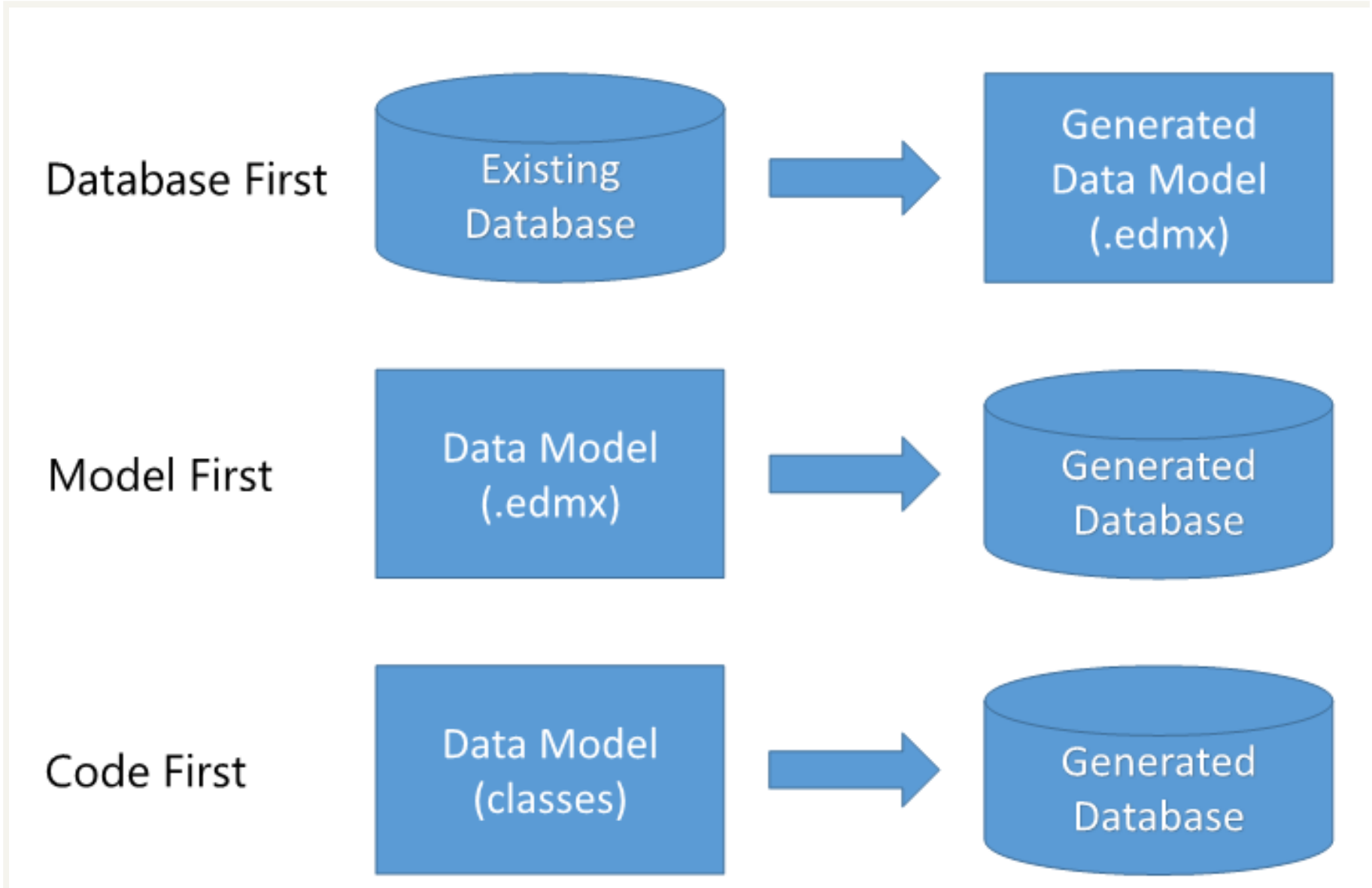
