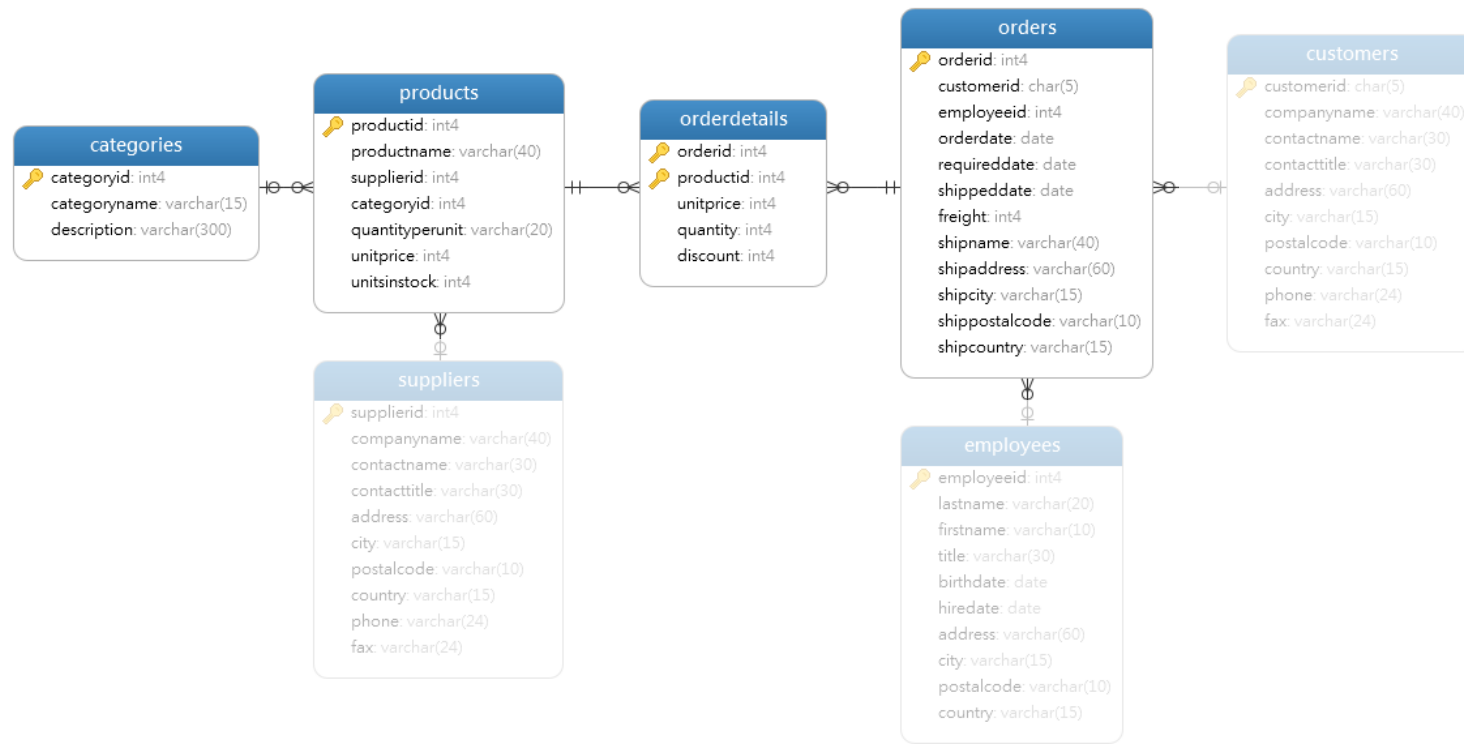# Complex IT-Systems Section 2

Henrik Bulskov

The Northwind Database

The Northwind Database

The Northwind Database

Create a new database on your local postgres named "northwind" (or whatever you like to call it – I use that name)

