efcore

dotnet -ef □□□□

```
[Console]::OutputEncoding = [System.Text.Encoding]::UTF8
```

□□□□□

```
1 □□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
public class Entity
{
public int Id { get; set; }
public string Name { get; set; }
public int Version { get; set; } // □□□
}
// EF Core □□
modelBuilder.Entity<Entity>().Property(e => e.Version).IsConcurrencyToken();
// □□□□□□□
var entity = await context.Entities.FindAsync(id);
entity.Name = "new name";
await context.SaveChangesAsync(); // □□□□□□□□□              DbUpdateConcurrencyException
□□□□□□□□
try
{
await context.SaveChangesAsync();
}
catch (DbUpdateConcurrencyException)
{
// □□□□□□□□□□□□□□□□□□
}
2 □□□
□□□□□□□□□□□□□□□
// SQL Server
SELECT * FROM Entities WITH (UPDLOCK, ROWLOCK) WHERE Id = @id
// MySQL
```

```sql
SELECT * FROM Entities WHERE Id = @id FOR UPDATE
```

在EF Core中执行原始SQL锁定，然后在事务中更新

3 分布式锁

适合跨服务或多实例场景，使用Redis等

```csharp
var lockKey = $"lock:entity:{id}";
var acquired = await redis.StringSetAsync(lockKey, "1", TimeSpan.FromSeconds(10), When.NotExists);
if (!acquired)
{
throw new Exception("资源正在被其他请求修改");
}
try
{
// 执行更新操作
}
finally
{
await redis.KeyDeleteAsync(lockKey);
}
```

4 业务层面控制

接口幂等性设计，前端防重复提交，后端验证状态

建议优先使用乐观锁，性能最好且实现简单。高冲突场景再考虑悲观锁或分布式锁。

版本号 #4
由 叶子 创建于 9 三月 2026 13:53:18
由 叶子 更新于 21 七月 2026 10:06:30