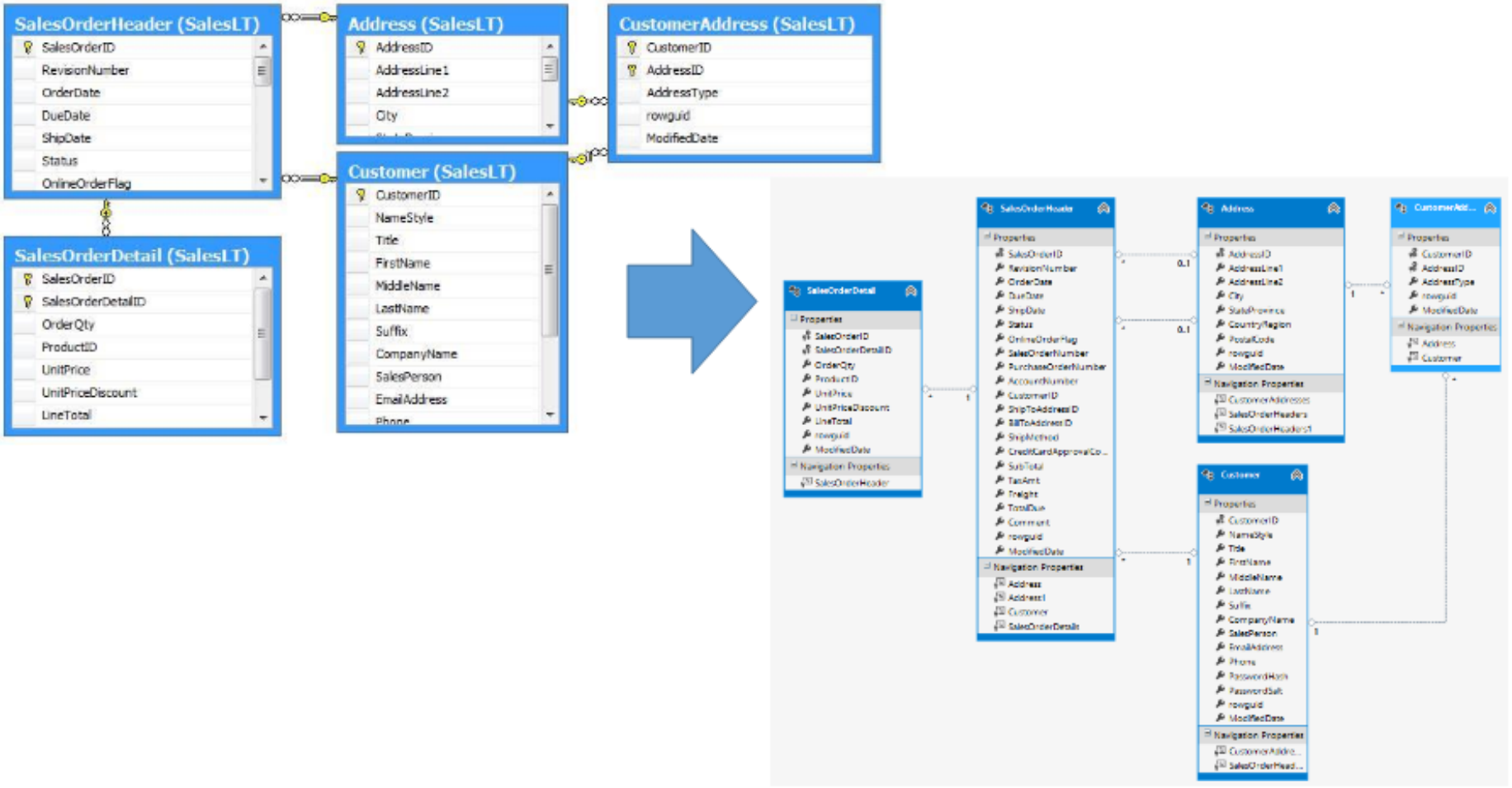
Entity Framework. Code First

