在Entity Framework 中修改数据

北京理工大学计算机学院
金旭亮

什么是CRUD？

C	• Create： 增
R	• Read： 查
U	• Update： 改
D	• Delete： 删

Entity Framework如何实现CRUD？

查询（R）

- LINQ to Entities
- 标准查询扩展方法+Lambda表达式

新增（C）

- new一个实体对象
- 加入到对应的DbSet中

删除（D）

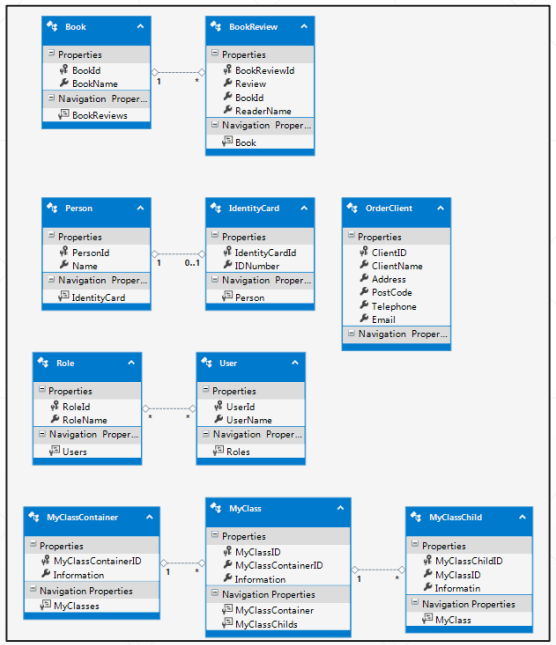
- 定位要删除的实体对象
- 从DbSet中移除

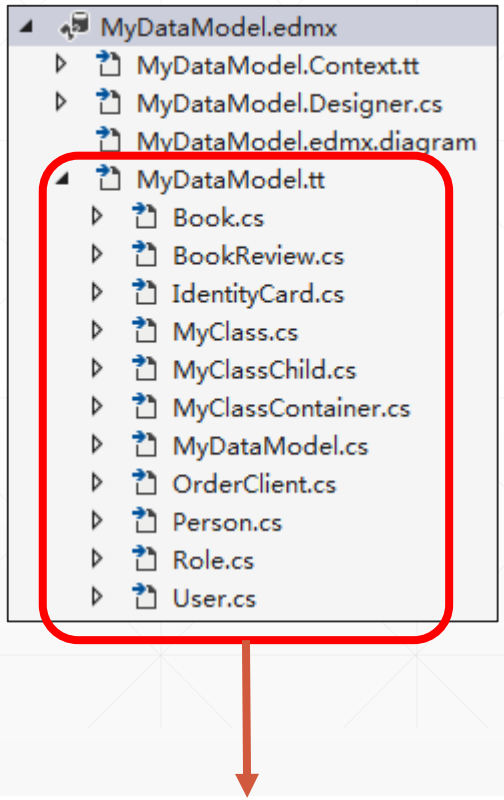
修改（U）

- 定位要修改的实体对象
- 直接设置其属性

DbContext.SaveChanges()

Database First开发模式的致命弱点





生成的数据实体模型无法直接修改!

如果需要修改数据实体类

Entity Framework生成的实体类都是“**分部的（partial）**”，因此，可以编写独立的分部类，以修改EF自动生成的实体类。

例如：以下代码为OrderClient实体类重写了ToString()方法：

```
public partial class OrderClient    {

    public override string ToString()
    {
        return String.Format("{0}:{1}",ClientName,AddressStr);
    }
}
```

除了可以重写基类方法，你甚至还可以在分部类中实现新的接口，添加新的方法和属性……

实现Add功能的典型代码：

```
//1. 创建一个新的数据实体对象
OrderClient client = new OrderClient()
                {
                    AddressStr = …,
                    ClientName = …,
                    PostCode = "10081"
                };

//2.追加到DbSet中
context.OrderClients.Add(client);

//3.保存到数据库
context.SaveChanges();
```

新建对象时,无需指定主键字段ClientID的值。

当对象插入DB之后，client对象的ClientID属性会自动获取到新值。

单元测试方法：TestAdd()

实现Modify功能的典型代码

先找着对象，再直接修改其属性，之后saveChanges()即可。

```
//1. 查找定位要修改的对象
OrderClient firstClient = context.OrderClients.First();

//2. 用新值取代老值
firstClient.ClientName = "modified Client Name";
firstClient.Address= "modified Address";

//3. 保存到数据库中
context.SaveChanges();
```

单元测试方法：TestModify()

实现Delete功能的典型代码

先找着它,再Remove,最后**SaveChanges()**

```
//1. 查找定位要删除的对象
OrderClient firstClient = context.OrderClients.First();

//2. 从DbSet中移除这个对象
context.OrderClients.Remove(firstClient);

//3. 保存到数据库中
context.SaveChanges();
```

要删除一个对象，应该调用DbSet的**Remove()**方法。

单元测试方法：TestDelete()

直接删除对象的便捷方法-1

如果已经知道要删除对象的主键值，那么可以通过Attach一个“样板”对象，再删除之：

```
OrderClient stubClient=new OrderClient{ ClientID = id };
context.OrderClients.Attach(stubClient);
context.OrderClients.Remove(stubClient);
context.SaveChanges();
```

这样做的好处在于能直接删除一个对象，而不需要先从数据库中提取数据，创建实体对象，再查找并删除之，从而能有效地提升数据处理性能。

直接删除对象的便捷方法-2

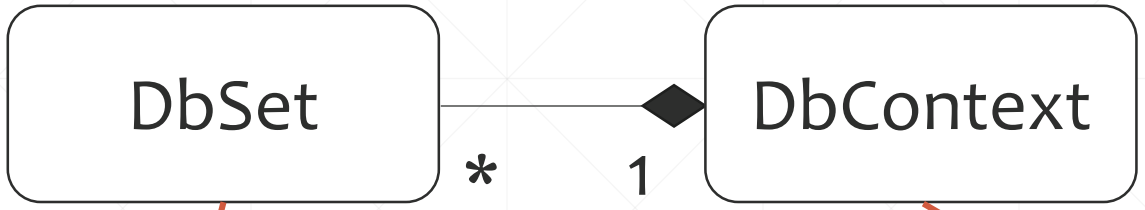
更“武断”的方法，直接向数据库发送SQL命令，在设置了“级联删除”特性之后，只需发送一条SQL命令，就能删除所有相关联的记录，效率极高：

```
context.Database.ExecuteSqlCommand(
          "DELETE  FROM …… where id={0} ", id);
```