On Moodle you find an SQL script with the schema and data
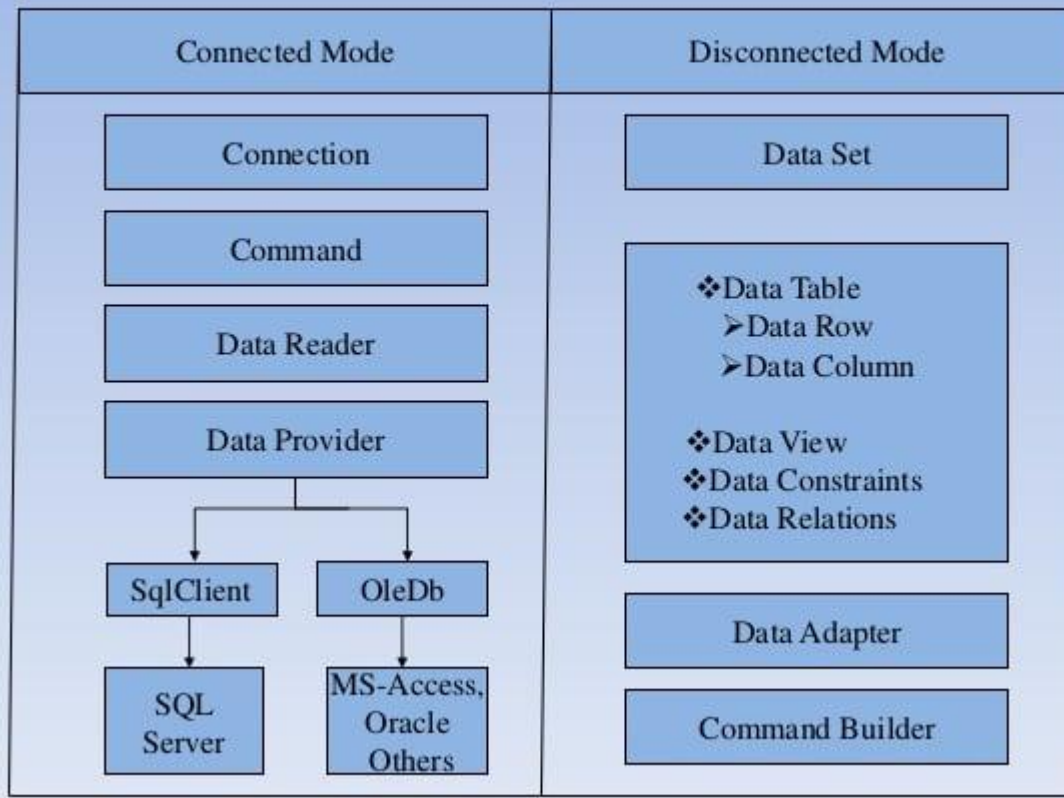
Just copy and past the script and run all the statements (it is small so it will only take a second)

You are then ready to start working with the database that we will use in the next assignment and in all the examples in the lectures ☺

ADO.NET

ADO.NET

ADO.NET Architecture

Connected Mode	Disconnected Mode
Connection	Data Set
Command	❖Data Table ➤Data Row ➤Data Column ❖Data View ❖Data Constraints ❖Data Relations
Data Reader	
Data Provider	
SqlClient / OleDb	Data Adapter
SQL Server / MS-Access, Oracle Others	Command Builder

7

```csharp
var connStr = "host=localhost;db=northwind;uid=bulskov;pwd=henrik";
using var conn = new NpgsqlConnection(connStr);
conn.Open();
using var cmd = new NpgsqlCommand();

cmd.Connection = conn;
cmd.CommandText = "select * from categories";

var reader = cmd.ExecuteReader();

while (reader.Read())
{
    Console.WriteLine(
        $"Id={reader.GetInt32(ordinal: 0)}, Name={reader.GetString(ordinal: 1)}");
}
```

ADO.NET Example

```csharp
using Npgsql;


var connStr = "host=localhost;db=northwind;uid=bulskov;pwd=henrik";
using var conn = new NpgsqlConnection(connStr);
conn.Open();
using var cmd = new NpgsqlCommand();

cmd.Connection = conn;
cmd.CommandText = "select * from categories";

var reader = cmd.ExecuteReader();

while (reader.Read())
{
    Console.WriteLine(
        $"Id={reader.GetInt32(ordinal: 0)}, Name={reader.GetString(ordinal: 1)}");
}
```