Основные способы создания моделей

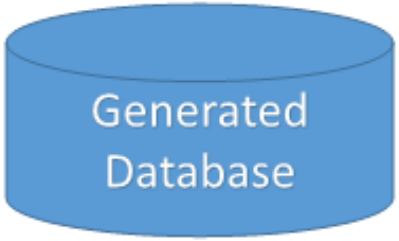


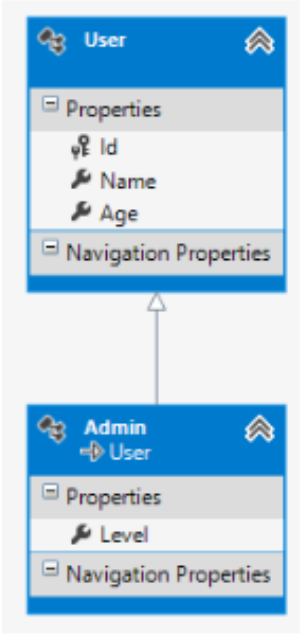
Database First



Model First

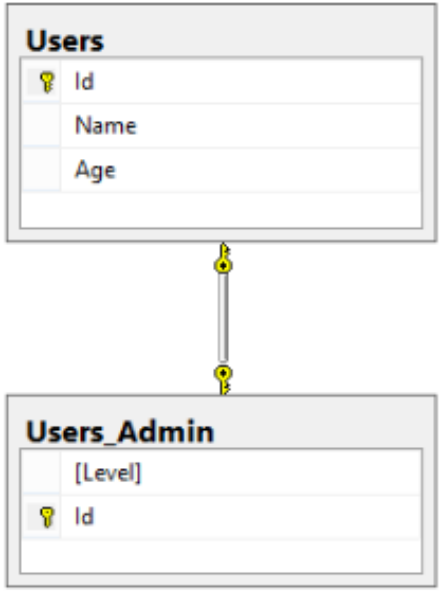
Model First



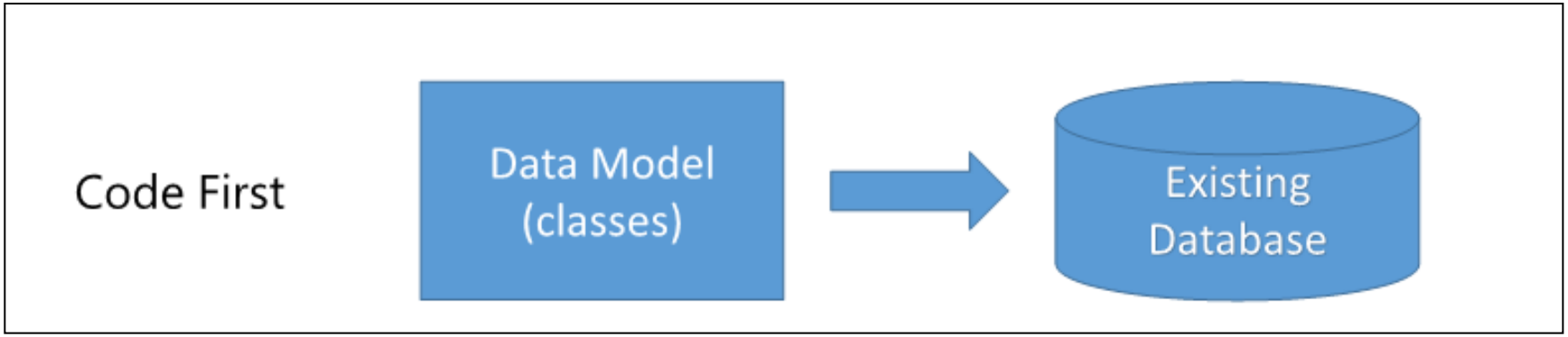


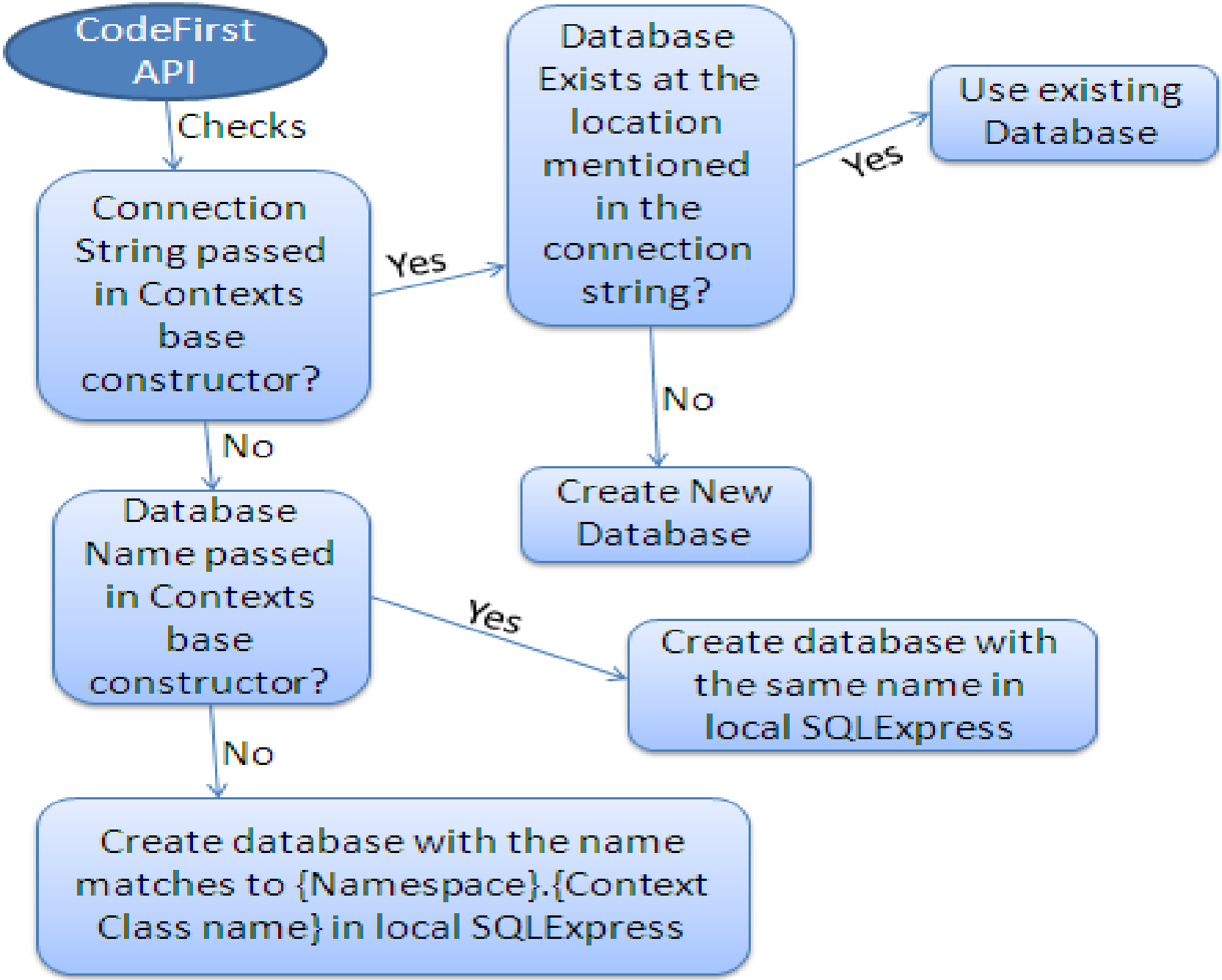

```
public partial class User
 {
      public int Id { get; set; }
      public string Name { get; set; }
      public DateTime? Age { get; set; }
 }

public partial class Admin : User
{
     public string Level { get; set; }
}
```