注意：这种方式使用要慎重！

探索Entity Framework数据更新原理

理解两个核心类型的职责

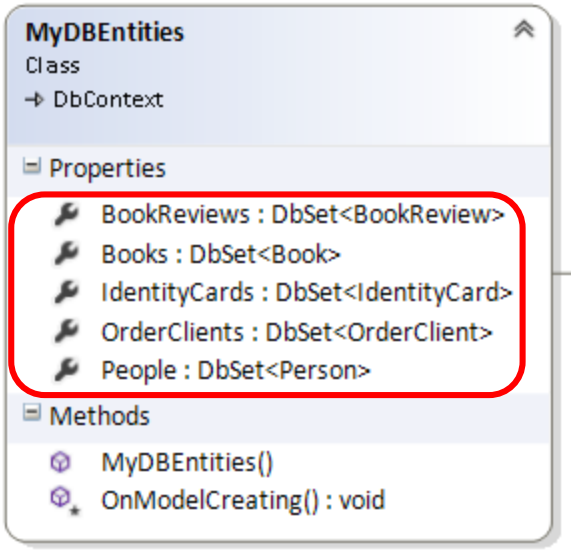


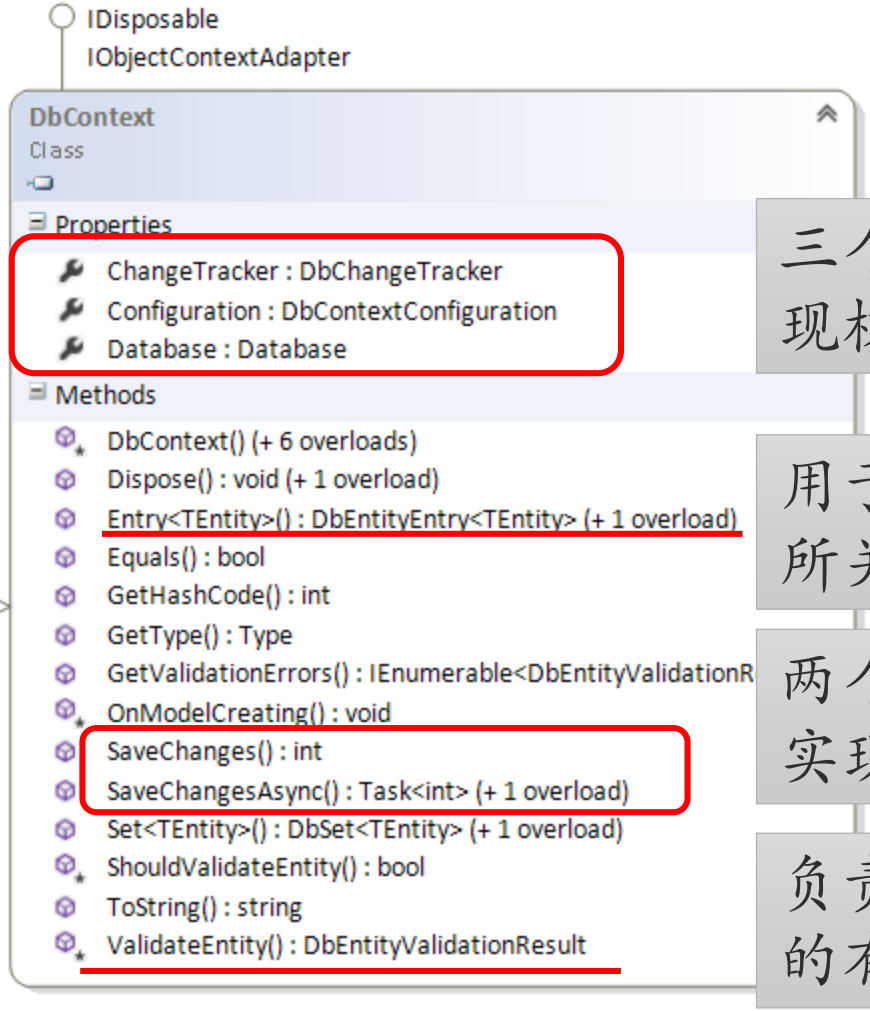
DbSet，是实体对象的集合，提供了实现CRUD的相应方法。

封装与数据库和数据模型相关的功能，依据数据实体状态创建SQL命令，将数据更改保存到数据库中。

CRUD功能的核心对象——DBContext

自定义DbContext类的子类，包容数据库应用程序中用到的所有数据实体对象集合





三个重要属性，实现核心功能

用于提取实体对象所关联的状态对象

两个重载的方法，实现数据的保存

负责检测实体数据的有效性

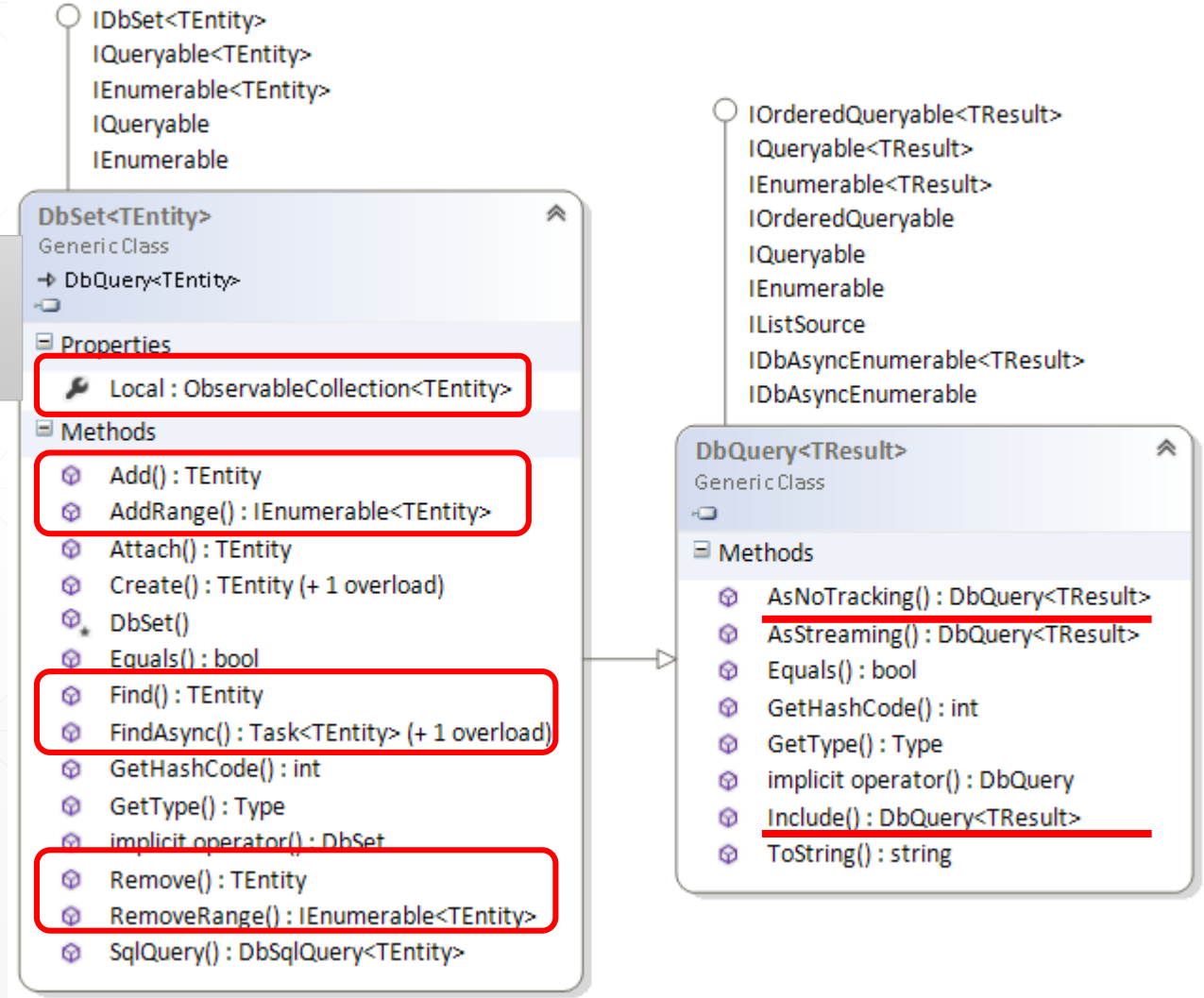
DbSet包容实现CRUD的相关方法

引用本地缓存的实体数据对象集合

新增实体对象

查找实体对象

移除实体对象



禁用状态跟踪

显式加载关联数据

数据实体对象的状态

在程序运行时，数据实体（Data Entity）总处于以下状态之一

```
public enum EntityState
{
        Detached = 1,       //状态未被DbContext跟踪
        Unchanged = 2,      //未改变
        Added = 4,          //是新加的数据实体
        Deleted = 8,        //已被删除
        Modified = 16       //已被修改
}
```

EF数据更新原理

EF使用一个**Change Tracker**对象来跟踪用户操作.

