# Связь многие ко многим

**Последнее обновление: 31.10.2015**

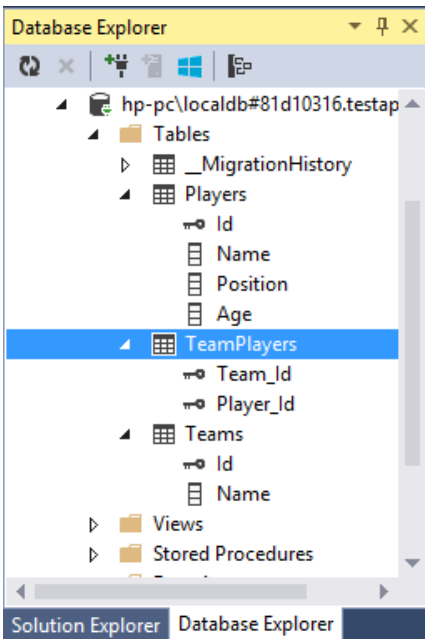
Еще одним способом ассоциации объектов является связь многие-ко-многим. Например, у нас есть модель футболистов и есть модель команд. На протяжении всей жизни футболист может поиграть в различных командах, а в одной команде может поиграть множество разных футболистов. На уровне моделей это выглядит так:

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

    public ICollection<Player> Players { get; set; }
        public Team()
    {
        Players = new List<Player>();
    }
}
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 Player()
    {
        Teams = new List<Team>();
        }
}
```

Обе модели имеют свойства-коллекции, через которые и будет осуществляться связь многие-ко-многим. В итоге, если мы используем CodeFirst, то автоматически будет создаваться база данных со следующей схемой:




То есть создается промежуточная таблица, которая хранит наборы пар Player-Team. И если бы мы использовали подход Database First, то нам надо было также создать эту таблицу.

Используем модели. Добавление и вывод:

```
using(SoccerContext db = new SoccerContext())
{
    // создание и добавление моделей
    Player pl1 = new Player { Name = "Роналду", Age = 31, Position = "Нападающий" };
    Player pl2 = new Player { Name = "Месси", Age = 28, Position = "Нападающий" };
    Player pl3 = new Player { Name = "Хави", Age = 34, Position = "Полузащитник" };
    db.Players.AddRange(new List<Player> { pl1, pl2, pl3 });
    db.SaveChanges();

    Team t1 = new Team { Name = "Барселона" };
    t1.Players.Add(pl2);
    t1.Players.Add(pl3);
    Team t2 = new Team { Name = "Реал Мадрид" };
    t2.Players.Add(pl1);
    db.Teams.Add(t1);
    db.Teams.Add(t2);
    db.SaveChanges();
    foreach(Team t in db.Teams.Include(t=>t.Players))
    {
        Console.WriteLine("Команда: {0}", t.Name);
        foreach(Player pl in t.Players)
        {
            Console.WriteLine("{0} - {1}", pl.Name, pl.Position);
        }
        Console.WriteLine();
    }
}
```

При добавление одной модели в список к другой важно помнить, что это список уже должен быть создан, иначе будет выброшено исключение. В данном случае мы создаем список в конструкторе обоих моделей. Также допустимо создание списка непосредственно в программе.

Редактирование:

```
// удаляем связи с одним объектом
Player pl_edit = db.Players.First(p=>p.Name=="Месси");
Team t_edit = pl_edit.Teams.First(p=>p.Name=="Барселона");
t_edit.Players.Remove(pl_edit);
```

Удаление игрока из списка команды будет означать удаление строки из таблицы TeamPlayers, в которой id игрока сопоставляется id команды.

Удаление же игрока или команды вообще из базы данных приводит к тому, что все строки в таблице TeamPlayers, которые содержат id удаленного объекта, также будут удалены:

```
Player pl_delete = db.Players.First(p=>p.Name=="Месси");
db.Players.Remove(pl_delete);
```

**Назад** **Содержание** **Вперед**