An Introduction to

Entity Framework (EF)

Microsoft’s recommended data access technology for .Net applications

Microsoft .NET Entity Framework

Pre-Requisites

- Basic knowledge of .Net Framework,
- C#,
- ADO.Net
- Visual Studio
- MS SQL Server is required.

The Problem: Programming Data is Hard

- **Writing queries is difficult**
  - No help from compiler
  - Results are untyped rectangular records

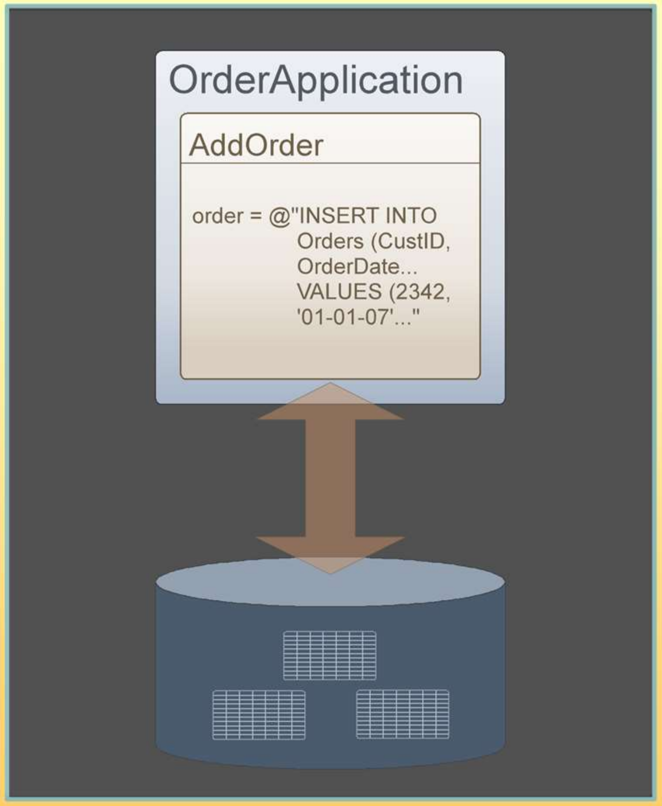
- **Database Schemas optimized for storage concerns**
  - Relational Tables contain flat, homogenous records
    Implicit Logic Embedded in Application
  - Brittle, Hard to maintain

- **Lack of common syntax across relational databases**

The Opportunity: Increase Developer Productivity

- **Rapid Development**
  - Strongly typed queries
  - Strongly typed results with Business Logic
- **Lower TCO**
  - Work with an explicit data model
    - Types, Inheritance, Relationships, Complex Properties,…
  - Decouple application from storage schema
- **Better Portability**
  - Common query language across disparate sources

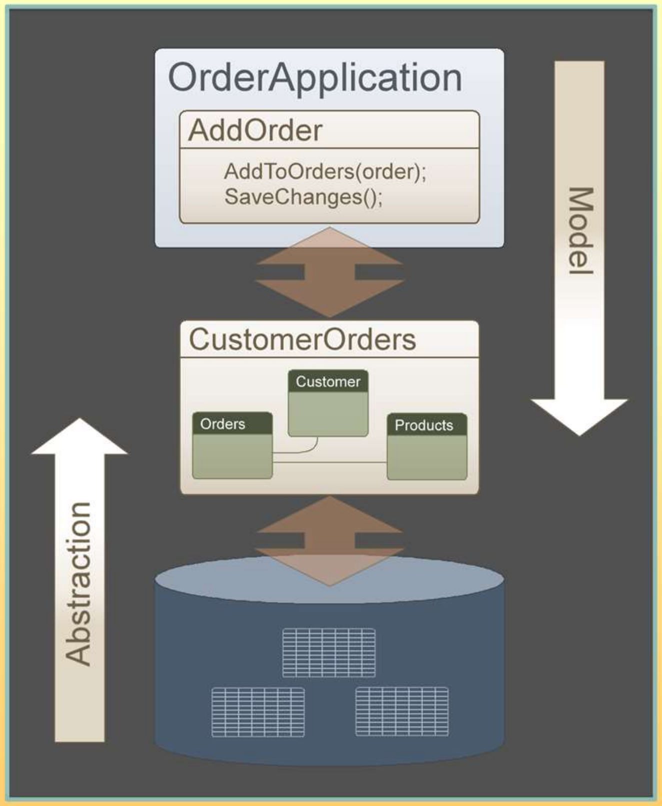
Where's Your Data Model?



```
order = @"INSERT INTO
          Orders (CustID,
          OrderDate...
          VALUES (2342,
          '01-01-07'..."
```

- **Applications Today…**
  - Implicitly Contain the Data Model
  - Logic and Model Intertwined
  - Conceptual Mismatch
  - Often encapsulate in a "Data Access Layer"

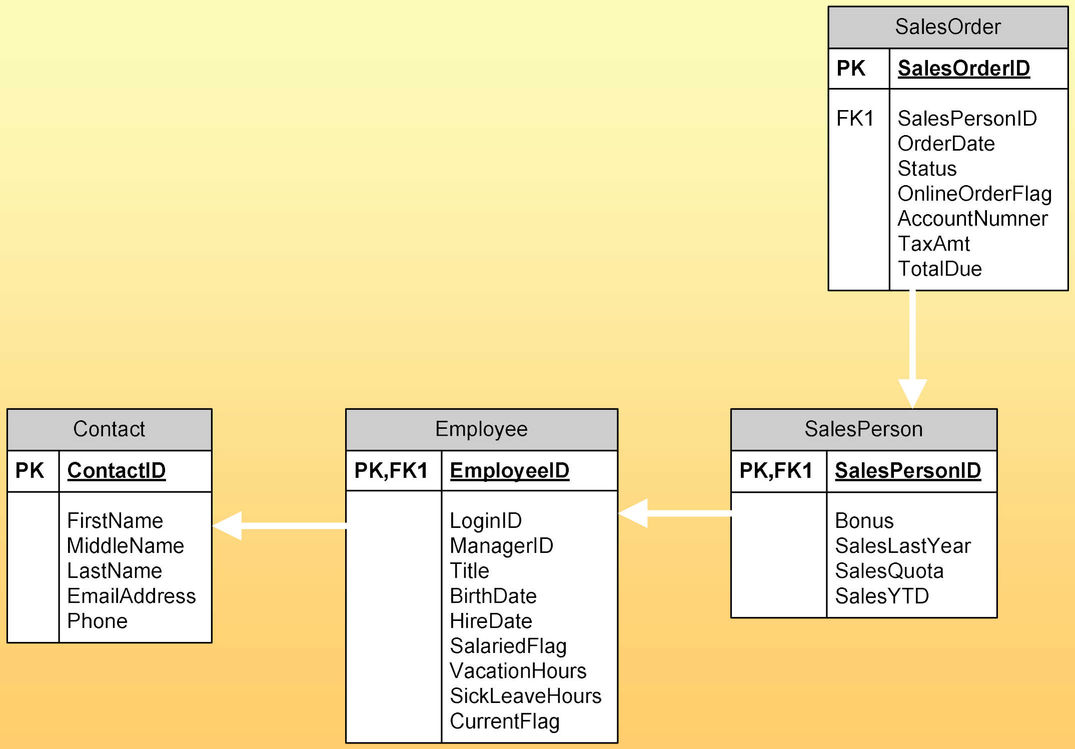
Where's Your Data Model?