当EF从查询结果中提取数据创建实体类时，它会同步创建另一个对应的**DbEntityEntry**对象，加入到DbChangeTracker对象的**Entries**集合中。

在需要时，我们可以使用**DbContext.Entry(entity)**方法获取entity所对应的状态对象，从而了解对象当前所处的状态

实体对象状态的确定

DbEntityEntry管理的三个值

- **database value:** 代表数据库中的值，此值有可能会被其他用户所改变，调用**DbEntityEntry.GetDatabaseValues()**方法可以获取此值
- **original value:** 代表从数据库中Load之后，对象属性的初始值
- **current value**：代表对象属性的当前值

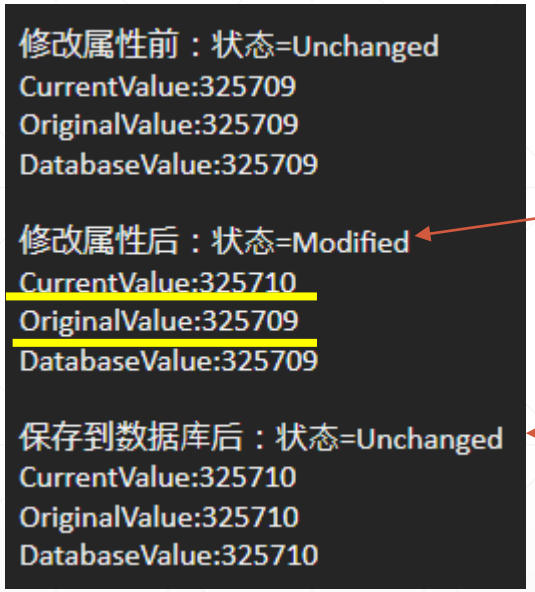
Change Tracker对象通过比对**CurrentValues**和**OriginalValues**即可确定实体对象状态，并设置DbEntityEntry.**State**属性为合适的值。

在合适的时机，DbContext. ChangeTracker对象检查对象属性值的更改或DbSet对象集合中对象个数的变化，负责同步更新对应的**DbEntityEntry**对象。

引发实体状态自动更新的场景

- DbSet.Add()
- DbSet.Find()
- DbSet.Remove()
- 存取DbSet.Local引用的数据缓存
- DbContext.SaveChanges()
- 针对DbSet执行任何一个LINQ查询
- DbSet.Attach()
- DbContext.GetValidationErrors()
- DbContext.Entry()
- 访问DbChangeTracker.Entries集合