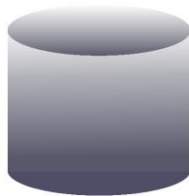# ADO.NET Example

Language INtegrated Query

LINQ

LINQ

Database
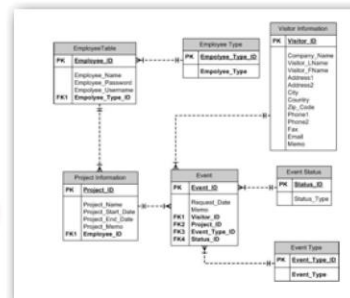
Data

Query

Collection

```
IEnumerable<T>
```

Object-Relational Mapping

- Object-relational mapping (ORM, O/RM, and O/R mapping tool) is a programming technique for converting data between incompatible type systems in object-oriented programming languages.
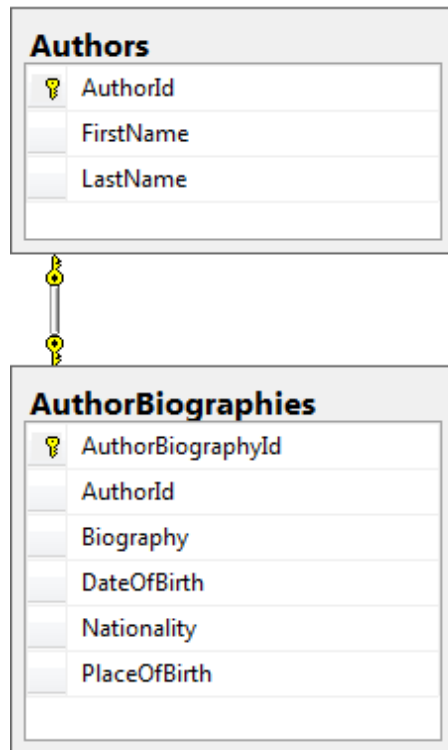
Mapping Object Relationships

One-to-one relationships

One-to-many relationships

Many-to-many relationships

One-to-one relationships

Authors
- 🔑 AuthorId
- FirstName
- LastName

AuthorBiographies
- 🔑 AuthorBiographyId
- AuthorId
- Biography
- DateOfBirth
- Nationality
- PlaceOfBirth

```csharp
public class Author
{
    public int AuthorId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public AuthorBiography Biography { get; set; }
}
public class AuthorBiography
{
    public int AuthorBiographyId { get; set; }
    public string Biography { get; set; }
    public DateTime DateOfBirth { get; set; }
    public string PlaceOfBirth { get; set; }
    public string Nationality { get; set; }
    public int AuthorId { get; set; }
    public Author Author { get; set; }
}
```

One-to-many relationships
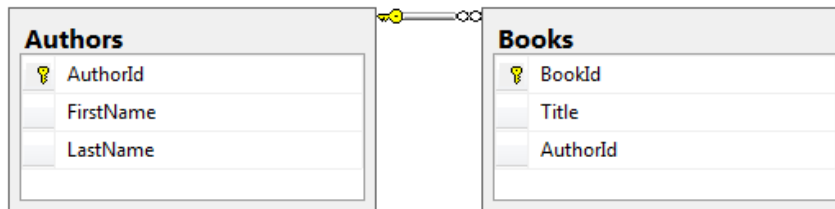

```
public class Author
{
    public int AuthorId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public ICollection<Book> Books { get; set; }
}
public class Book
{
    public int BookId { get; set; }
    public string Title { get; set; }
    public Author Author { get; set; }
}
```

https://www.learnentityframeworkcore.com

15

Many-to-many relationships

```csharp
public class Book
{
    public int BookId { get; set; }
    public string Title { get; set; }
    public Author Author { get; set; }
    public ICollection<Category> Categories { get; set; }
}

public class Category
{
    public int CategoryId { get; set; }
    public string CategoryName { get; set; }
    public ICollection<Book> Books { get; set; }
}
```

Many-to-many relationships

```csharp
public class Book
{
    public int BookId { get; set; }
    public string Title { get; set; }
    public Author Author { get; set; }
    public ICollection<BookCategory> BookCategories { get; set; }
}
public class Category
{
    public int CategoryId { get; set; }
    public string CategoryName { get; set; }
    public ICollection<BookCategory> BookCategories { get; set; }
}
public class BookCategory
{
    public int BookId { get; set; }
    public Book Book { get; set; }
    public int CategoryId { get; set; }
    public Category Category { get; set; }
}
```