CodeFirst – написание кода классов предметной области, при отсутствии модели и БД.
Генерация БД и модели сущностей EDM происходит после построения проекта.




CodeFirst API
Checks
Connection String passed in Contexts base constructor?
Yes
Database Exists at the location mentioned in the connection string?
Yes
Use existing Database
No
Create New Database
No
Database Name passed in Contexts base constructor?
Yes
Create database with the same name in local SQLExpress
No
Create database with the name matches to {Namespace}.{Context Class name} in local SQLExpress

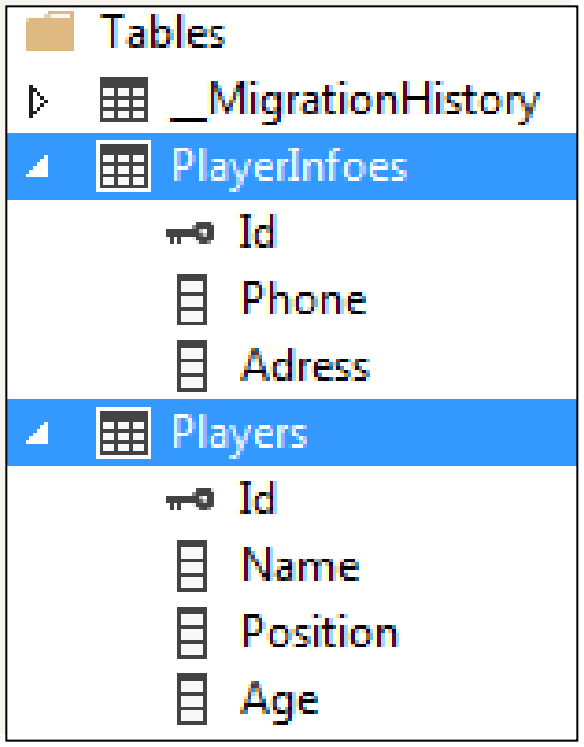
Демонстрация

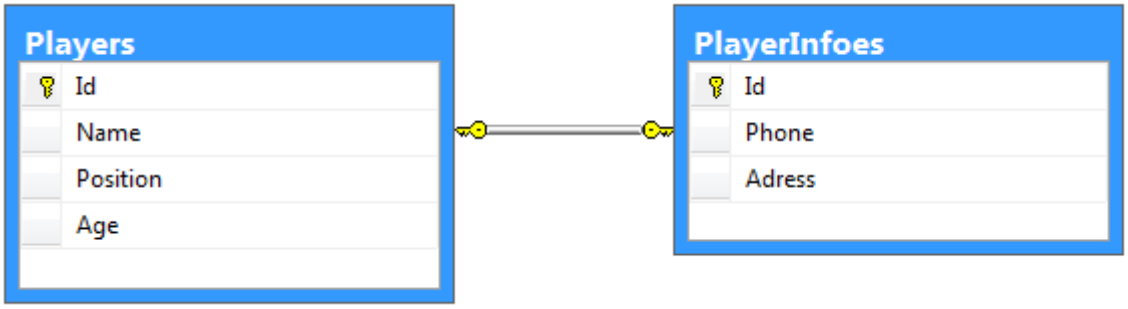
- создание базы с разными параметрами

Связи (отношений)«один к одному»

```
public  class Player
{
 public int Id { get; set; } // внимание!!!
 public string Name { get; set; }
 public string Position { get; set; }
 public int Age { get; set; }
 public PlayerInfo PlayerInfo { get; set; }
}
```

```
public class PlayerInfo
{  [ForeignKey("Player")]
   public int Id { get; set; }
   public string Phone { get; set; }
   public string Adress { get; set; }
   public Player Player { get; set; }
}
```