- **Applications Today…**
  - Implicitly Contain the Data Model
  - Logic and Model Intertwined
  - Conceptual Mismatch
  - Often encapsulate in a "Data Access Layer"
- **The Need…**
  - Applications work with a well Defined Model
  - Storage Schema Abstraction
    - Declarative mapping between application and storage models
    - No brittle, hard-coded mapping

Data Modeling

- Every business application has, explicitly or implicitly, a conceptual data model
- Is the Database Schema Ideal for us?



Working with Database Schemas (1)

- **Get all full-time employees that were hired during 2006 and list their names and titles:**

```
SELECT c.FirstName, e.Title
FROM   Employee e
INNER JOIN Contact c ON e.EmployeeID = c.ContactID
WHERE e.HireDate >= '2006-01-01'
AND   e.SalariedFlag = 1
```

Create "Appropriate View" using JOIN

View only Subset of Data Explicitly

Working with Database Schemas (2)

- **Obtain all of the sales persons that have sales orders for more than $200,000:**

```
SELECT SalesPersonID, FirstName, LastName, HireDate
FROM  SalesPerson sp
INNER JOIN Employee e
      ON sp.SalesPersonID = e.EmployeeID
INNER JOIN Contact c
      ON e.EmployeeID = c.ContactID
INNER JOIN SalesOrder o
      ON sp.SalesPersonID = o.SalesPersonID
WHERE o.TotalDue > 200000
AND   e.SalariedFlag = 1
```

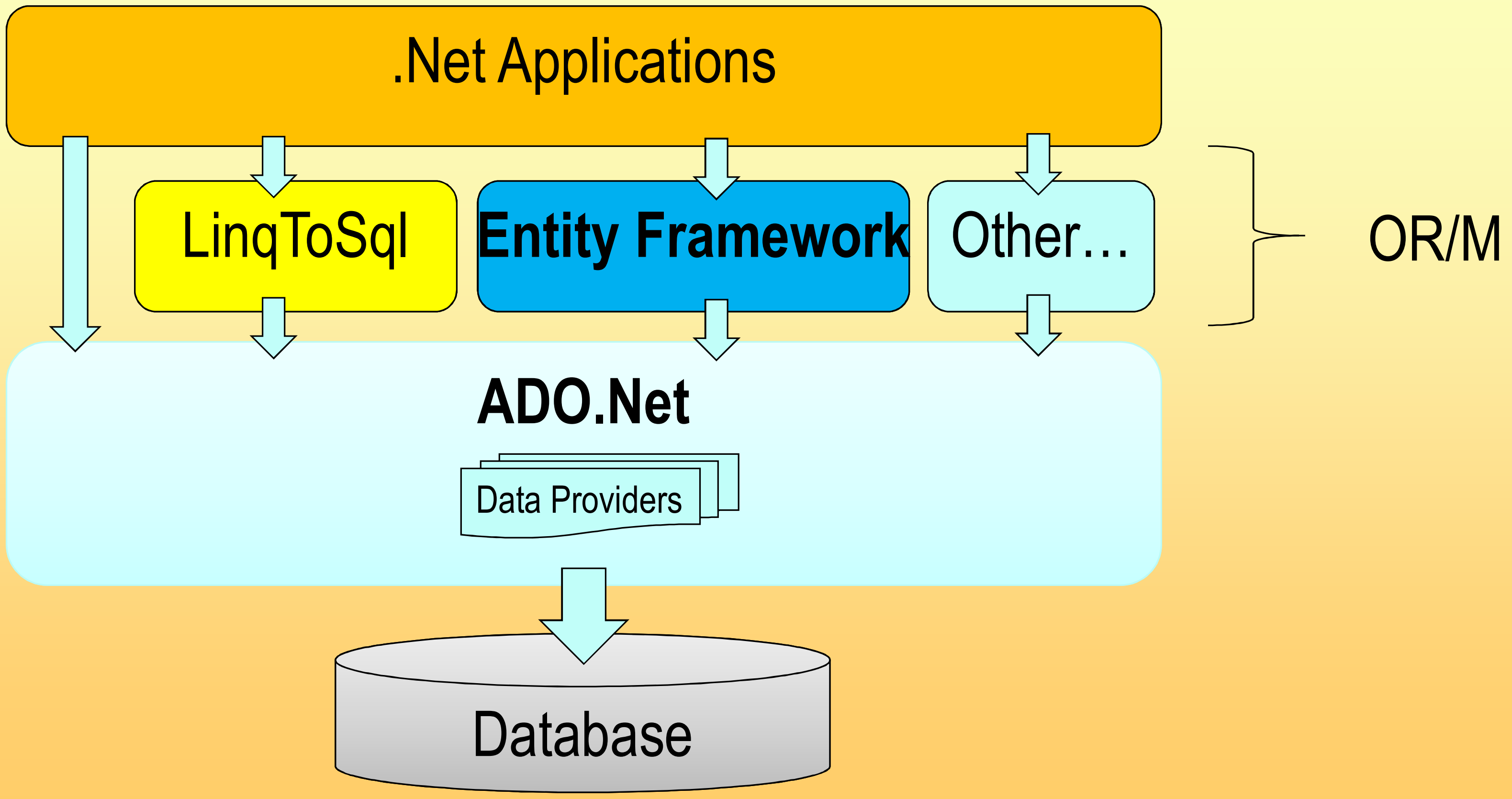
What is a Sales Person actually?

Very Complicated !