Inheritance

18
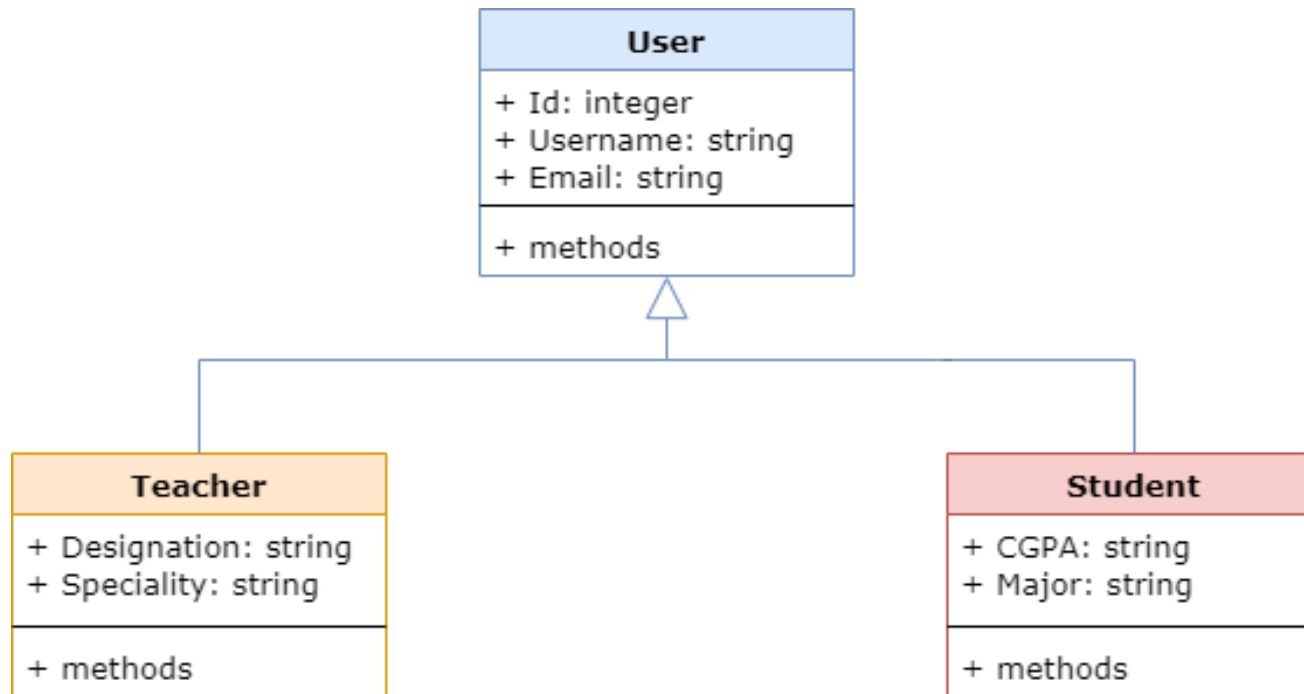
Mapping Inheritance Structures

- Table Per Hierarchy (TPH)
  - One table is used to represent all classes in the hierarchy. A "discriminator" column is used to discriminate between differing types.

- Table Per Type (TPT)
  - A separate table is used to represent each type in the inheritance chain, including abstract types. Tables that represent derived types are related to their base type via foreign keys.

- Table Per Concrete Type (TPC)
  - A separate table is used to represent each concrete type in the inheritance chain. Properties in any abstract base type are generated as fields in each concrete type's table. There is no relationship between differing types.

https://www.learnentityframeworkcore.com

19

Table per Hierarchy (TPH)

Table per Type (TPT)