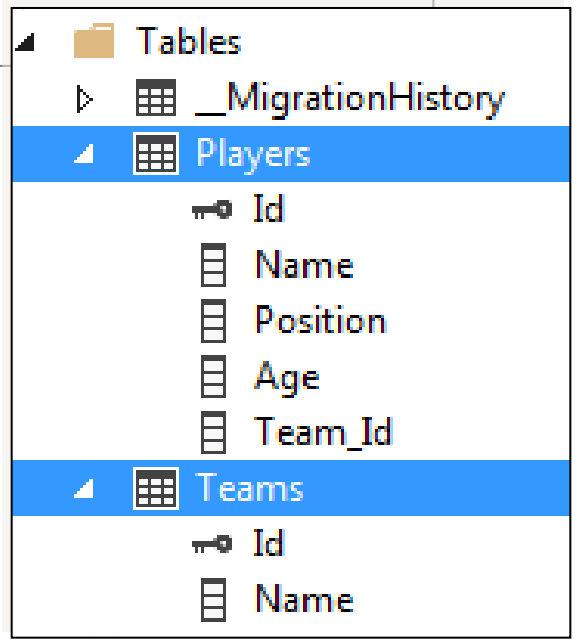


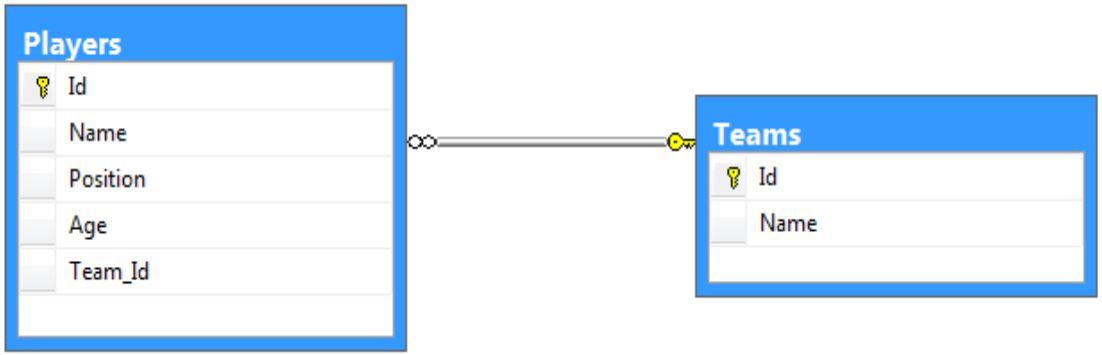


Связи (отношений) «один ко многим»

```
public class Team
{
  public int Id { get; set; }
  public string Name { get; set; }
  public ICollection<Player> Players { get; set; }
}
```

```
public class Player
{
  public int Id { get; set; }
  public string Name { get; set; }
  public string Position { get; set; }
  public int Age { get; set; }
  public Team Team { get; set; }
}
```

