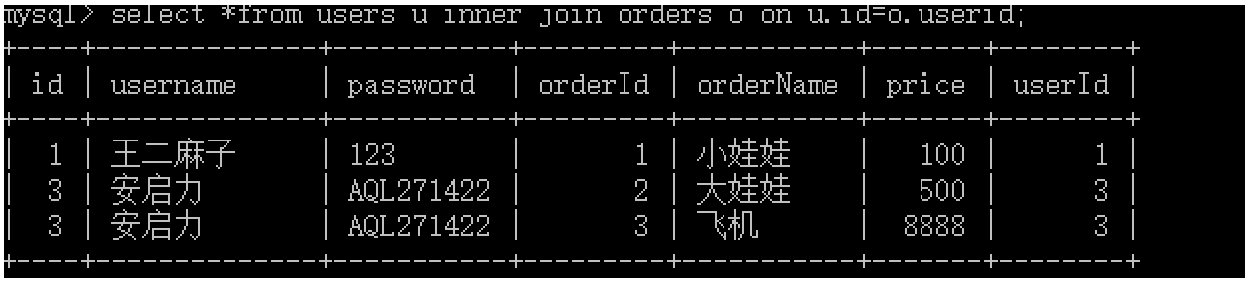
一，一对多的关联关系（***）

总结：逆向工程能够帮助程序员快速生成实体类、以及局部配置文件等信息。

注意使用逆向工程生成的实体类的实体属性的命名都不是采用驼峰式命名，需要进行转换。

1，假设数据库中已经有用户表users和订单表orders；



```
mysql> select *from users u inner join orders o on u.id=o.userid;
+----+-------------+-----------+---------+-----------+-------+--------+
| id | username    | password  | orderId | orderName | price | userId |
+----+-------------+-----------+---------+-----------+-------+--------+
|  1 | 王二麻子    | 123       |       1 | 小娃娃    |   100 |      1 |
|  3 | 安启力      | AQL271422 |       2 | 大娃娃    |   500 |      3 |
|  3 | 安启力      | AQL271422 |       3 | 飞机      |  8888 |      3 |
+----+-------------+-----------+---------+-----------+-------+--------+
```

其中id和orderid是主键并且是自增的，userid是和id一样。

2，java的测试demo代码：

```
package cn.java.dao;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Map;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import cn.java.entity.user;
public class one2manyDao {
	static SqlSession session = null;
	static {
		SqlSessionFactoryBuilder sessionFactoryBuilder = new 
SqlSessionFactoryBuilder();
		try {
			InputStream ins = Resources.getResourceAsStream("mybatis.xml");
			SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
			session = sessionFactory.openSession();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	@Test
	public void one2manyTest1(){
		List<user> maps = 
session.selectList("cn.java.dao.one2manyDao.one2manyTest1");
		System.out.println(maps);
```

```
        }
}
```

3，局部配置文件one2manyDao.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!-- namespace表示命名空间，一个局部配置文件只有一个命名空间。其值是某一个dao层类的具体路径 -->
<mapper namespace="cn.java.dao.one2manyDao">
        <!-- sql语句保存在mybatis的局部配置文件中 ，增删改查的SQL语句需要放置在相对应的标签内部 -->
        <!-- select存放查询语句，id属性在整个局部配置文件中必须唯一，其值SQL查询在dao层的方法名 -->
        <!-- resultType指定当前SQL查询返回的数据类型 ，存放某一条数据的数据类型 类型不是SQL语句的最终类型，而是某一条数据的类型 -->
        <resultMap type="cn.java.entity.user" id="baseMap">
                <result property="id" javaType="Long" column="id" />
                <result property="username" javaType="String" column="username" />
                <result property="password" javaType="String" column="password" />
                <!-- collection中的property表示在一端中多端属性的属性名；ofType表示该属性名对应的实体类所在的路径 -->
                <collection property="orders" ofType="cn.java.entity.order">
                        <!-- 作为实体类中的主键字段，可以使用id标签包含，也可以使用result标签包含 -->
                        <result property="orderid" javaType="Long" column="orderid" />
                        <result property="ordername" javaType="String" column="ordername" />
                        <result property="price" javaType="Float" column="price" />
                        <result property="userid" javaType="Long" column="userid" />

                </collection>
        </resultMap>
        <select id="one2manyTest1" resultMap="baseMap">
                <!-- sql语句后面的分号可要可不要 -->
                select *from users u inner join orders o on u.id=o.userid;
        </select>
</mapper>
```

4，主配置文件mybatis.xml配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
<!-- 配置连接数据库的连接环境 -->
```

```
	<environments default="mysql">
	<!-- 开始配置mysql 可以配置多个<environment id="值">，如mysql、Oracle等，
若实际应用中需要默认使用哪个数据库，则在上面的default写上mysql或者Oracle即可-->
		<environment id="mysql">
				<!-- 配置事务 -->
				<transactionManager type="JDBC"></transactionManager>
				<!-- 配置数据源 -->
				<dataSource type="POOLED">
					<property name="driver"
value="com.mysql.cj.jdbc.Driver"/>
					<property name="url"
value="jdbc:mysql://localhost:3306/db_mybatis?
useSSL=false&serverTimezone=UTC&characterEncoding=utf8"/>
					<property name="username" value="root"/>
					<property name="password" value="AQL271422"/>
				</dataSource>
		</environment>
	</environments>
	<!-- 关联局部配置文件 -->
	<mappers>
	<mapper resource="cn/java/dao/one2manyDao.xml"/>
	</mappers>
</configuration>
```

5，user实体类：

```
package cn.java.entity;
import java.util.List;
public class user {
	private Long id;
	private String username;
	private String password;
	private List<order> orders;
	@Override
	public String toString() {
		return "user [id=" + id + ", username=" + username + ", password=" +
password + ", orders=" + orders + "]";
	}
	public user() {
		super();
	}
	public List<order> getOrders() {
		return orders;
	}
	public void setOrders(List<order> orders) {
		this.orders = orders;
	}
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username == null ? null : username.trim();
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password == null ? null : password.trim();
	}
}
```

6，order实体类：

```
package cn.java.entity;
public class order {
	@Override
	public String toString() {
		return "order [orderid=" + orderid + ", ordername=" + ordername + ",
price=" + price + ", userid=" + userid
				+ "]";
	}
	private Long orderid;
	private String ordername;
	private Float price;
	private Long userid;
	public Long getOrderid() {
		return orderid;
	}
	public void setOrderid(Long orderid) {
		this.orderid = orderid;
	}
	public String getOrdername() {
		return ordername;
	}
	public void setOrdername(String ordername) {
		this.ordername = ordername == null ? null : ordername.trim();
	}
	public Float getPrice() {
		return price;
	}
	public void setPrice(Float price) {
		this.price = price;
	}
	public Long getUserid() {
		return userid;
	}
	public void setUserid(Long userid) {
		this.userid = userid;
	}
}
```