Schema is too fragmented

Cannot leverage Relationships

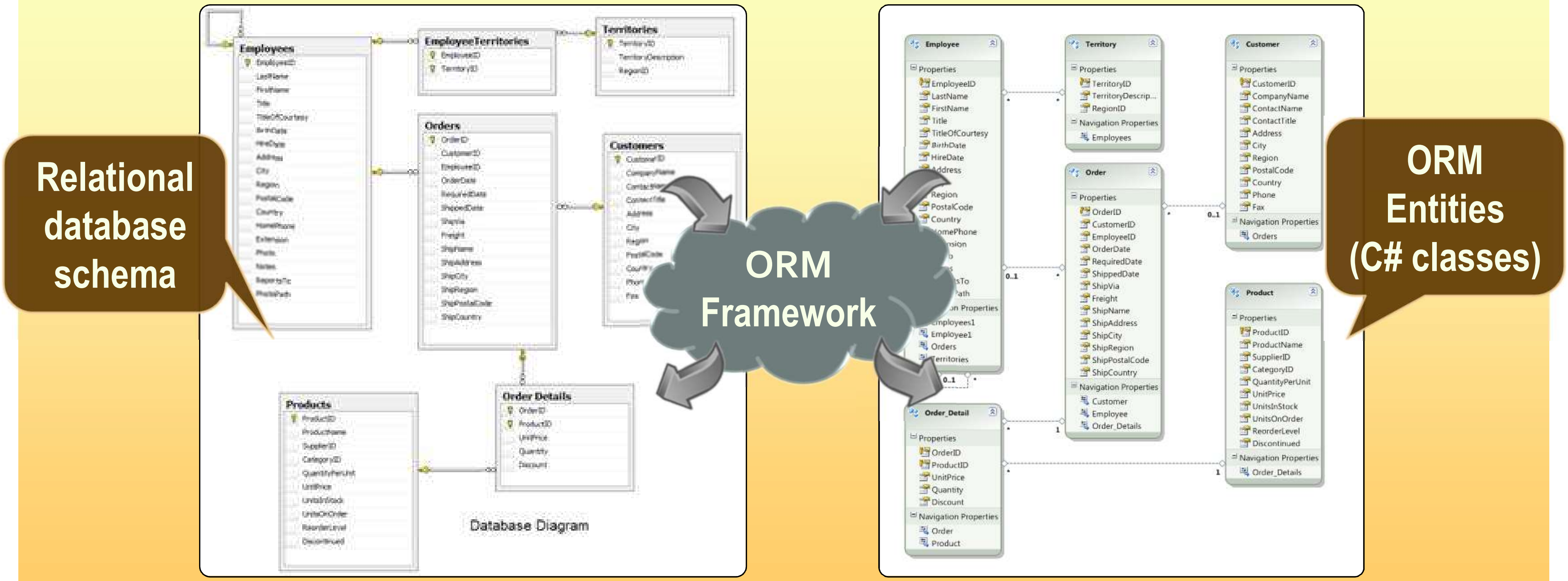
Data Access Technologies in .Net



What is Object-Relational Mapping - OR/M?

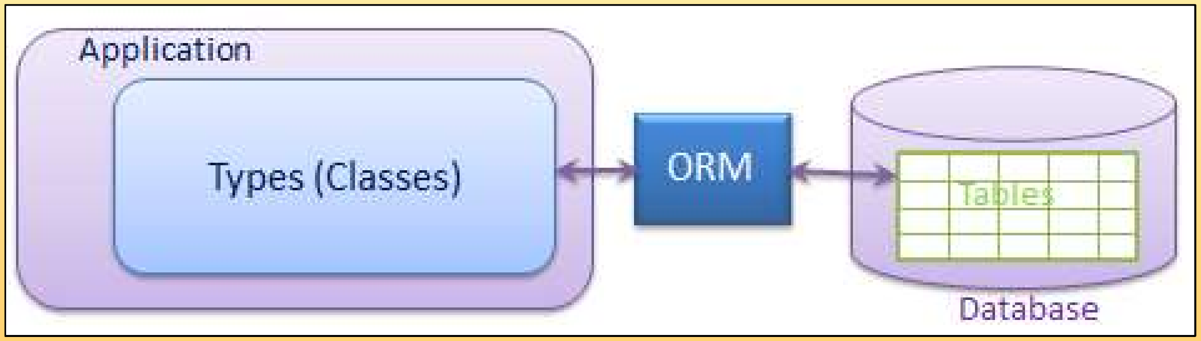
- **OR/M is a programming technique for automatic mapping data and database schema**
  - Map relational DB tables to classes and objects
- **OR/M creates a "virtual object database"**
  - Used from the programming language (C#, Java, PHP, …)
- **OR/M frameworks automate the ORM process**
  - A.k.a. **Object-Relational Persistence Frameworks**

OR/M Mapping – Example



What is EF?

- An Object/Relational Mapping **(O/RM)** framework
- It's an open source
- It's an enhancement to ADO.Net
- Gives developers an automated mechanism for accessing & storing the data in the database.
- Microsoft's recommended data access technology for new applications

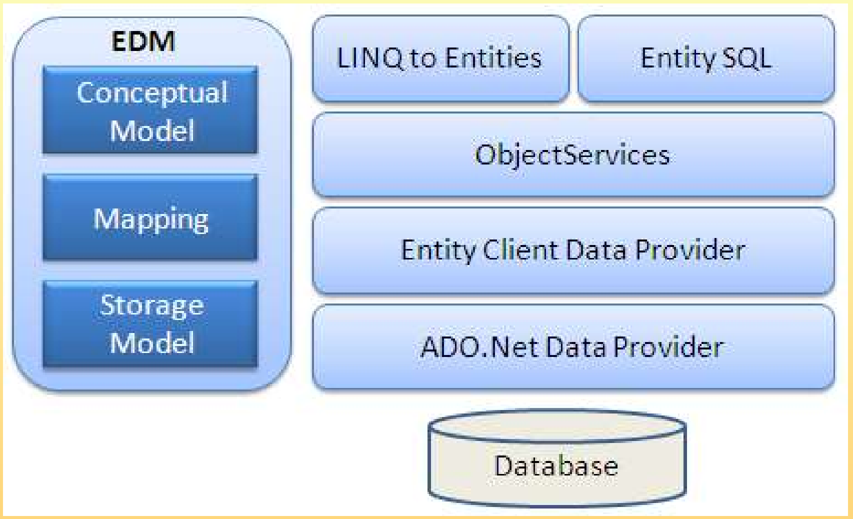


Overview of EF

- **Entity Framework (EF) is the standard ORM framework for .NET**
  - Maps relational database to C# object model
  - Powerful data manipulation API over the mapped schema
  - CRUD operations and complex querying with LINQ
- **Database first approach: from database to C# classes**
- **Code first approach: from classes to DB schema**
- **Visual Studio generates EF data models**
  - Data mappings consist of C# classes, attributes and XML

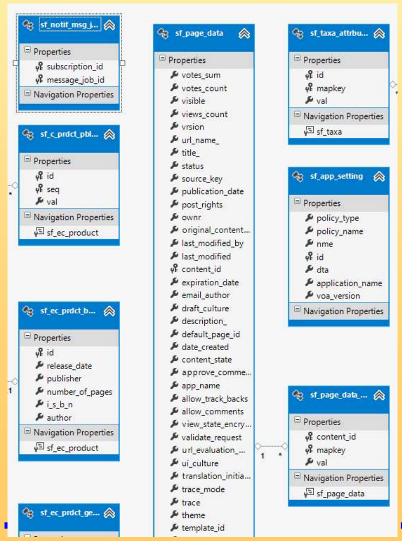
EF Architecture

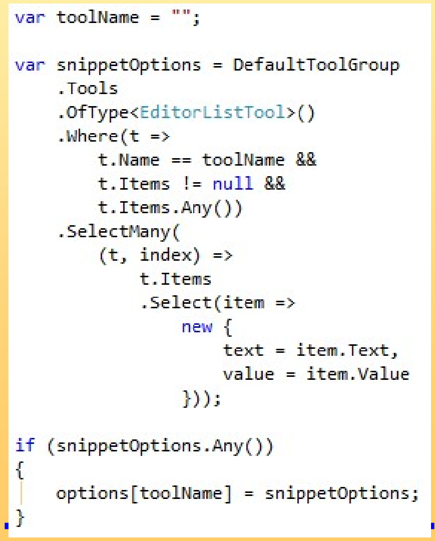
- **EDM – Entity Data Model**
  - Maps Tables with Entity Classes
- **Linq to Entities and Entity SQL**
  - Queries against object model and Returns Entities
- **Object Services**
  - Main entry point from application to database
  - Data materialization – Rows to Objects
- **Entity Client Data Provider**
  - Converts L2E and ESql to SQL statements
- **ADO.Net Data Provider**
  - Communicates with the database using ADO.Net



EF: Basic Workflow

1. Define the data model (use a DB visual designer or code first)
2. Write & execute query over **`IQueryable`**
3. EF generates & executes an SQL query in the DB