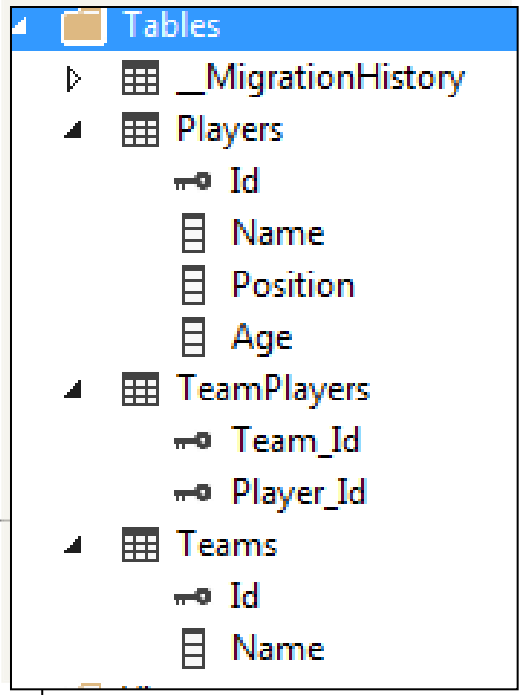


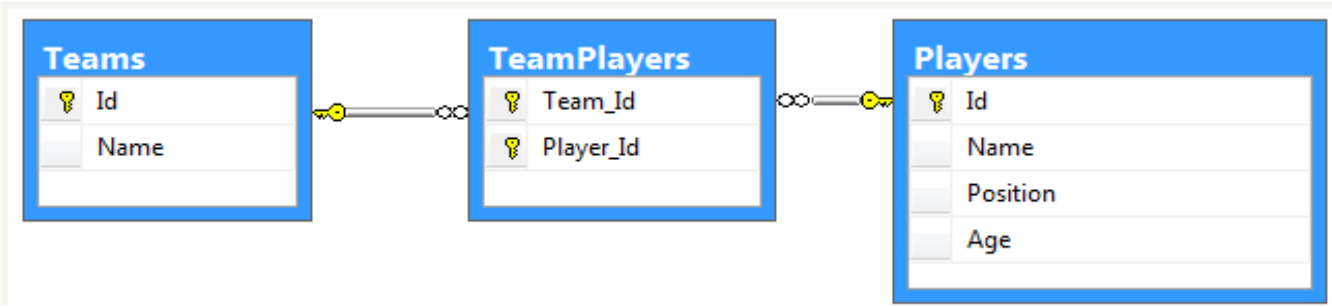


Связи (отношений)«многие ко многим»

```
public class Player
{
  public int Id { get; set; }
  public string Name { get; set; }
  public string Position { get; set; }
  public int Age { get; set; }
  public ICollection<Team> Teams { get; set; }
}
```

```
public class Team
{
  public int Id { get; set; }
  public string Name { get; set; }
  public ICollection<Player>Players { get; set; }
}
```





Демонстрация

- создание связанных таблиц базы

Настройка модели

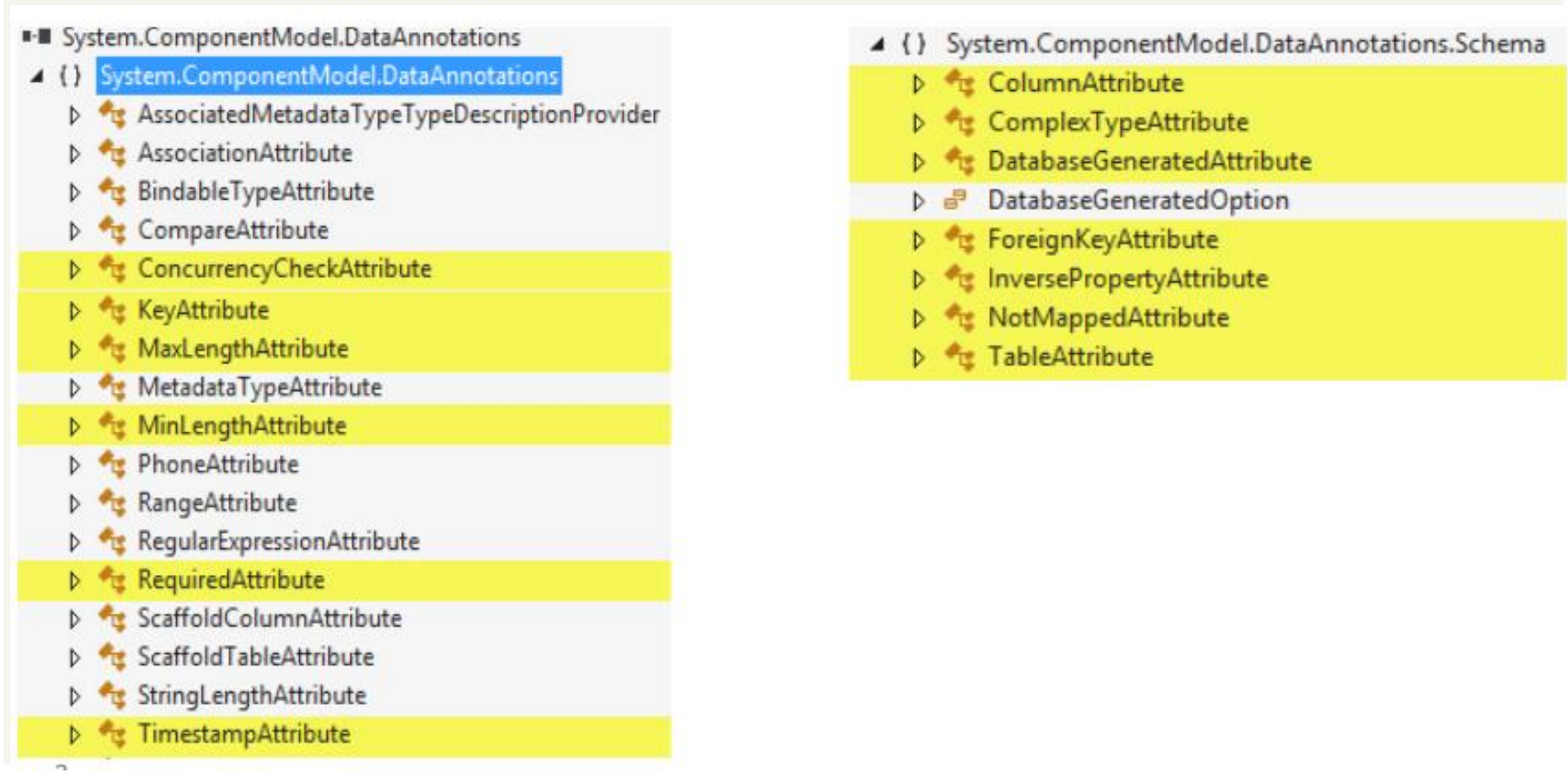
Data Annotations

```
[MaxLength(20), MinLength(5)]
public string Name
```

