

#### 咕泡学院（每天20：30腾讯课堂公开课）

今日课程：基于SpringJDBC实现ORM及分库分表

|  |  |
| --- | --- |
| 高级课题 (3) | **不只做一个技术者，更要做一个思考者。**  咕泡学院-Tom老师 QQ :441221062  咕泡学院VIP课程咨询 QQ : 1370315370  咕泡学院往期视频 QQ： 2915505261  咕泡学院-技术交流群 QQ ：608583947 (视频下载联系)  打开腾讯课堂 搜索 咕泡学院，立即报名即可天天听老师讲课啦。 |

课程安排

一、什么是ORM

对象关系映射（Object Relational Mapping）

二、为什么要使用ORM框架

ORM 框架有哪些？

Hibernate 全自动、MyBatis 半自动对比（手自一体）、SpringJDBC纯手动档（设计思想）

Hibernate的数据库移植性远大于Mybatis

Hibernate有完整的日志系统，Mybatis只有基本的记录功能

SQL优化上，Mybatis更方便，可控性较强。

SSM /SSH 一个S搞定（Spring生态系统SpringMVC/SpringJDBC/AOP/IOC/SpringData/Spring Cloud/Spring Boot）

三、代码演示ORM实现原理

1、了解ResultSet的 getMetaData()方法和getColumnCount()

2、利用反射机制扫描出实体中的注解，将实体类名和表名，属性名和字段名关联

3、从结果集中取值并封装实体对象

四、为什么要分库、分表

数据库中的数据量不一定是可控的，在未进行分库分表的情况下，随着时间和业务的发展，库中的表会越来越多，表中的数据量也会越来越大，相应地，数据操作，增删改查的开销也会越来越大；另外，由于无法进行分布式式部署，而一台服务器的资源（CPU、磁盘、内存、IO等）是有限的，最终数据库所能承载的数据量、数据处理能力都将遭遇瓶颈。

五、分享企业内部框架、设计ORM及分库、分表操作API

1、单表操作实现NoSql

2、如何设计分库API，利用ThreadLocal动态切换数据源

3、如何设计分表API，自定义分表算法，动态计算表名

4、如何设计读写分离API，数据库主从实现读写分离

5、设计QueryRule类，NoSql的重要中间件

6、设计常用的ORM操作API，增、删、改、查

六、浅谈常用分库分表规则

设计表的时候需要确定此表按照什么样的规则进行分库分表。例如，当有新用户时，程序得确定将此用户信息添加到哪个表中；同理，当登录的时候我们得通过用户的账号找到数据库中对应的记录，所有的这些都需要按照某一规则进行。

路由

通过分库分表规则查找到对应的表和库的过程。如分库分表的规则是user\_id mod 4的方式，当用户新注册了一个账号，账号id的123,我们可以通过id mod 4的方式确定此账号应该保存到User\_0003表中。当用户123登录的时候，我们通过123 mod 4后确定记录在User\_0003中。

Range（范围）–这种模式允许将数据划分不同范围。例如可以将一个表通过年份划分成若干个分区。

Hash（哈希）–这中模式允许通过对表的一个或多个列的Hash Key进行计算，最后通过这个Hash码不同数值对应的数据区域进行分区。例如可以建立一个对表主键进行分区的表。

Key（键值）-上面Hash模式的一种延伸，这里的Hash Key是MySQL系统产生的。

List（预定义列表）–这种模式允许系统通过预定义的列表的值来对数据进行分割。

七、分库分表产生的问题，及注意事项

1.分库分表维度的问题

假如用户购买了商品,需要将交易记录保存取来，如果按照用户的纬度分表，则每个用户的交易记录都保存在同一表中，所以很快很方便的查找到某用户的购买情况，但是某商品被购买的情况则很有可能分布在多张表中，查找起来比较麻烦。反之，按照商品维度分表，可以很方便的查找到此商品的购买情况，但要查找到买人的交易记录比较麻烦。

所以常见的解决方式有：

a.通过扫表的方式解决，此方法基本不可能，效率太低了。

b.记录两份数据，一份按照用户纬度分表，一份按照商品维度分表。

c.通过搜索引擎解决，但如果实时性要求很高，又得关系到实时搜索(聚合查询)。

2.联合查询的问题

联合查询基本不可能，因为关联的表有可能不在同一数据库中。

3.避免跨库事务

避免在一个事务中修改db0中的表的时候同时修改db1中的表，一个是操作起来更复杂，效率也会有一定影响。

4.尽量把同一组数据放到同一DB服务器上。

八、总结

1、ORM优、缺点

优点：

1.隐藏了数据访问细节，“封闭”的通用数据库交互，ORM的核心。他使得我们的通用数据库交互变得简单易行，并且完全不用考虑该死的SQL语句。快速开发。

2.ORM使我们构造固化数据结构变得简单易行。

缺点：

1.无可避免的，自动化意味着映射和关联管理，代价是牺牲性能（早期，这是所有不喜欢ORM人的共同点）。现在的各种ORM框架都在尝试使用各种方法来减轻这块（LazyLoad，Cache），效果还是很显著的。

2.面向对象的查询语言(X-QL)作为一种数据库与对象之间的过渡,虽然隐藏了数据层面的业务抽象,但并不能完全的屏蔽掉数据库层的设计,并且无疑将增加学习成本。

3.对于复杂查询，ORM仍然力不从心。虽然可以实现，但是不值得。

世上没有驴是不吃草的(又想好又想巧,买个老驴不吃草)，任何优势的背后都隐藏着缺点，这是不可避免的。问题在于，我们是否能容忍缺点。

2、数据库操作最多的就是Query

QueryRule的设计降低了操作复杂度

3、分库、分表操作，将分库分表规则抽离统一管理