21

Table per Concrete Type (TPC)

https://nishanc.medium.com/inheritance-strategy-in-entity-framework-core-5-46345e90e297
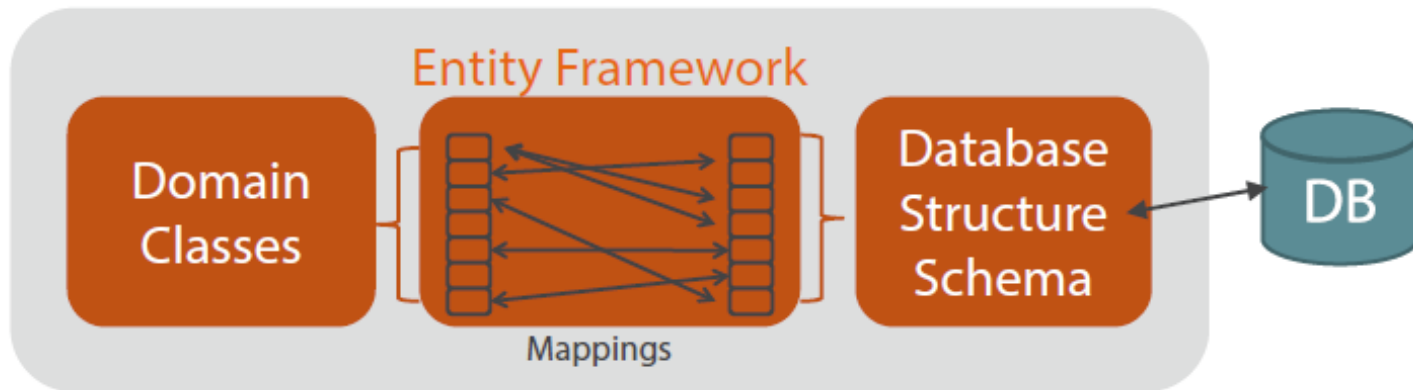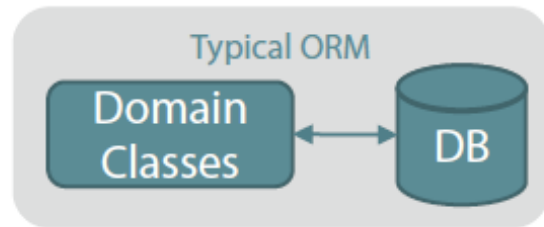
What is Entity Framework

Entity Framework Architecture

24
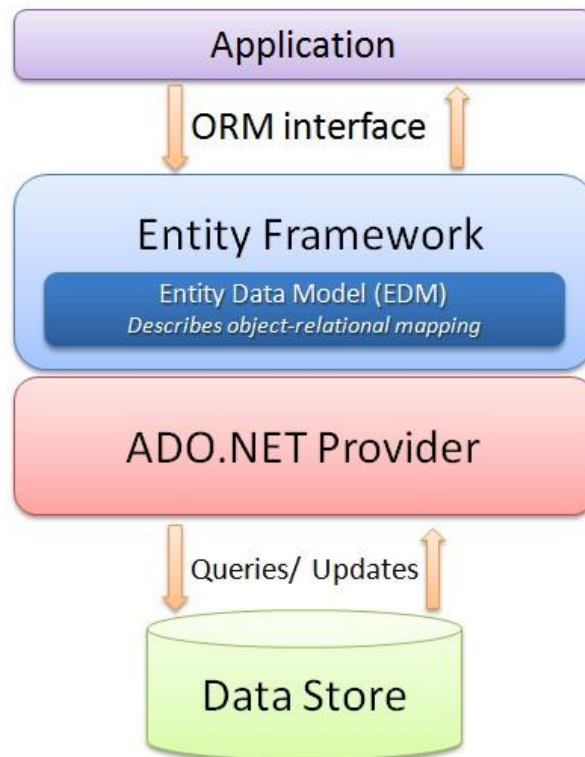
Entity Framework vs. Other ORMs

Why Entity Framework Core?

Developer Productivity

Consistent query syntax with LINQ to Entities

Focus on domain. Not on DB, connections, commends, etc.

How EF Works

```
public class Customer
{
  public int Id {get;set;}          ← Conventional Mapping
  public string FirstName {get;set;} ← Column Name="First_Name", Length=30
  public DateTime DateOfBirth {get;set;} ← Conventional Mapping
}
```

Table Name: Customers
Id (PK, int, not null)
First_Name(nvarchar(30),not null)
DateOfBirth(date, not null)