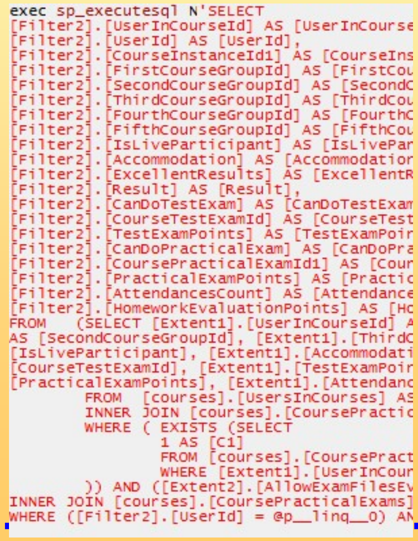




```
var toolName = "";

var snippetOptions = DefaultToolGroup
    .Tools
    .OfType<EditorListTool>()
    .Where(t =>
        t.Name == toolName &&
        t.Items != null &&
        t.Items.Any())
    .SelectMany(
        (t, index) =>
            t.Items
            .Select(item =>
                new {
                    text = item.Text,
                    value = item.Value
                }));

if (snippetOptions.Any())
{
    options[toolName] = snippetOptions;
}
```



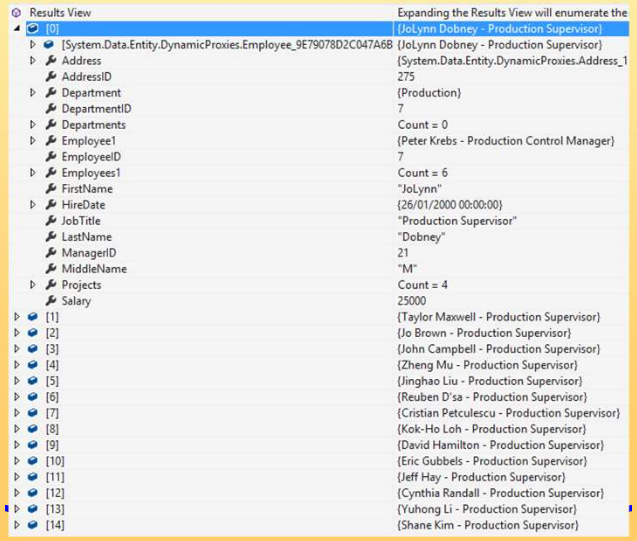
```
exec sp_executesql N'SELECT
[Filter2].[UserInCourseId] AS [UserInCourse
[Filter2].[UserId] AS [UserId],
[Filter2].[CourseInstanceId1] AS [CourseIns
[Filter2].[FirstCourseGroupId] AS [FirstCou
[Filter2].[SecondCourseGroupId] AS [SecondC
[Filter2].[ThirdCourseGroupId] AS [ThirdCou
[Filter2].[FourthCourseGroupId] AS [FourthC
[Filter2].[FifthCourseGroupId] AS [FifthCou
[Filter2].[IsLiveParticipant] AS [IsLivePar
[Filter2].[Accommodation] AS [Accommodation
[Filter2].[ExcellentResults] AS [ExcellentR
[Filter2].[Result] AS [Result],
[Filter2].[CanDoTestExam] AS [CanDoTestExam
[Filter2].[CourseTestExamId] AS [CourseTest
[Filter2].[TestExamPoints] AS [TestExamPoin
[Filter2].[CanDoPracticalExam] AS [CanDoPra
[Filter2].[CoursePracticalExamId1] AS [Cour
[Filter2].[PracticalExamPoints] AS [Practic
[Filter2].[AttendancesCount] AS [Attendance
[Filter2].[HomeworkEvaluationPoints] AS [Ho
FROM   (SELECT [Extent1].[UserInCourseId] A
AS [SecondCourseGroupId], [Extent1].[ThirdC
[IsLiveParticipant], [Extent1].[Accommodati
[CourseTestExamId], [Extent1].[TestExamPoin
[PracticalExamPoints], [Extent1].[Attendanc
        FROM  [courses].[UsersInCourses] AS
        INNER JOIN [courses].[CoursePractic
        WHERE ( EXISTS (SELECT
                1 AS [C1]
                FROM [courses].[CoursePract
                WHERE [Extent1].[UserInCour
        )) AND ([Extent2].[AllowExamFilesEv
INNER JOIN [courses].[CoursePracticalExams]
WHERE ([Filter2].[UserId] = @p__linq__0) AN
```

EF: Basic Workflow (2)

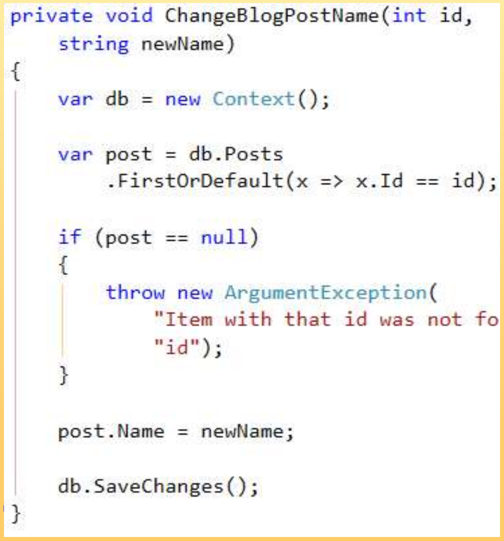
4. EF transforms the query results into .NET objects

5. Modify data with C# code and call "Save Changes"

6. Entity Framework generates & executes SQL command to modify the DB



```
Results View                                                   Expanding the Results View will enumerate the
◢ [0]                                                          {JoLynn Dobney - Production Supervisor}
   ▷ [System.Data.Entity.DynamicProxies.Employee_9E79078D2C047A6B {JoLynn Dobney - Production Supervisor}
   ▷ Address                                                   {System.Data.Entity.DynamicProxies.Address_1
     AddressID                                                 275
   ▷ Department                                                {Production}
     DepartmentID                                              7
   ▷ Departments                                               Count = 0
   ▷ Employee1                                                 {Peter Krebs - Production Control Manager}
     EmployeeID                                                7
   ▷ Employees1                                                Count = 6
     FirstName                                                 "JoLynn"
   ▷ HireDate                                                  {26/01/2000 00:00:00}
     JobTitle                                                  "Production Supervisor"
     LastName                                                  "Dobney"
     ManagerID                                                 21
     MiddleName                                                "M"
   ▷ Projects                                                  Count = 4
     Salary                                                    25000
▷ [1]                                                          {Taylor Maxwell - Production Supervisor}
▷ [2]                                                          {Jo Brown - Production Supervisor}
▷ [3]                                                          {John Campbell - Production Supervisor}
▷ [4]                                                          {Zheng Mu - Production Supervisor}
▷ [5]                                                          {Jinghao Liu - Production Supervisor}
▷ [6]                                                          {Reuben D'sa - Production Supervisor}
▷ [7]                                                          {Cristian Petculescu - Production Supervisor}
▷ [8]                                                          {Kok-Ho Loh - Production Supervisor}
▷ [9]                                                          {David Hamilton - Production Supervisor}
▷ [10]                                                         {Eric Gubbels - Production Supervisor}
▷ [11]                                                         {Jeff Hay - Production Supervisor}
▷ [12]                                                         {Cynthia Randall - Production Supervisor}
▷ [13]                                                         {Yuhong Li - Production Supervisor}
▷ [14]                                                         {Shane Kim - Production Supervisor}
```



