Table of Contents

为什么学习数据库？

1、90%以上的软件都需要操作数据，比如游戏、社交、新闻、商城、财务等，这些软件都在不停的展示、存储数据，它们的数据都存储在数据库，数据库是软件的基础。

2、测试工程师在测试软件的过程中，不仅需要在界面上操作，还需要检查数据库中的数据是否正确。从而在软件出问题时，测出更深层的问题。

例如：测试注册登录功能，在输入了注册信息后，提示注册成功，但是使用刚才注册的信息登录不成功。 这时需要检查数据库中是否保存了正确的注册信息，如果数据库中没有保存数据，那么使用刚才注册的账号肯定登录不了，这样就能定位问题：注册时，没有把数据存储起来。

数据库阶段知识点

1. 数据库的基本概念

2. Navicat 操作数据库
3. SQL 语言（重点）
4. MySQL 高级（了解）
5. Redis

学习目标

- 熟练编写 SQL 语言中的查询语句

Redis

Redis介绍

1. NoSQL 简介

- “NoSQL”一词最早于1998年被用于一个轻量级的关系数据库的名字
- 随着 Web2.0 的快速发展，NoSQL 概念在2009年被提了出来，非关系型、分布式数据存储得到了快速的发展，它们不保证关系数据的 ACID 特性
- NoSQL 在2010年风生水起，现在国内外众多大小网站，如 Facebook、Google、淘宝、京东、百度等，都在使用 NoSQL 开发高性能的产品
- 对于一名程序员来讲，使用nosql已经成为一条必备技能
- NoSQL 最常见的解释是“non-relational”，“Not Only SQL”也被很多人接受，指的是非关系型的数据库

2. Redis 简介

- Redis 是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。从2010年3月15日起，Redis的开发工作由VMware主持
- Redis 是一个开源（BSD许可）的、内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件
- redis是一个高性能的key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sortedset--有序集合)和hash（哈希类型）。redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了Python，Ruby，Erlang，PHP客户端，使用很方便
- Redis 支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。从盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助

2.1 服务端操作

启动服务，命令行输入下面命令

```
redis-server
```

2.2 客户端操作

启动客户端，命令行输入下面命令

```
redis-cli

如果上面打开方式显示中文时会出现乱码，用下面命令打开客户端可以正常显示中文
redis-cli --raw
```

运行测试命令

```
ping
```

切换数据库

数据库没有名称，默认有16个，通过0-15来标识

```
select 1
```

2.3 数据操作

- Redis 是 key-value 的数据结构，每条数据都是一个键值对
- 键的类型是字符串
- 注意：键不能重复
- 值的类型分为五种：
  - 字符串string
  - 哈希hash
  - 链表list
  - 集合set
  - 有序集合zset

string

- string是 Redis 最基本的类型
- 最大能存储512MB数据
- string类型是二进制安全的，可以存储任何数据，比如数字、图片等

增加、修改

- 如果设置的键不存在则为添加，如果设置的键已经存在则修改
- 设置键值

```
set key value
```

- 例1：设置键为'user1'值为'wzj'的数据

```
set 'user1' 'wzj'
```

- 设置键值及过期时间，以秒为单位

```
setex key seconds value
```

- 例2：设置键为'user2'值为'lb'过期时间为3秒的数据

```
setex 'user2' 3 'lb'
```

- 设置多个键值

```
mset key1 value1 key2 value2 ...
```

- 例3：设置键为'user4'值为'xq'、键为'user5'值为'dq'、键为'user6'值为'zgl'、键为'user7'值为'bq'的数据

```
mset 'user4' 'xq' 'user5' 'dq' 'user6' 'zgl' 'user7' 'bq'
```

- 追加值

```
append key value
```

- 例4：向键为user1中追加值' haha'

```
append 'user1' ' haha'
```

获取

- 获取：根据键获取值，如果不存在此键则返回nil

```
get key
```

- 例5：获取键'user1'的值

```
get 'user1'
```

- 根据多个键获取多个值

```
mget key1 key2 ...
```

- 例6：获取键'user3'、'user4'、'user5'、'user6'的值

```
mget 'user3' 'user4' 'user5' 'user6'
```

删除

- 详见下节键的操作，删除键时会将值删除

键命令

- 查找键，参数支持正则表达式

```
keys pattern
```

- 例1：查看所有键

```
keys *
```

- 例2：查看名称中包含a的键

```
keys '*a*'
```

- 判断键是否存在，如果存在返回1，不存在返回0

```
exists key1
```

- 例3：判断键'user1'、'user2'是否存在

```
exists 'user1'
exists 'user2'
```

- 查看键对应的value的类型

```
type key
```

- 例4：查看键'user1'的值类型，为redis支持的五种类型中的一种

```
type 'user1'
```

- 删除键及对应的值

```
del key1 key2 ...
```

- 例5：删除键'user3'、'user4'、'user5'、'user6'

```
del 'user3' 'user4' 'user5' 'user6'
```