Mappings

28
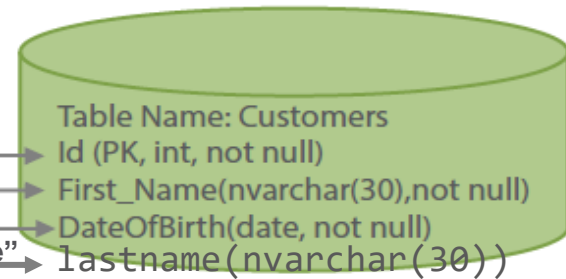
Mappings and Postgres
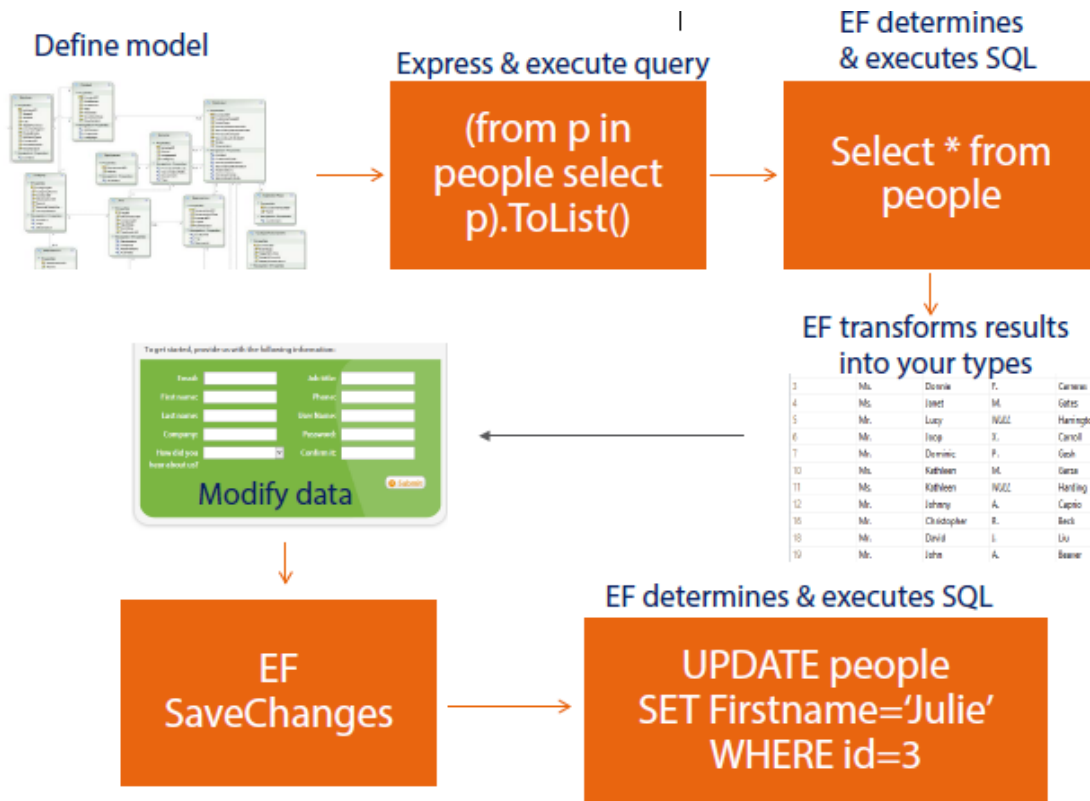
Basic Workflow