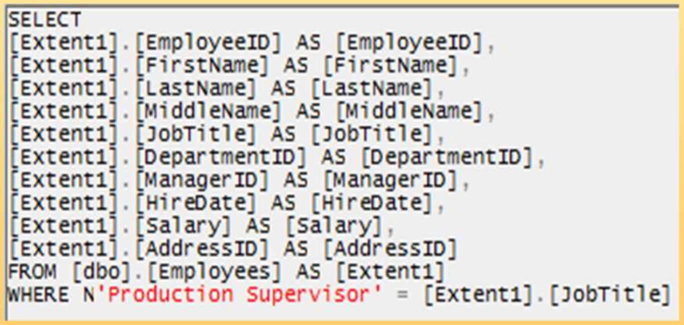
```csharp
private void ChangeBlogPostName(int id,
    string newName)
{
    var db = new Context();

    var post = db.Posts
        .FirstOrDefault(x => x.Id == id);

    if (post == null)
    {
        throw new ArgumentException(
            "Item with that id was not fo
            "id");
    }

    post.Name = newName;

    db.SaveChanges();
}
```



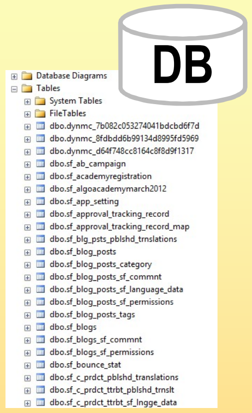
```sql
SELECT
[Extent1].[EmployeeID] AS [EmployeeID],
[Extent1].[FirstName] AS [FirstName],
[Extent1].[LastName] AS [LastName],
[Extent1].[MiddleName] AS [MiddleName],
[Extent1].[JobTitle] AS [JobTitle],
[Extent1].[DepartmentID] AS [DepartmentID],
[Extent1].[ManagerID] AS [ManagerID],
[Extent1].[HireDate] AS [HireDate],
[Extent1].[Salary] AS [Salary],
[Extent1].[AddressID] AS [AddressID]
FROM [dbo].[Employees] AS [Extent1]
WHERE N'Production Supervisor' = [Extent1].[JobTitle]
```

EF: Development Approaches

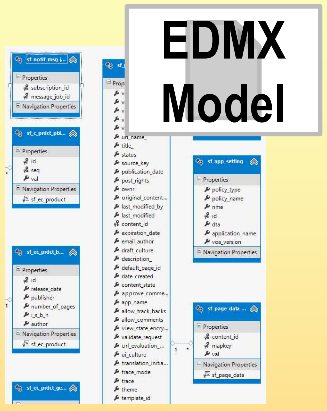
- Database First
- Model First
- Code First

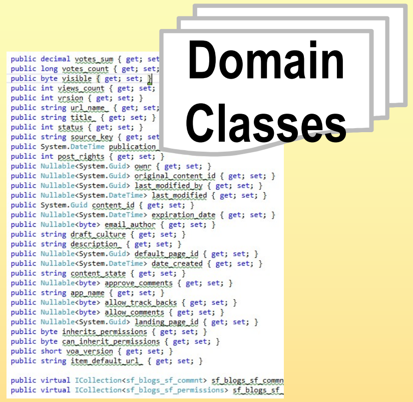
EF: DB First Approach

- If you already have a database, the Entity Framework automatically generate a data model that consists of classes and properties that correspond to existing database objects such as tables and columns.



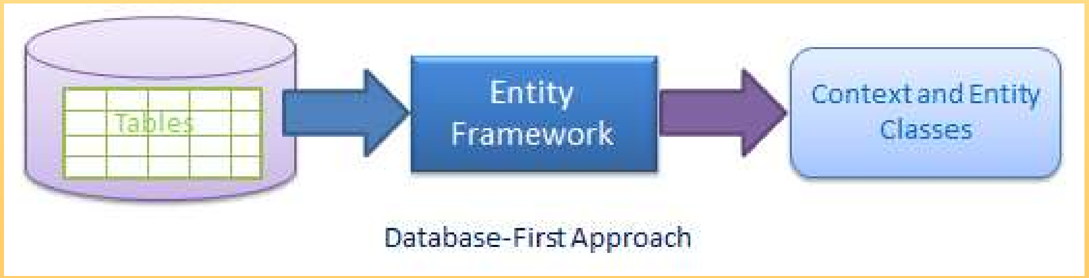
```
Database Diagrams
Tables
  System Tables
  FileTables
  dbo.dynmc_7b082c053274041bdcbd6f7d
  dbo.dynmc_8fdbdd6b99134d8995fd5969
  dbo.dynmc_d64f748cc8164c8f8d9f1317
  dbo.sf_ab_campaign
  dbo.sf_academyregistration
  dbo.sf_algoacademymarch2012
  dbo.sf_app_setting
  dbo.sf_approval_tracking_record
  dbo.sf_approval_tracking_record_map
  dbo.sf_blg_psts_pblshd_trnslations
  dbo.sf_blog_posts
  dbo.sf_blog_posts_category
  dbo.sf_blog_posts_sf_commnt
  dbo.sf_blog_posts_sf_language_data
  dbo.sf_blog_posts_sf_permissions
  dbo.sf_blog_posts_tags
  dbo.sf_blogs
  dbo.sf_blogs_sf_commnt
  dbo.sf_blogs_sf_permissions
  dbo.sf_bounce_stat
  dbo.sf_c_prdct_pblshd_translations
  dbo.sf_c_prdct_ttrbt_pblshd_trnslt
  dbo.sf_c_prdct_ttrbt_sf_lngge_data
```





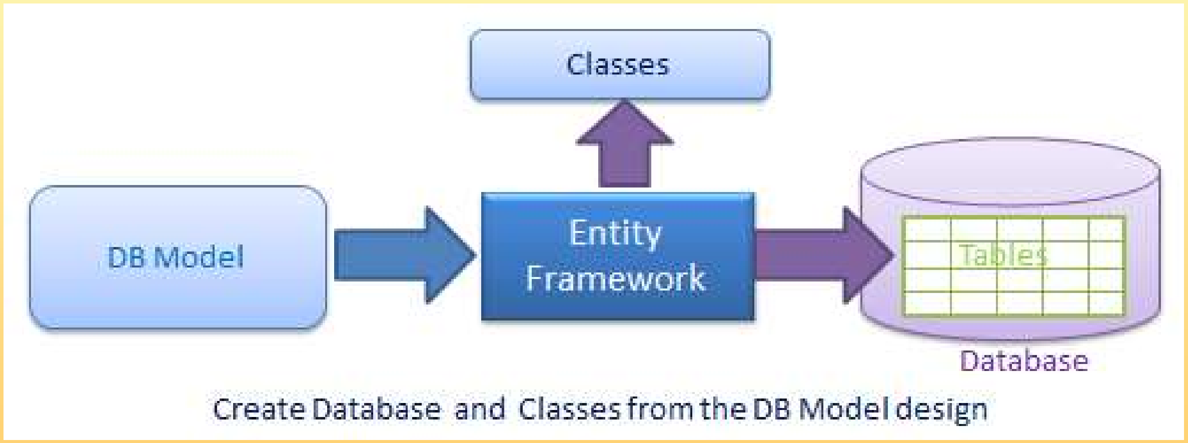
```
public decimal votes_sum { get; set
public long votes_count { get; set;
public byte visible { get; set; }
public int views_count { get; set;
public int vrsion { get; set; }
public string url_name_ { get; set;
public string title_ { get; set; }
public int status { get; set; }
public string source_key { get; set
public System.DateTime publication_
public int post_rights { get; set; }
public Nullable<System.Guid> ownr { get; set; }
public Nullable<System.Guid> original_content_id { get; set; }
public Nullable<System.Guid> last_modified_by { get; set; }
public Nullable<System.DateTime> last_modified { get; set; }
public System.Guid content_id { get; set; }
public Nullable<System.DateTime> expiration_date { get; set; }
public Nullable<byte> email_author { get; set; }
public string draft_culture { get; set; }
public string description_ { get; set; }
public Nullable<System.Guid> default_page_id { get; set; }
public Nullable<System.DateTime> date_created { get; set; }
public string content_state { get; set; }
public Nullable<byte> approve_comments { get; set; }
public string app_name { get; set; }
public Nullable<byte> allow_track_backs { get; set; }
public Nullable<byte> allow_comments { get; set; }
public Nullable<System.Guid> landing_page_id { get; set; }
public byte inherits_permissions { get; set; }
public byte can_inherit_permissions { get; set; }
public short voa_version { get; set; }
public string item_default_url_ { get; set; }

public virtual ICollection<sf_blogs_sf_commnt> sf_blogs_sf_commn
public virtual ICollection<sf_blogs_sf_permissions> sf_blogs_sf_
```



