要求：

生成一个用户名-角色名的多对多关联关系，尽量少写sql语句，不使用外键约束

## Jhipster

**思路：**

jhi官方支持的关系维护表带有外键约束，因此采用创建简单实体类，在应用层利用代码逻辑实现关联查询的方法实现要求。

实际过程：

1. 利用命令行生成的方式生成三个实体类（UserBase/RoleName/UserBaseRoleName），其中UserBase和RoleName生成时在命令行选项中选择生成DTO和对应的mapper、生成service层。
2. 为DTO增加对方的属性，并在mapper创建新方法实现toDTO()
3. 修改service，在service实现关联逻辑，并以DTO的形式返回给controller使用。
4. Controller可以直接使用Jhi生成的，因为Jhi在controller层使用的是service层的方法，实体类对象是DTO对象。

**手动工作：**

修改DTO，为mapper增加注解以生成映射方法，修改service层实现关联逻辑，为持久层增加接口方法供复杂查询使用。

**自动生成：**

实体类生成功能、生成controller、生成service功能，还使用了jhi支持的MapStruct实现toDTO()、toEntity()。

## Jeecg：

**思路：**

Jeecg支持无外键约束创建表关联关系，但只支持一对一、一对多关系，并且其一对多关系并不支持一张附表被两张主表共享，因此不能利用其一对多关系实现多对多关系。因此也采用创建简单实体类，在应用层利用代码逻辑实现关联查询的方法实现要求。

Jeecg大量使用了继承关系，修改起来很麻烦，耦合度很高。Service层是有mybatis-plus内置的继承关系，在service层添加DTO对象并实现查询逻辑困难，因此采用给实体类增加对方的成员list，并打注解@TableField(exist = false) 标记该属性不是数据库字段的方法实现功能，这里是把实体类当作DTO对象使用。

1. 利用online生成三张实体类单表（UserBase/RoleName/UserBaseRoleName）
2. 修改生成的实体类UserBase和RoleName，为其增加一个对方的成员list
3. 在controller层实现关联查询逻辑
4. Mybatis-plus的无sql语句查询方式为使用QueryWrapper条件构造器，然后传入MP实现的service方法，进而实现复杂查询。

**手动工作：**

修改Entity实体类、增加service接口、实现service接口方法、修改controller实现关联逻辑、利用Wrapper构造复杂查询语句

**自动生成：**

单表生成（附带entity/mapper/service/controller生成）、MyBatis-Plus的条件构造器实现无sql语句复杂查询。

**对比：**

使用这两个脚手架实现多对多查询后，有以下体会。

jhi使用聚合关系较多，因而耦合度不高，要修改逻辑比较简单，而且jhi支持实体类、service、controller各层单独生成，实际使用较方便；

jeecg使用mybatis-plus作为数据库交互框架，且使用到了mybatis-plus的service层curd接口，这些类使用了继承关系，因此有很强的依赖关系，修改起来不是那么方便。最后我只能选择修改实体类的方式实现DTO的功能。

在编写sql语句方面，JPA和MyBatis-Plus都提供了条件构造器，因而两者都实现了无sql语句复杂查询。在实际使用过程中，JPA构造查询方法时IDEA支持自动提示，MyBatis-Plus的条件构造器需要自己核对数据库字段，没有自动检错提示。