

# 第38期-多租户系统的最佳实践

2019年6月18日 10:59

## 全局筛选器的一些限制

**局限性一：HasQueryFilter方法过滤筛选无法应用于导航属性。**

```
builder.HasQueryFilter(f => !f.IsDeleted && f.Posts.All(w => !w.IsDeleted));
```

**局限性二：HasQueryFilter方法过滤筛选只能定义在基类中，无法对子类进行过滤。**

```
modelBuilder.Entity<Payment>().HasQueryFilter(b => !b.IsDeleted);
```

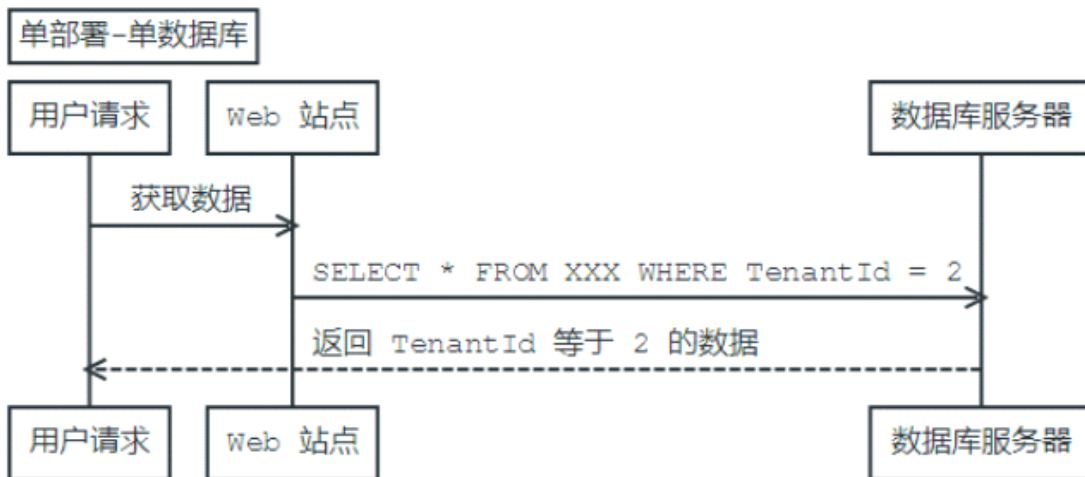
```
modelBuilder.Entity<AlipayPayment>().HasQueryFilter(b => !b.IsDeleted);
```

## 多租户架构设计方案

### 共享数据库

如果软件系统仅部署一个实例，并且所有租户的数据都是存放在一个数据库里面的，那么可以通过一个 TenantId (租户 Id) 来进行数据隔离。那么当我们执行 SELECT 操作的时候就会附加上当前登录用户租户 Id 作为过滤条件，那么查出来的数据也仅仅是当前租户的数据，而不会查询到其他租户的数据。

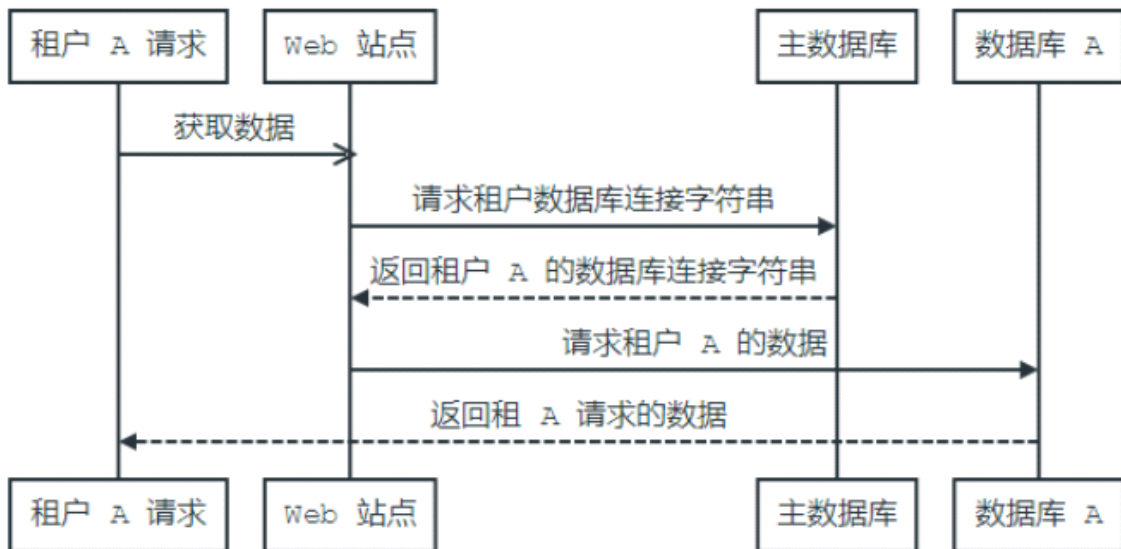
这是共享程度最高、隔离级别最低的模式。需要在设计开发时加大对安全的开发量。



### 多数据库

为每一个租户提供一个单独的数据库，在用户登录的时候根据用户对应的租户 ID，从一个数据库连接映射表获取到当前租户对应的数据库连接字符串，并且在查询数据与写入数据的时候，不同租户操作的数据库是不一样的。

这种方案的用户数据隔离级别最高，安全性最好，但维护和购置成本较高。



## 基于架构的单数据库

也有一种介于两者之间的方案: 共享数据库, 独立 Schema. 但实际应用的应该不多.

## 引用参考文档

[EF Core 实现多租户](#)

[如何理解多租户架构?](#)

[多租户技术](#)

[多租户与多用户的区别?](#)

## 多租户的应用

[Multi-Tenancy.md](#)

[Tenants \(OrchardCore.Tenants\)](#)

[Setting up a multi tenant Orchard site](#)