动手试验实体状态的改变

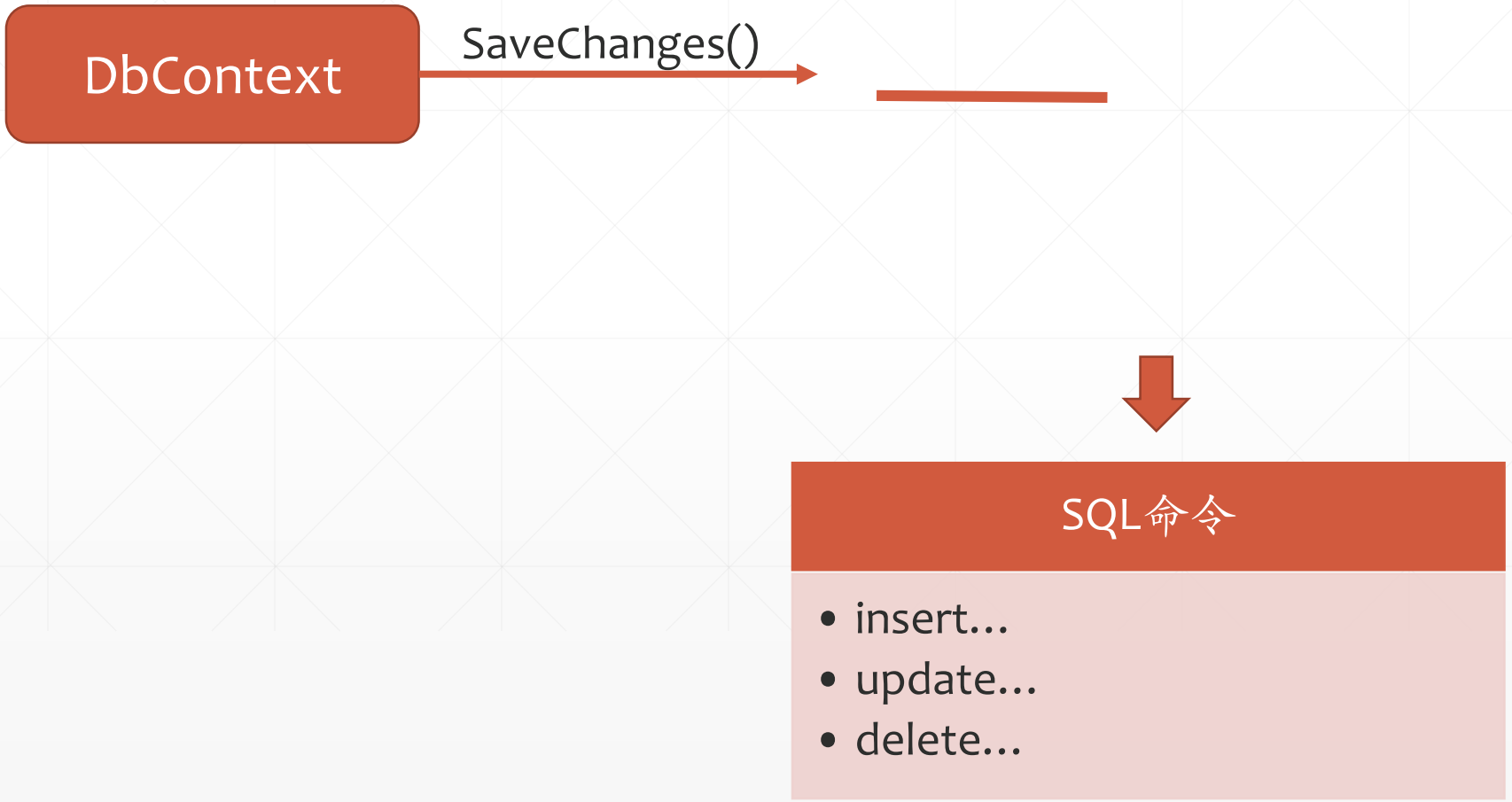


通过比对 CurrentValue 和 OriginalValue 的值，Entity Framework可以知道实体处于Modifed状态。

SaveChange()成功之后，又变成UnChanged状态。

单元测试方法：TestEntityState()

SQL命令的生成



禁用状态跟踪

如果数据是只读的，那么，可以禁用状态跟踪以获取较优的性能

```
var client = (from c in context.OrderClients.AsNoTracking()
                        orderby c.ClientID descending
                        select c).FirstOrDefault ();
……
```

单元测试方法：TestDisableStateTracking()

AsNoTracking()的影响

状态跟踪被禁用之后

- 对应的数据实体状态为**Detached**
- 不再允许访问**CurrentValues**和**OriginalValues**两个属性（硬要访问，将抛出**InvalidOperationException**异常）
- DbContext.**SaveChanges**()方法会因为检测不到数据变更，不会生成任何的SQL命令发给数据库

关联数据的CRUD

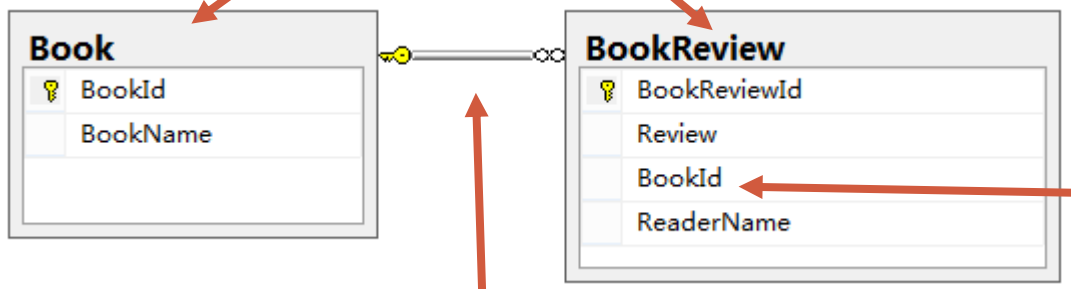
关于“一对一关联”