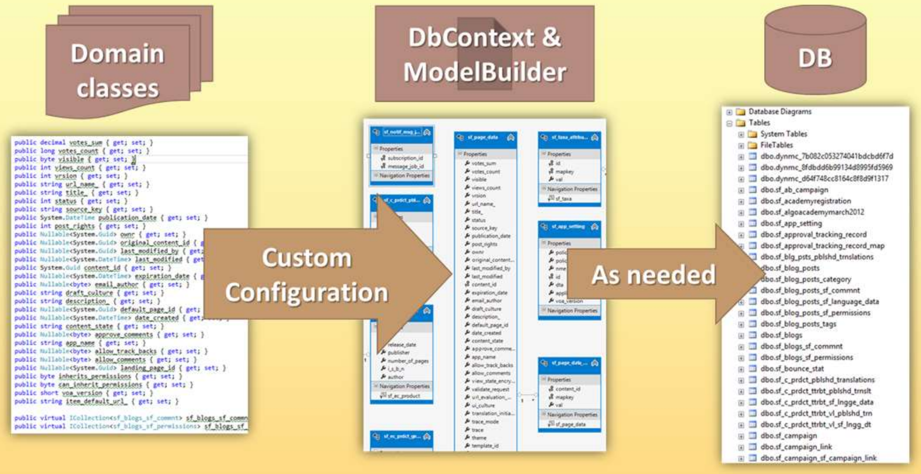
EF: Model First Approach

- **If you don't yet have a database, you can begin by creating a model using the Entity Framework designer in Visual Studio.**
  - The designer can generate DDL statements to create the database.



EF: Code First Approach

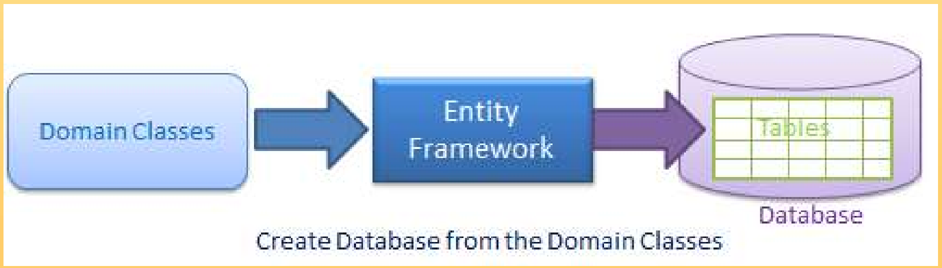
- Whether you have an existing database or not, you can code your own classes and properties that correspond to tables and columns and use them with EF without an .edmx file.



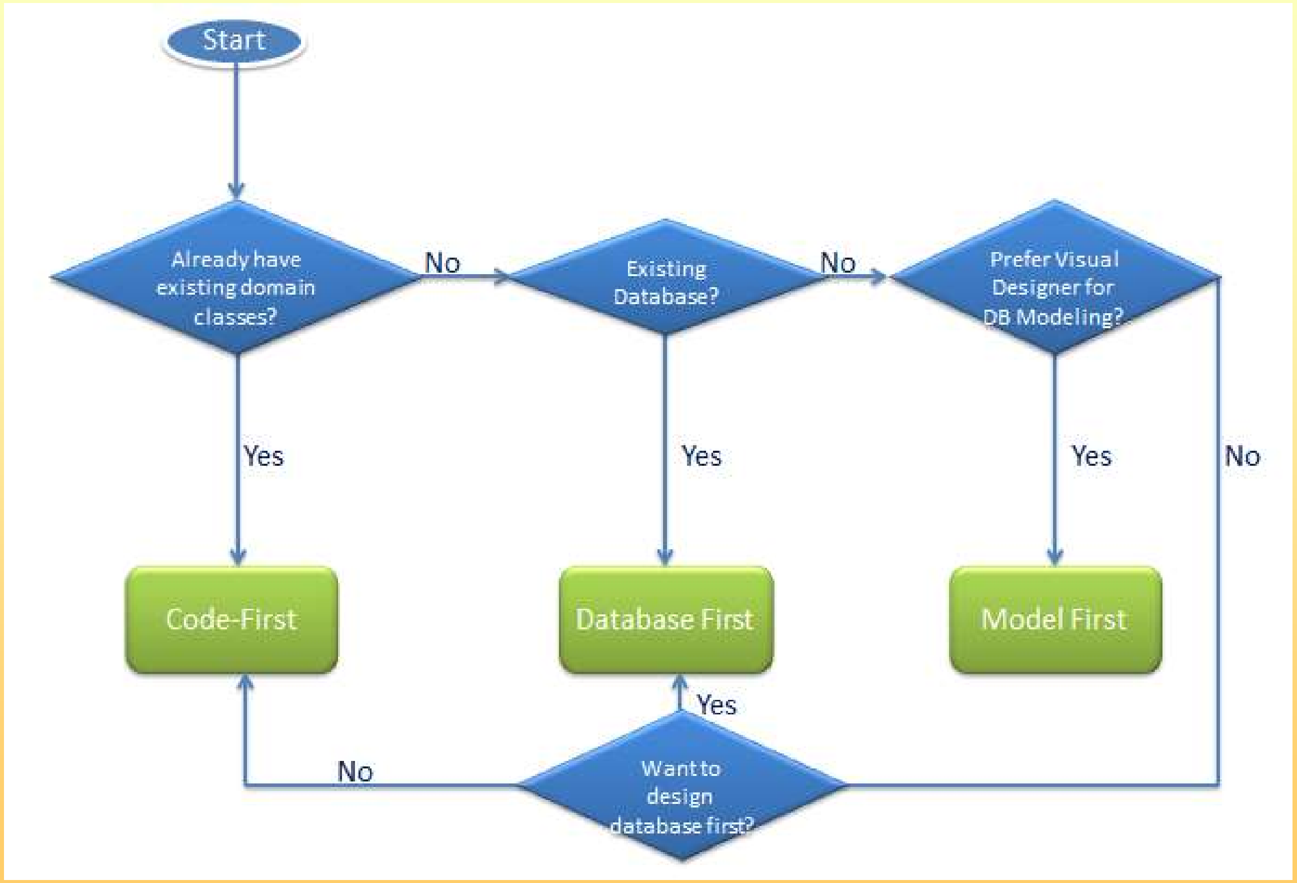
```
public decimal votes_sum { get; set; }
public long votes_count { get; set; }
public byte visible { get; set; }
public int views_count { get; set; }
public int vrsion { get; set; }
public string url_name_ { get; set; }
public string title_ { get; set; }
public int status { get; set; }
public string source_key { get; set; }
public System.DateTime publication_date { get; set; }
public int post_rights { get; set; }
public Nullable<System.Guid> ownr { get; set; }
public Nullable<System.Guid> original_content_id { g
public Nullable<System.Guid> last_modified_by { get;
public Nullable<System.DateTime> last_modified { get
public System.Guid content_id { get; set; }
public Nullable<System.DateTime> expiration_date { g
public Nullable<byte> email_author { get; set; }
public string draft_culture { get; set; }
public string description_ { get; set; }
public Nullable<System.Guid> default_page_id { get;
public Nullable<System.DateTime> date_created { get;
public string content_state { get; set; }
public Nullable<byte> approve_comments { get; set; }
public string app_name { get; set; }
public Nullable<byte> allow_track_backs { get; set; }
public Nullable<byte> allow_comments { get; set; }
public Nullable<System.Guid> landing_page_id { get; set; }
public byte inherits_permissions { get; set; }
public byte can_inherit_permissions { get; set; }
public short voa_version { get; set; }
public string item_default_url_ { get; set; }

public virtual ICollection<sf_blogs_sf_commnt> sf_blogs_sf_commn
public virtual ICollection<sf_blogs_sf_permissions> sf_blogs_sf_
```

