

# 第58期-数据库分区分表分库与读写分离

2019年8月23日 11:30

## 课程说明

- 1、后续将使用 Microsoft.EntityFrameworkCore 3.X 版本。
- 2、微软在 EF Core 中日志书写重构。

```
ILoggerFactory myLoggerFactory = LoggerFactory.Create(builder => { builder.AddConsole(); });
dbContextOptionsBuilder..UseLoggerFactory(myLoggerFactory);
```

## 索引

索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息。

## 分区

就是把一张表的数据分成N个区块，在逻辑上看最终只是一张表，但底层是由N个物理区块组成的。

## 分表

就是把一张表按一定的规则分解成N个具有独立存储空间的实体表，系统读写时需要根据定义好的规则得到对应的表名称，然后再操作它。

## 分库

一旦分表，一个库中的表会越来越多，计算机处理性能是有限的，单机数据库的瓶颈也是显而易见的，分库后可以单独服务器集群部署，更好的提高大数据扩展能力。

## 读写分离

对于时效性不高的数据，可以通过读写分离缓解数据库压力。需要解决的问题：在业务上区分哪些业务上是允许一定时间延迟的，以及数据同步问题。

读写分离的实现：日志传送、事务复制和 Always On 技术。

## 在 EF Core 中实现读写分离

<https://www.cnblogs.com/CreateMyself/p/9241523.html>

<https://github.com/Arch/UnitOfWork/tree/master/UnitOfWork>

```
public static class ReadWriteExtensions
{
    /// <summary>
    /// Changes the database name. This require the databases in the same machine. NOTE: This only work
    for MySQL right now.
    /// </summary>
    /// <param name="database">The database name.</param>
```

```

    /// <remarks>
    /// This only been used for supporting multiple databases in the same model. This require the
databases in the same machine.
    /// </remarks>
    public static void ChangeDatabase(this DbContext dbContext, string nameOrConnectionString)
    {
        var connection = dbContext.Database.GetDbConnection();

        if (connection.State.HasFlag(ConnectionState.Open))
        {
            connection.ChangeDatabase(nameOrConnectionString);
        }
        else
        {
            connection.ConnectionString = nameOrConnectionString;
        }
    }

    /// <summary>
    /// Changes the table name. This require the tables in the same database.
    /// </summary>
    /// <param name="table"></param>
    /// <remarks>
    /// This only been used for supporting multiple tables in the same model. This require the tables in
the same database.
    /// </remarks>
    public virtual void ChangeTable<TEntity>(this DbContext dbContext, string table)
    {
        if (dbContext.Model.FindEntityType(typeof(TEntity)).Relational() is
RelationalEntityTypeAnnotations relational)
        {
            relational.TableName = table;
        }
    }
}

```