在使用EF开发数据库应用程序时，一对一关联的多个表会带来很多麻烦，不建议使用，因此，我们跳过对它的介绍，直接讨论用得最多的“一对多关联”。

一对多关联

一对多关联数据库设计规范

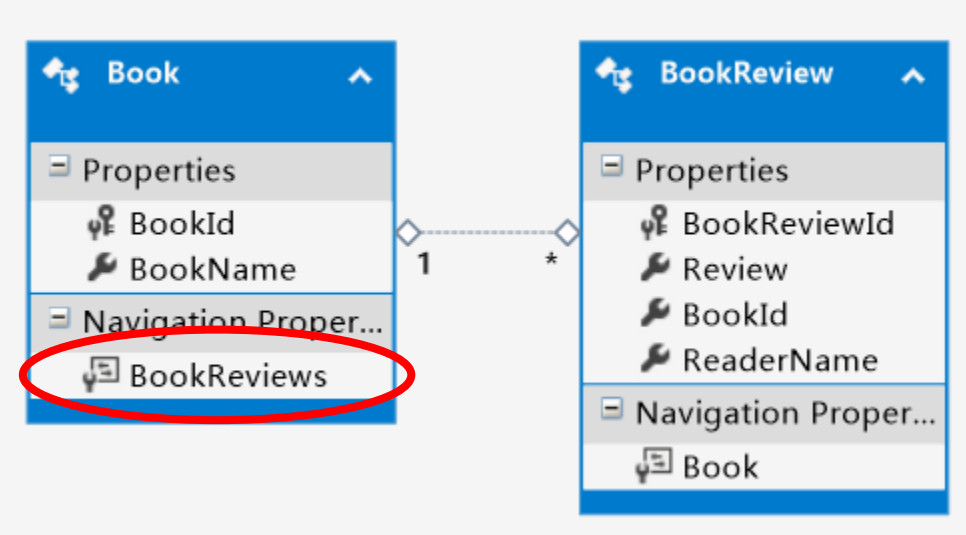
两个表都必须有主键，且都应该设置为**自增，从表外键字段不允许为空**



EF使用DatabaseFirst方式导入数据实体模型时，一定要选中“包容外键”这个属性

如果可以，尽量给关联添加“**级联删除**”特性

C：Create

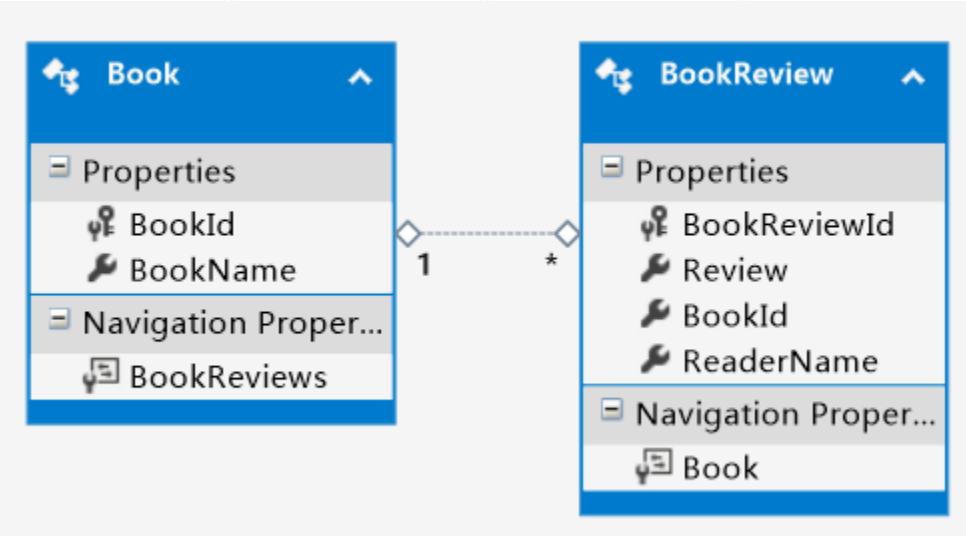


一对多的关联，首先应该创建好“一”这端的对象（Book），然后再给其添加“多”这端的从对象（BookReview）

一对多关联的具体创建方式

- 主对象（Book）从对象（BookReview）一次创建完毕，将BookReview对象Add到Book的导航属性BookReviews中，将主对象加入DbSet，SaveChanges()
- Book对象对应的记录已存在于数据表中，将其装入内存后，向其所包容BookReviews对象集合添加BookReview对象后，SaveChanges()

D： Delete



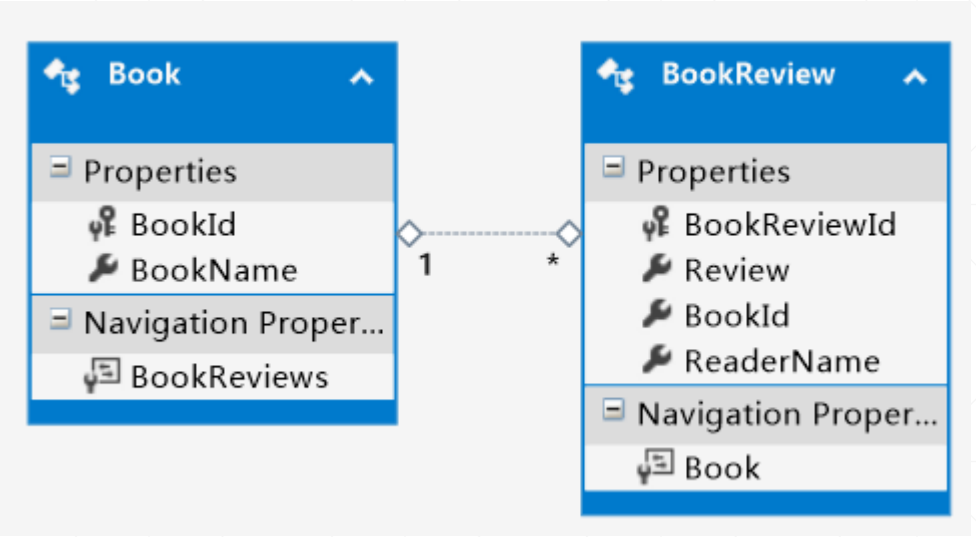
由于为一对多设置了“级联删除”特性，因此，只要删除了主对象Book，它所关联的所有从对象都会被移除……

对于需要删除单个从对象BookReview的场景，不能仅将其从主对象Book的集合属性中移除，而应该**从BookReview所对应的DbContext.DbSet中直接移除**！

