如下示例：

var task1 = Task.Run (() => {

    ...

    var myDbContext = new MyDbContext (options);

    var myEntity = myDbContext.MyEntities.Find (1);

myEntity.Name = myEntity.Name + "1";

    myDbContext.SaveChanges ();

});

var task2 = Task.Run (() => {

    ...

var myDbContext = new MyDbContext (options);

    var myEntity = myDbContext.MyEntities.Find (1);

myEntity.Name = myEntity.Name + "1";

    myDbContext.SaveChanges ();

});

Task.WaitAll (task1, task2);

如果myEntity.Name为“2”，那么执行该语句，最后myEntity.Name的结果为“21”，最后完成的task覆盖了前面完成的task，但却没有告诉我们发生了并发冲突

**并发冲突解决**

public class MyEntity

{

    [Key]

    public int Id { get; set; }

    public string Name { get; set; }

    // 并发控制字段

    [ConcurrencyCheck]

    public long Timestamp { get; set; }

}

var task1 = Task.Run (() => {

    ...

    var myDbContext = new MyDbContext (options);

    var myEntity = myDbContext.MyEntities.Find (1);

myEntity.Name = myEntity.Name + "1";

    // 更新并发控制字段

myEntity.Timestamp = DateTime.Now.Ticks;

    myDbContext.SaveChanges ();

});

var task2 = Task.Run (() => {

    ...

var myDbContext = new MyDbContext (options);

    var myEntity = myDbContext.MyEntities.Find (1);

myEntity.Name = myEntity.Name + "1";

    // 更新并发控制字段

myEntity.Timestamp = DateTime.Now.Ticks;

    myDbContext.SaveChanges ();

});

Task.WaitAll (task1, task2);

执行上面的代码

前面完成的task会成功保存

后面完成的task会抛出并发异常DbUpdateConcurrencyException，其task的更新会失败

其控制并发原理是在更新筛选时加上控制并发字段的筛选

UPDATE [MyEntity] SET [Name] = "21"

WHERE [Id] = 1 AND [Timestamp] = "原先取回的Timestamp";