# 分表分库

## 垂直拆分

垂直拆分就是要把表按模块划分到不同**数据库**表中（当然原则还是不破坏第三范式），这种拆分在大型网站的演变过程中是很常见的。当一个网站还在很小的时候，只有小量的人来开发和维护，各模块和表都在一起，当网站不断丰富和壮大的时候，也会变成多个子系统来支撑，这时就有按模块和功能把表划分出来的需求。其实，相对于垂直切分更进一步的是服务化改造，说得简单就是要把原来强耦合的系统拆分成多个弱耦合的服务，通过服务间的调用来满足业务需求看，因此表拆出来后要通过服务的形式暴露出去，而不是直接调用不同模块的表，淘宝在架构不断演变过程，最重要的一环就是服务化改造，把用户、交易、店铺、宝贝这些核心的概念抽取成独立的服务，也非常有利于进行局部的优化和治理，保障核心模块的稳定性

垂直拆分用于分布式场景。

## 水平拆分

上面谈到垂直切分只是把表按模块划分到不同数据库，但没有解决单表大数据量的问题，而水平切分就是要把一个表按照某种规则把数据划分到不同表或数据库里。例如像计费系统，通过按时间来划分表就比较合适，因为系统都是处理某一时间段的数据。而像SaaS应用，通过按用户维度来划分数据比较合适，因为用户与用户之间的隔离的，一般不存在处理多个用户数据的情况，简单的按user\_id范围来水平切分

通俗理解：水平拆分行，行数据拆分到不同表中， 垂直拆分列，表数据拆分到不同表中

### 水平分割案例

思路:在大型电商系统中，每天的会员人数不断的增加。达到一定瓶颈后如何优化查询。

可能大家会想到索引，万一用户量达到上亿级别，如何进行优化呢？

使用水平分割拆分数据库表。

### 如何使用水平拆分数据库

使用水平分割拆分表，具体根据业务需求，有的按照注册时间、取摸、账号规则、年份等。

### 使用取摸方式分表

首先我创建三张表 user0 / user1 /user2 , 然后我再创建 uuid表，该表的作用就是提供自增的id。

|  |
| --- |
| create table user0(  id int unsigned primary key ,  name varchar(32) not null default '',  pwd varchar(32) not null default '')  engine=myisam charset utf8;  create table user1(  id int unsigned primary key ,  name varchar(32) not null default '',  pwd varchar(32) not null default '')  engine=myisam charset utf8;  create table user2(  id int unsigned primary key ,  name varchar(32) not null default '',  pwd varchar(32) not null default '')  engine=myisam charset utf8;  create table uuid(  id int unsigned primary key auto\_increment)engine=myisam charset utf8; |

### 创建一个demo项目

#### POM文件

|  |
| --- |
| **<parent>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-parent</artifactId>**  **<version>1.3.3.RELEASE</version>**  **</parent>**  **<dependencies>**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-jdbc</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-test</artifactId>**  **<scope>test</scope>**  **</dependency>**  **<dependency>**  **<groupId>mysql</groupId>**  **<artifactId>mysql-connector-java</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-web</artifactId>**  **</dependency>**  **</dependencies>** |

#### Service代码

|  |
| --- |
| **@Service**  **public class UserService {**  **@Autowired**  **private JdbcTemplate jdbcTemplate;**  **public String regit(String name, String pwd) {**  **// 1.先获取到 自定增长ID**  **String idInsertSQL = "INSERT INTO uuid VALUES (NULL);";**  **jdbcTemplate.update(idInsertSQL);**  **Long insertId = jdbcTemplate.queryForObject("select last\_insert\_id()", Long.class);**  **// 2.判断存储表名称**  **String tableName = "user" + insertId % 3;**  **// 3.注册数据**  **String insertUserSql = "INSERT INTO " + tableName + " VALUES ('" + insertId + "','" + name + "','" + pwd**  **+ "');";**  **System.*out*.println("insertUserSql:" + insertUserSql);**  **jdbcTemplate.update(insertUserSql);**  **return "success";**  **}**  **public String get(Long id) {**  **String tableName = "user" + id % 3;**  **String sql = "select name from " + tableName + " where id="+id;**  **System.*out*.println("SQL:" + sql);**  **String name = jdbcTemplate.queryForObject(sql, String.class);**  **return name;**  **}**  **}** |

#### Controller

|  |
| --- |
| **@RestController**  **public class UserController {**  **@Autowired**  **private UserService userService;**  **@RequestMapping("/regit")**  **public String regit(String name, String pwd) {**  **return userService.regit(name, pwd);**  **}**  **@RequestMapping("/get")**  **public String get(Long id) {**  **String name = userService.get(id);**  **return name;**  **}**  **}** |

# 读写分离