U：Update

修改普通属性值

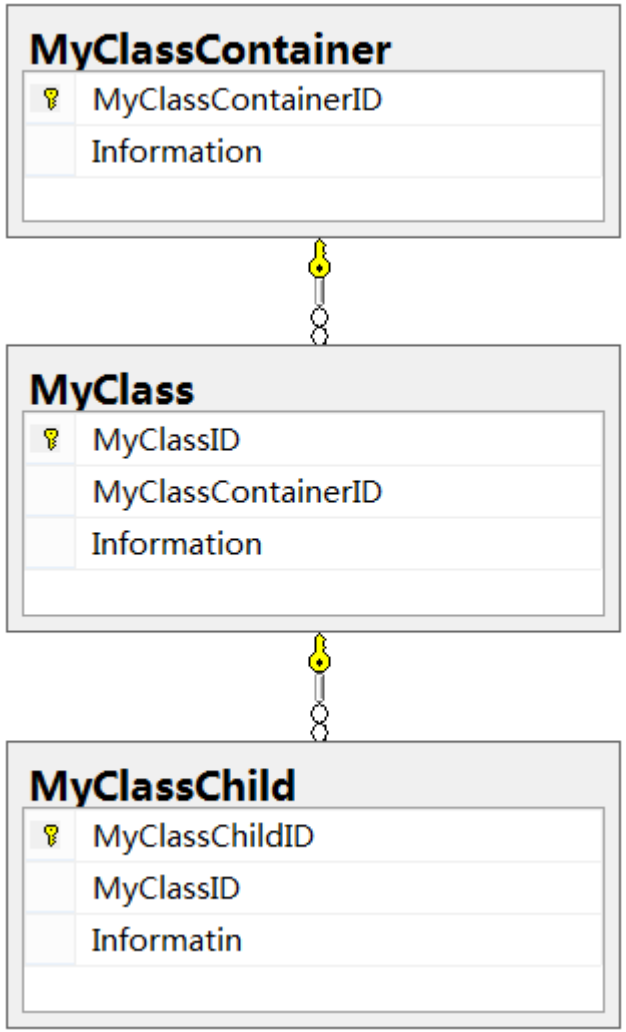
- 直接找到Book或BookReview对象，给其属性赋上新值
- SaveChanges()



子对象搬家

- 从源实体对象的集合属性中Remove掉“要搬家的”那个子对象
- 将子对象Add到目标实体对象的集合属性中。
- SaveChanges()

插入多层组合的对象



对于多层组合的对象，为保证EF工作正常，请遵循以下数据库设计规范：

（1）每个表对应一个实体对象类型
（2）每个表都有主键，外键不允许为null。
（3）相关表之间建立一对多关联，并且设置删除规则为“级联”

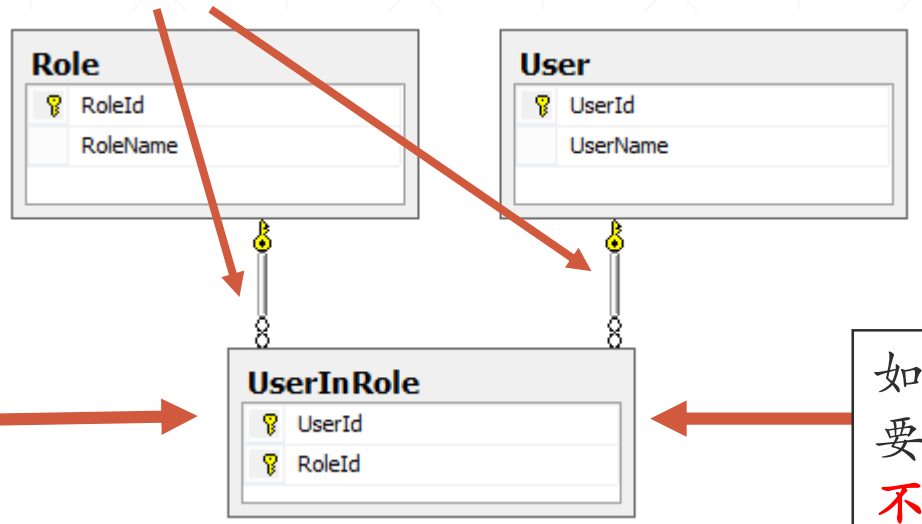
插入时，先创建好所有对象，依据组合层次关系装配好对象，仅将最顶层的对象加入DbSet，SaveChange()，EF会识别出所有下属子对象均为“新增”的。