Fluent API

```
modelBuilder.Entity<Post>()
  .Property(p => p.Title).HasMaxLength(10);
```

Набор аттрибутов Data Annotations



DataAnnotations: KeyAttribute

1. Свойство с именем **Id**

```
public partial class Item
{
    public Guid Id { get; set; }
}
```

1. Свойство с именем **[имя_класса]Id**

```
public partial class Item
{
    public Guid ItemId { get; set; }
}
```

!

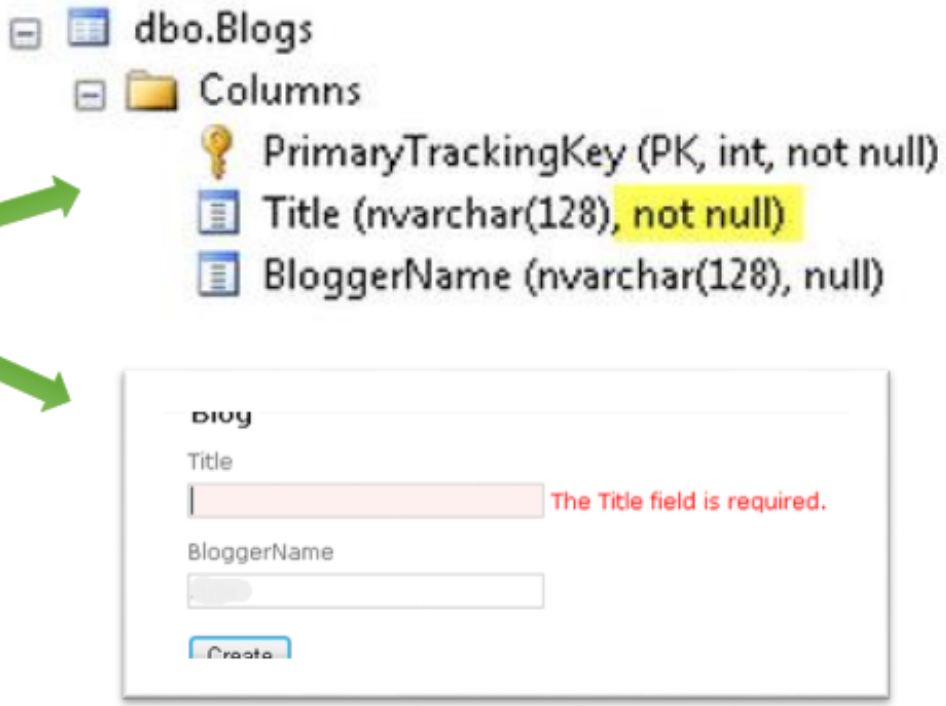
Необходим аттрибут **[Key]**, если имя ключевого свойства произвольно!

```
public partial class Item
{
  [Key]
  public Guid GlobalItemKey { get; set; }
}
```