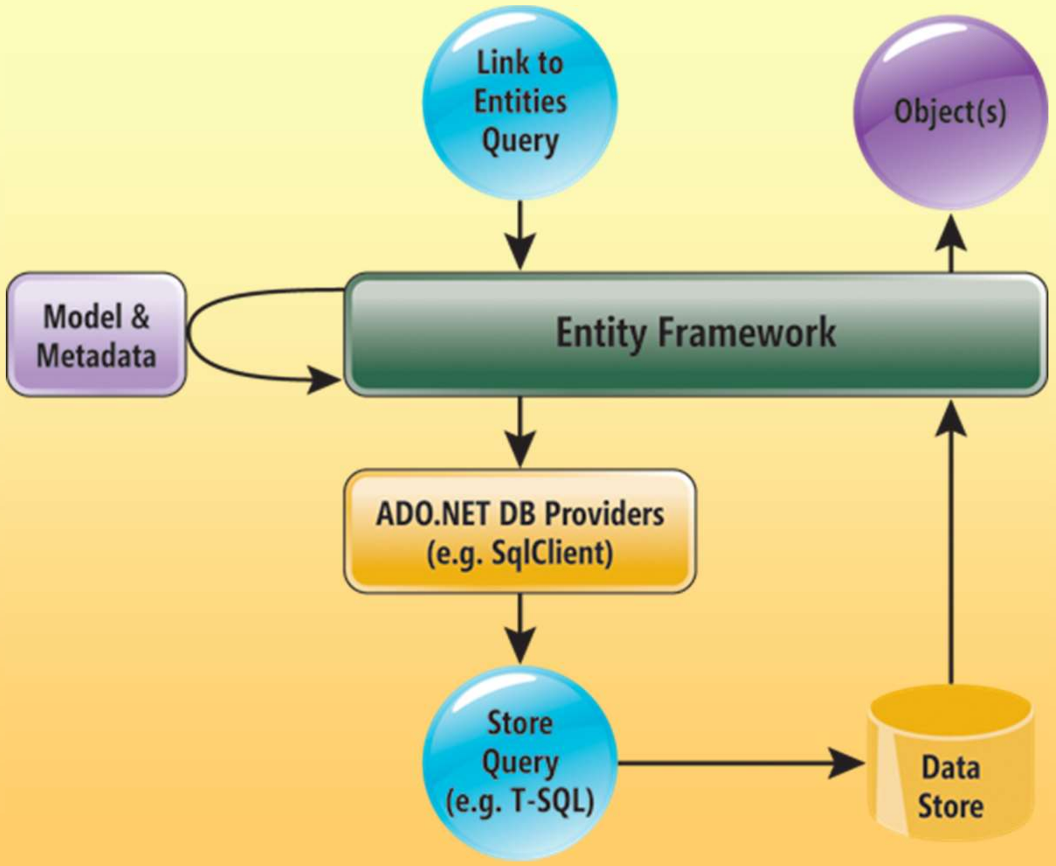
```
Database Diagrams
Tables
    System Tables
    FileTables
    dbo.dynmc_7b082c053274041bdcbd6f7d
    dbo.dynmc_8fdbdd6b99134d8995fd5969
    dbo.dynmc_d64f748cc8164c8f8d9f1317
    dbo.sf_ab_campaign
    dbo.sf_academyregistration
    dbo.sf_algoacademymarch2012
    dbo.sf_app_setting
    dbo.sf_approval_tracking_record
    dbo.sf_approval_tracking_record_map
    dbo.sf_blg_psts_pblshd_trnslations
    dbo.sf_blog_posts
    dbo.sf_blog_posts_category
    dbo.sf_blog_posts_sf_commnt
    dbo.sf_blog_posts_sf_language_data
    dbo.sf_blog_posts_sf_permissions
    dbo.sf_blog_posts_tags
    dbo.sf_blogs
    dbo.sf_blogs_sf_commnt
    dbo.sf_blogs_sf_permissions
    dbo.sf_bounce_stat
    dbo.sf_c_prdct_pblshd_translations
    dbo.sf_c_prdct_ttrbt_pblshd_trnslt
    dbo.sf_c_prdct_ttrbt_sf_lngge_data
    dbo.sf_c_prdct_ttrbt_vl_pblshd_trn
    dbo.sf_c_prdct_ttrbt_vl_sf_lngg_dt
    dbo.sf_campaign
    dbo.sf_campaign_link
    dbo.sf_campaign_sf_campaign_link
```