参看MultiLevelEmbedTest.cs中的单元测试代码

处理多对多关联

多对多关联数据库设计规范

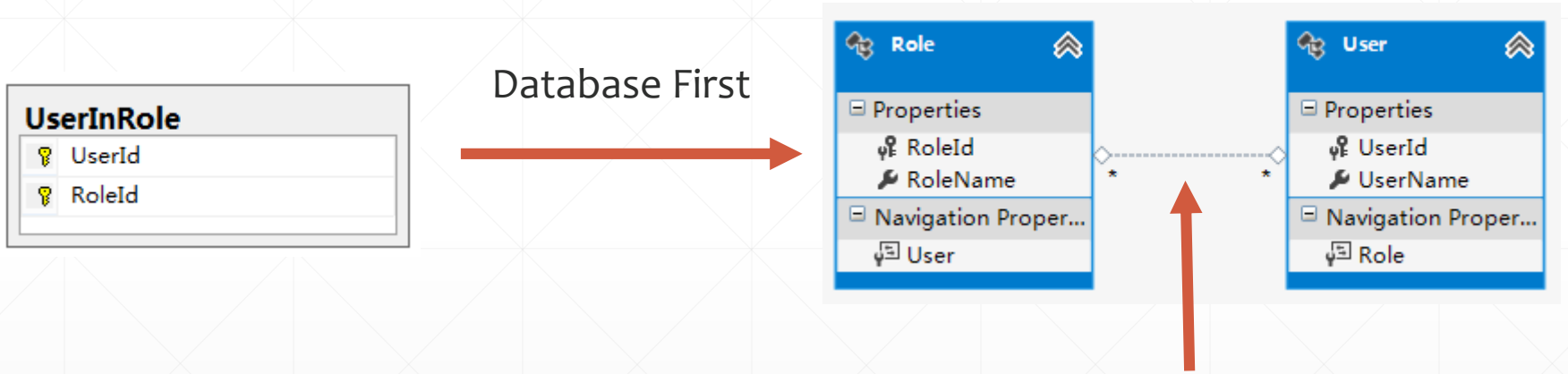
两个一对多的关联，应该启用“**级联删除**”特性



中间表**必须指定主键**：可以是**独立主键**（设计一个自增的ID字段），也可以是**复合主键**（如图，直接使用两个字段的组合作为本表主键）

如果采用复合主键，则要求这两个主键字段都**不能是自增类型的字段**

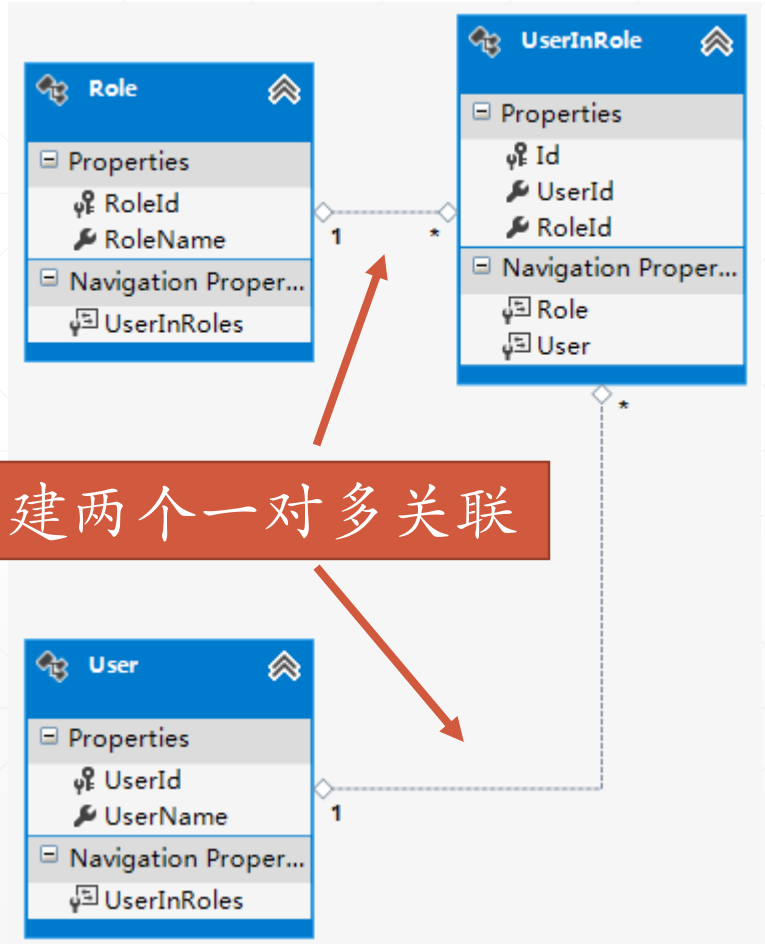
如果中间表没有定义独立主键……



EF会在生成的两个实体类之间建立多对多的关联

如果中间表定义了独立的主键……

Database First

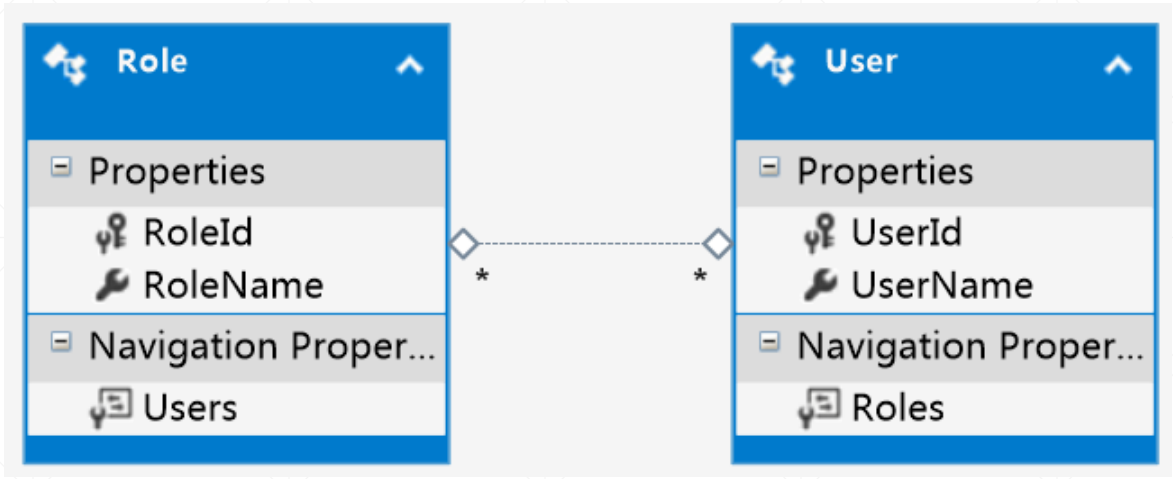


中间表成为独立的实体类

EF创建两个一对多关联

对于这种场景，其CRUD可参考一对多关联实现。

C：Create

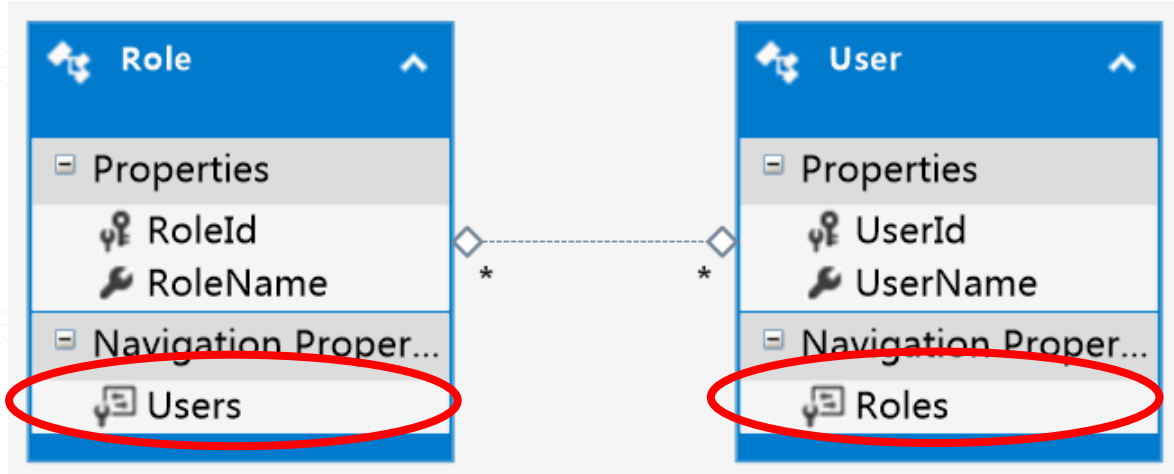


推荐的多对多关联创建方式

- 先创建独立的数据实体，并将它们追加到DbSet中，之后，再通过向导航对象集合添加实体对象，建立实体之间的关联
- 最后，SaveChanges()