DataAnnotations: RequiredAttribute

Задание обязательности значения

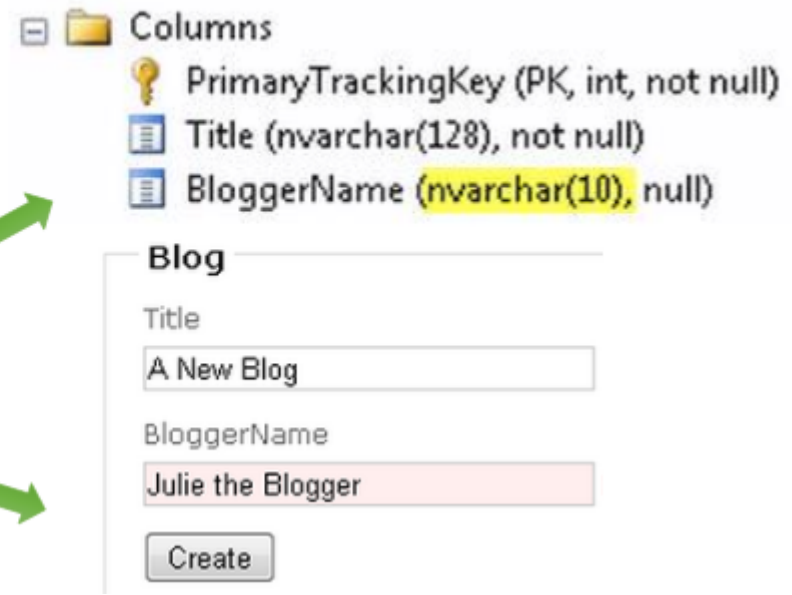
```
public partial class Blog
{
    [Key]
    public int PrimaryTrackingKey { get; set; }
    [Required]
    public string Title { get; set; }
}
```



DataAnnotations: MinLengthи MaxLength

Задание допустимой длинны значения

```
public partial class Blog
{
    [Key]
    public int PrimaryTrackingKey { get; set; }
    [Required]
    public string Title { get; set; }
    [MinLength(5), MaxLength(10)]
    public string BloggerName { get; set; }
}
```



DataAnnotations: NotMappedAttribute

Вычисляемое свойство

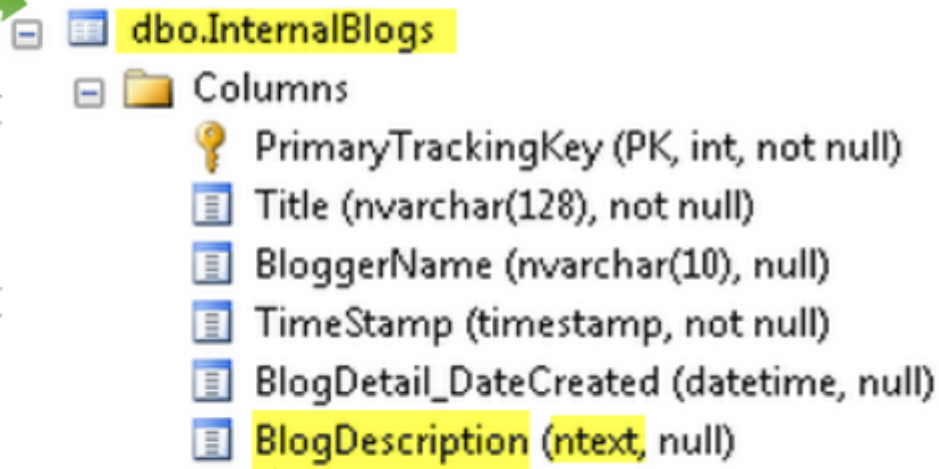
```
public partial class Blog
{
   [Key]
    public int PrimaryTrackingKey { get; set; }
    [Required]
    public string Title { get; set; }
    [MinLength(5), MaxLength(10)]
    public string BloggerName { get; set; }
    [NotMapped]
    public int BlogCode
         {
           get
            {
              return Title.Substring(0, 1) + ":" + BloggerName.Substring(0, 1);
            }
         }
}
```

Поле не сохраняется в БД.

DataAnnotations: TableAttribute и ColumnAttribute

Переименование таблиц/колонок

```
[Table("InternalBlogs")]
public partial class Blog
{
   [Key]
   public int PrimaryTrackingKey { get; set; }
   [Required]
   public string Title { get; set; }
   [MinLength(5), MaxLength(10)]
   public string BloggerName { get; set; }
   public BlogDetails BlogDetail { get; set; }
}
[ComplexType]
public partial class BlogDetails
{
   public DateTime? DateCreated { get; set; }
   [Column("BlogDescription", TypeName="ntext")]
   public string Description { get; set; }
}
```



FluentAPI

```
protected override void OnModelCreating
                                        (DbModelBuilder modelBuilder)
{
     base.OnModelCreating(modelBuilder);
    // конфигурация модели с применением Fluent Api
    modelBuilder.Configurations.Add(new TeamEntityTypeConfig());
    modelBuilder.Configurations.Add(new PlayerEntityTypeConfig());
}
```

```
public class PlayerEntityTypeConfig :
                                        EntityTypeConfiguration<Player>
{
  public PlayerEntityTypeConfig()
  {
     ToTable("TeamPlayers");
     HasKey(p => p.Id);
     Property(p => p.Name).IsRequired().HasMaxLength(20);
     Property(p => p.Position).IsRequired().HasColumnName("Post");

  }
}
```