TRACKING

Eager Loading   Explicit Loading   Lazy Loading

Loading Related Data

33

Eager Loading — Explicit Loading — Lazy Loading
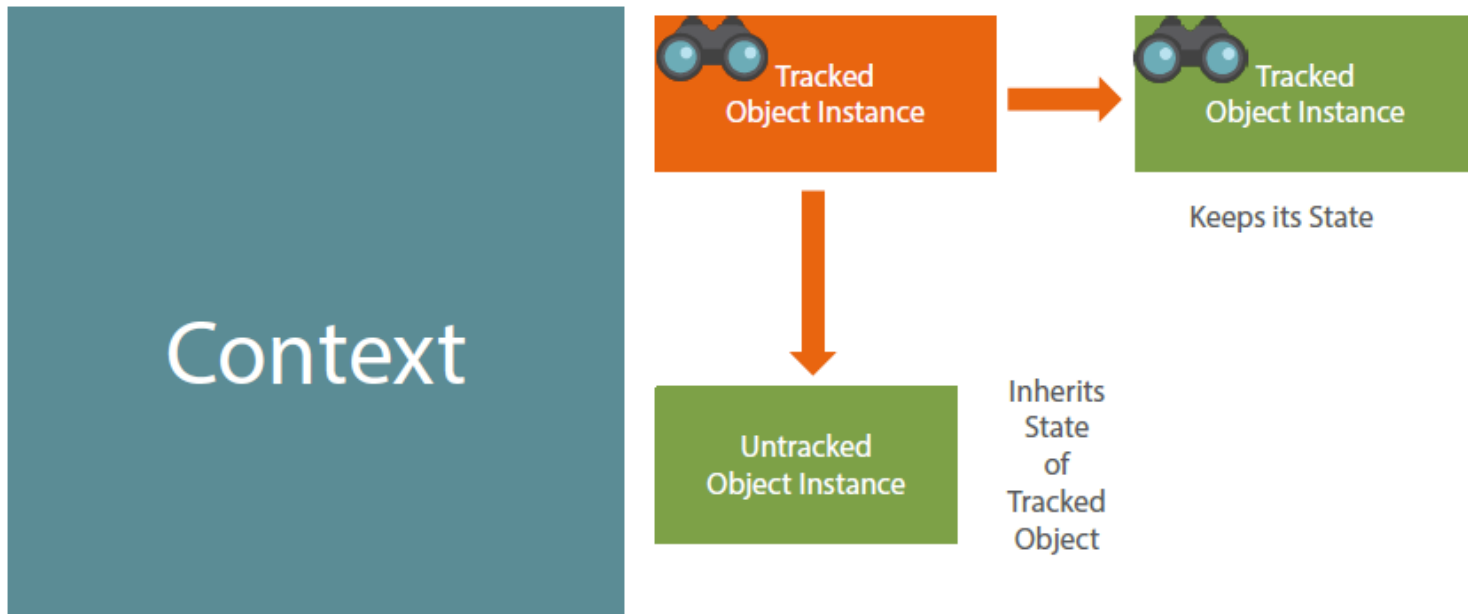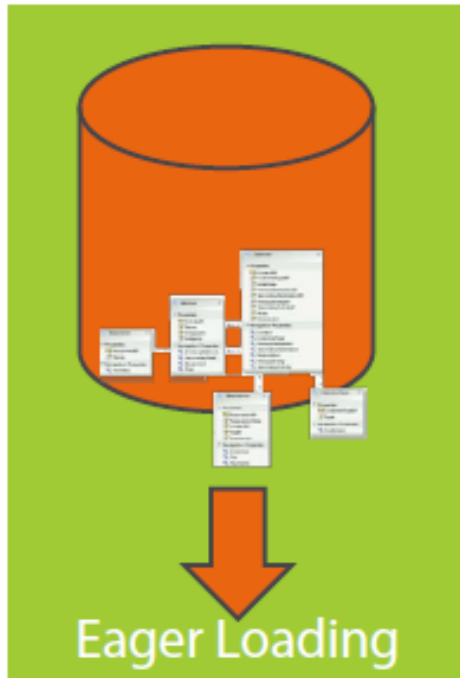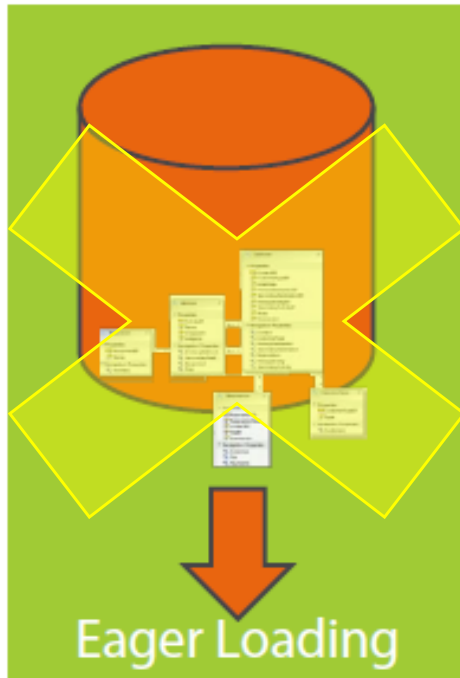
(Core < 2.1) Loading Related Data

34