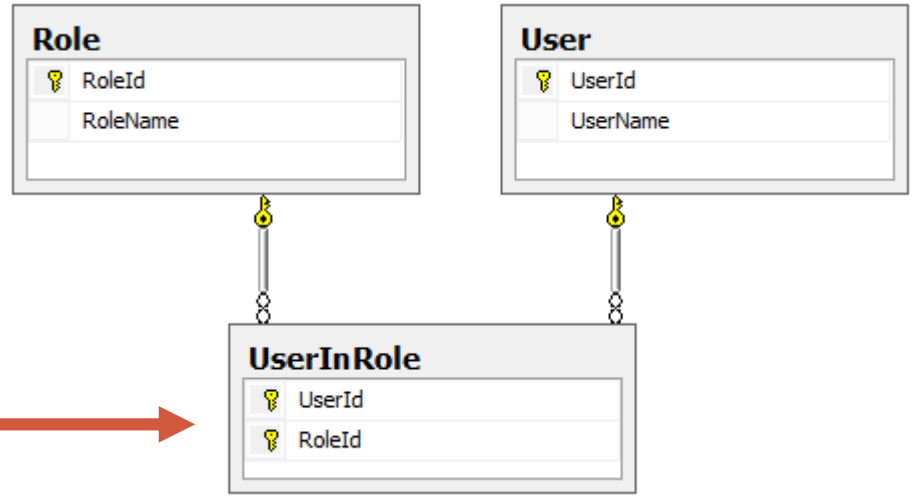
参看TestAddUserAndRoleAllInMemory ()方法

U：Update



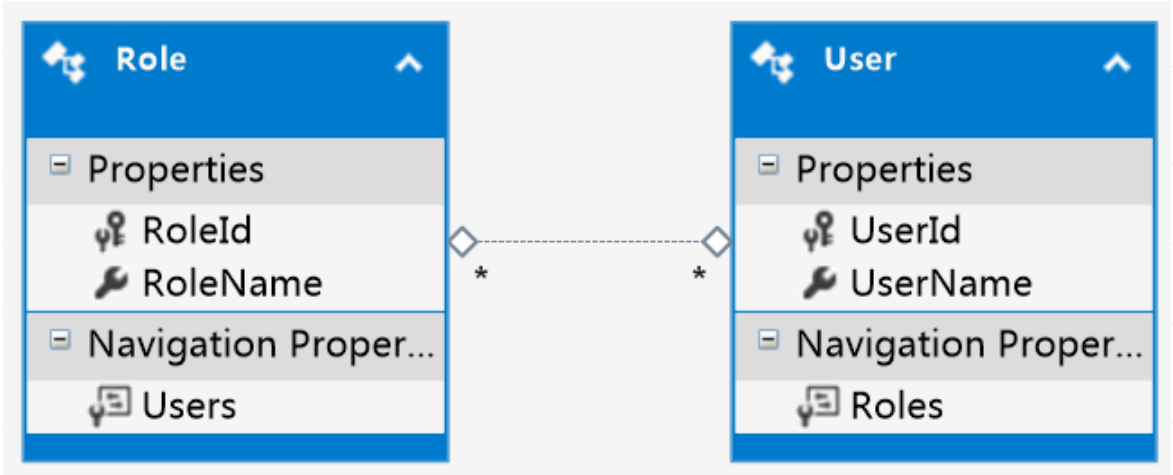
SaveChanges

通过**导航集合属性**把某个User加入到特定的Role的**Users**集合中，或者从此集合中移除某个User。



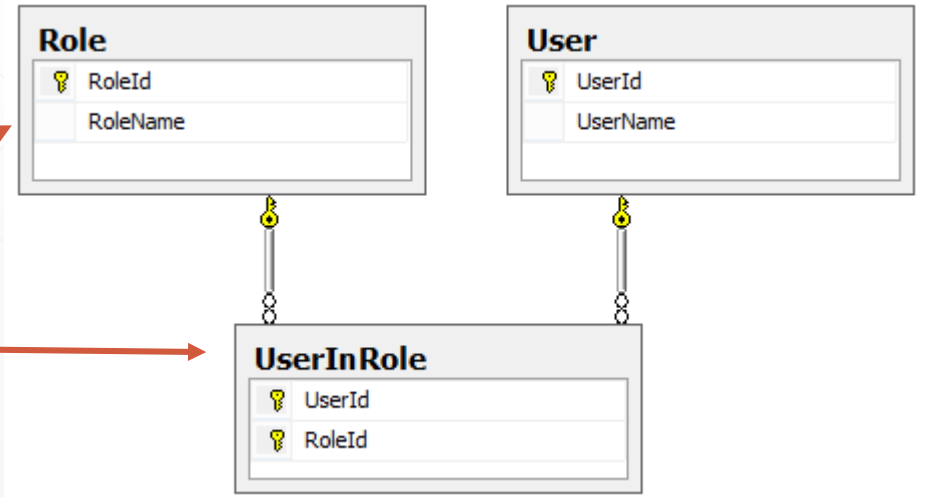
对于上述针对多对多关联本身的修改，如果User和Role记录都已经存在，EF将不动User表和Role表，**只在**中间表UserInRole中进行修改。

D：Delete



从DbSet中移除Role或User对象之后，再SaveChanges()

EF将删除Role或User表中的记录，同时UserInRole表中相关联的记录也将被同时删除



学习指南

掌握CRUD的基本编程技巧，是学习数据库应用程序开发技术的核心任务，请完成以下学习任务：

知识的整理与系统化工作：

1. 仔细琢磨，理解清楚Entity Framework的数据更新原理

2. 运行各个示例，仔细分析其输出，理解清楚其原因，并且整理好各种典型的CRUD代码，开发备用

在开发实践中学习：

自建一个数据库，自建相应数据表，然后自己编写代码，针对三种关联类型实现CRUD，只要你走完了这个过程，就能真正地用好Entity Framework。

不动手动脑，什么也学不到！