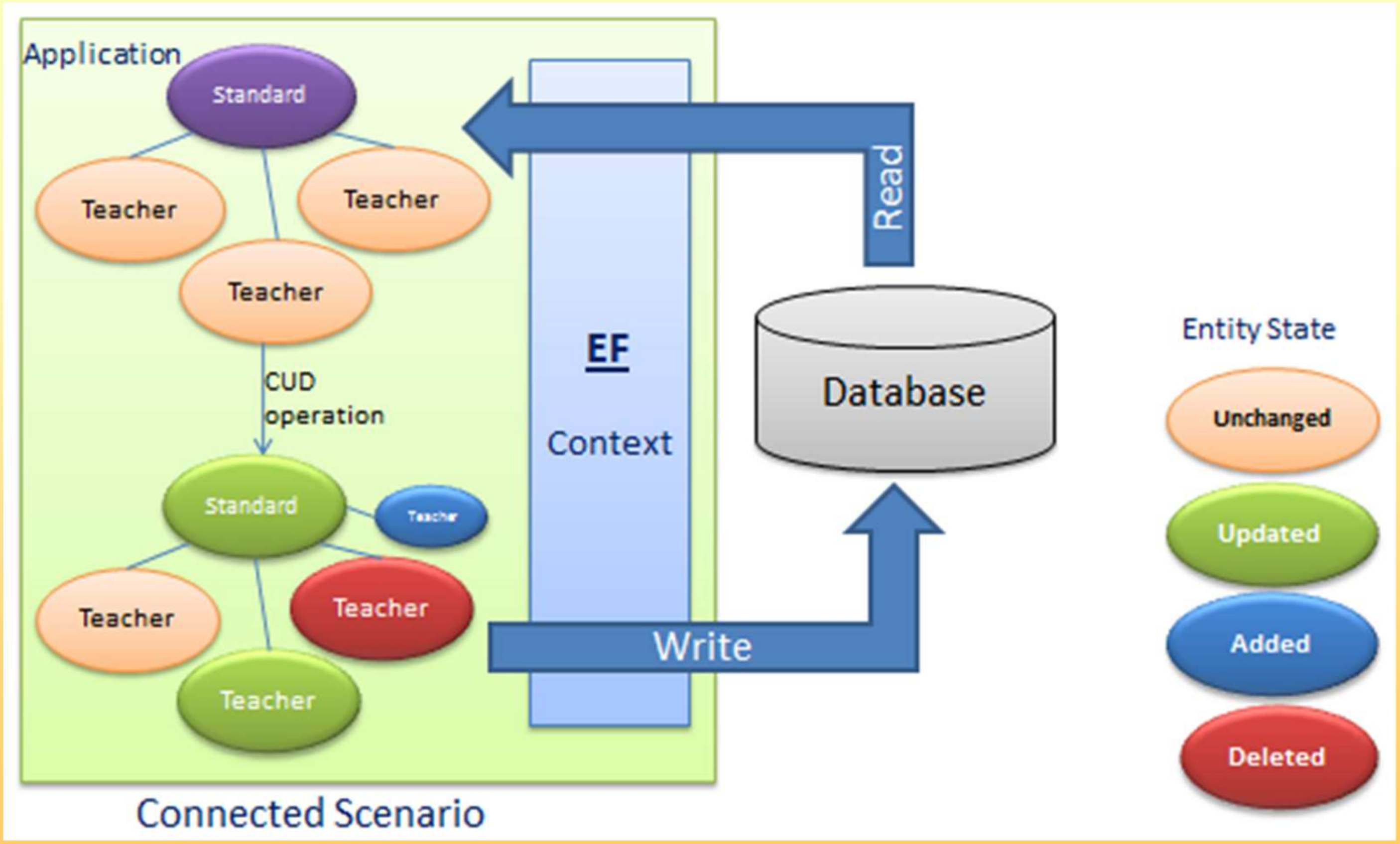
EF: How to Choose Development Approach?


Start
Already have existing domain classes?
No
Existing Database?
No
Prefer Visual Designer for DB Modeling?
Yes
Yes
Yes
No
Code-First
Database First
Model First
Yes
No
Want to design database first?

EF: Retrieve Data - How it works?

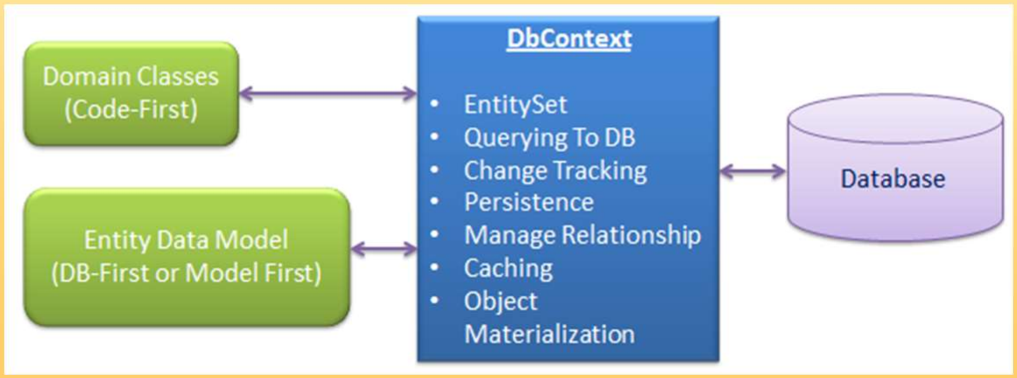


EF: Modify Data – How it works?



Entity Framework – Components

- **The `DbContext` class**
  - **`DbContext`** holds the DB connection
  - Holds the **`DbSet<T>`** for the entity classes
  - Provides LINQ-based data access (through **`IQueryable`**)
  - Provides API for CRUD operations
- **Entity classes**
  - Hold entities (objects with their attributes and relations)
  - Each database table is typically mapped to a single C# entity class



EF: Reading Data with LINQ Query

- We can also use extension methods for constructing the query

```
using (var context = new SampleEntities())
{
  var employees = context.Employees
      .Select(c => c.FirstName)
      .Where(c => c.JobTitle == "Design Engineering")
      .ToList();
}
```

This is called selection

This is called projection

ToList() method executes the SQL query

- Find element by id

```
using (var context = new SampleEntities())
{
  var project = context.Projects.Find(2);
  Console.WriteLine(project.Name);
}
```

EF: Creating New Data

- To create a new database row use the method `Add(…)` of the corresponding collection:

```
var project = new Project()
{
  Name = “CRM Application",
  StartDate = new DateTime(2015, 4, 15)
};

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

Create a new project object

Mark the object for inserting

This will execute an SQL INSERT

- `SaveChanges()` method executes the SQL insert / update / delete commands in the database

EF: Cascading Inserts

- **We can also add cascading entities to the database:**

```
Employee employee = new Employee();
employee.FirstName = "Venkat";
employee.LastName = "Shiva Reddy";
employee.Projects.Add(new Project { Name = "CRM Project" } );
softUniEntities.Employees.Add(employee);
softUniEntities.SaveChanges();
```

- **This way we don't have to add `Project` individually**
  - They will be added when the **`Employee`** entity (employee) is inserted to the database

EF: Updating Existing Data

- **`DbContext` allows modifying entity properties and persisting them in the database**
  - Just load an entity, modify it and call **`SaveChanges()`**
- **The `DbContext` automatically tracks all changes made on its entity objects**

```
Employees employee =
  sampleEntities.Employees.First();

employees.FirstName = "Alex";

context.SaveChanges();
```

This will execute an SQL SELECT to load the first row

This will execute an SQL UPDATE

EF: Deleting Existing Data

- Delete is done by `Remove()` on the specified entity collection
- `SaveChanges()` method performs the delete action in the database

```
Employees employee =
  sampleEntities.Employees.First();

sampleEntities.Employees.Remove(employee);

sampleEntities.SaveChanges();
```

Mark the entity for deleting at the next save

This will execute the SQL DELETE command

EF: Native SQL Queries

```
var context = new SampleEntities();
string nativeSQLQuery =
    "SELECT FirstName + ' ' + LastName " +
    "FROM dbo.Employees WHERE JobTitle = {0}";

var employees = context.Database.SqlQuery<string>(
    nativeSQLQuery, "Marketing Specialist");

foreach (var emp in employees)
{
    Console.WriteLine(emp);
}
```

Parameter placeholder

Return type

Parameter value