- 设置过期时间，以秒为单位
- 如果没有指定过期时间则一直存在，直到使用DEL移除

```
expire key seconds
```

- 例6：设置键'user2'的过期时间为10秒

```
expire 'user2' 10
```

- 查看有效时间，以秒为单位

```
ttl key
```

- 例7：查看键'user2'的有效时间

```
ttl 'user2'
```

hash

- hash用于存储键值对集合
- 值的类型为string
- 键可以理解为属性，一个属性对应一个值

增加、修改

- 设置单个属性

```
hset key field value
```

- 例1：设置键'huser1'的属性'name'为'lb'

```
hset 'huser1' 'name' 'lb'
```

- 设置多个属性

```
hmset key field1 value1 field2 value2 ...
```

- 例2：设置键'huser2'的属性'name'为'wzj'、属性'gender'为'nv'、属性'birthday'为'2017-1-1'

```
hmset 'huser2' 'name' 'wzj' 'gender' 'nv' 'birthday' '2017-1-1'
```

获取

- 获取指定键所有的属性

```
hkeys key
```

- 例3：获取键'huser2'的所有属性

```
hkeys 'huser2'
```

- 获取一个属性的值

```
hget key field
```

- 例4：获取键'huser2'属性'name'的值

```
hget 'huser2' 'name'
```

- 获取多个属性的值

```
hmget key field1 field2 ...
```

- 例5：获取键'huser2'属性'name'、'gender'、'birthday'的值

```
hmget 'huser2' 'name' 'gender' 'birthday'
```

- 获取所有属性的值

```
hvals key
```

- 例6：获取键'huser2'所有属性的值

```
hvals 'huser2'
```

例7：获取键'huser2'所有属性和所有值

```
hgetall 'huser2'
```

删除

- 删除整个hash键及值，使用del命令
- 删除属性，属性对应的值会被一起删除

```
hdel key field1 field2 ...
```

- 例8：删除键'huser2'的属性'gender'

```
hdel 'huser2' 'gender'
```

list

- 链表中存的数据类型为string
- 按照添加的顺序排序
- list是双向链表

增加

- 在左侧插入数据

```
lpush key value1 value2 ...
```

- 例1：从键为'luser1'的列表左侧加入数据'dq'、'xq'

```
lpush 'luser1' 'dq' 'xq'
```

- 在右侧插入数据

```
rpush key value1 value2 ...
```

- 例2：从键为'luser1'的列表右侧加入数据'lb'、'zf'

```
rpush 'luser1' 'lb' 'zf'
```

- 在指定元素的前或后插入新元素

```
linsert key before或after 现有元素 新元素
```

- 例3：在键为'luser1'的列表中元素'lb'前加入'wzj'

```
linsert 'luser1' before 'lb' 'wzj'
```

获取

- 返回列表里指定范围内的元素
  - start、stop为元素的下标索引
  - 索引从左侧开始，第一个元素为0
  - 索引可以是负数，表示从尾部开始计数，如-1表示最后一个元素

```
lrange key start stop
```

- 例4：获取键为'luser1'的列表所有元素

```
lrange 'luser1' 0 -1
```

修改

- 设置指定索引位置的元素值
  - 索引从左侧开始，第一个元素为0
  - 索引可以是负数，表示尾部开始计数，如-1表示最后一个元素

```
lset key index value
```

- 例5：修改键为'luser1'的列表中下标为1的元素值为'kai'

```
lset 'luser1' 1 'kai'
```

删除

- 删除指定元素
  - 将列表中前count次出现的值为value的元素移除
  - count >0: 从头往尾移除
  - count <0: 从尾往头移除
  - count = 0: 移除所有有

```
lrem key count value
```

- 例6.1：向列表'luser2'中加入元素'h0'、'h1'、'h2'、'h0'、'h1'、'h3'、'h0'、'h1'

```
rpush 'luser2' 'h0' 'h1' 'h2' 'h0' 'h1' 'h3' 'h0' 'h1'
```

- 例6.2：从'luser2'列表右侧开始删除2个'h0'

```
lrem 'luser2' -2 'h0'
```

set

- 无序集合
- 元素为string类型
- 元素具有唯一性，不重复
- 说明：对于集合没有修改操作

增加

- 添加元素

```
sadd key member1 member2 ...
```

- 例1：向键'suser1'的集合中添加元素'dq'、'xq'、'lb'

```
sadd 'suser1' 'dq' 'xq' 'lb'
```

获取

- 返回所有的元素

```
smembers key
```

- 例2：获取键'suser1'的集合中所有元素

```
smembers 'suser1'
```

删除

- 删除指定元素

```
srem key member
```

- 例3：删除键'suser1'的集合中元素'xq'

```
srem 'suser1' 'xq'
```

zset

- sorted set，有序集合
- 元素为string类型
- 元素具有唯一性，不重复
- 每个元素都会关联一个分数，分数可以为负数，通过分数将元素从小到大排序
- 说明：没有修改操作

增加

- 添加

```
zadd key score1 member1 score2 member2 ...
```

- 例1：向键'zuser1'的集合中添加元素'dq'、'xq'、'lb'、'bq'，分数分别为1、5、8、3

```
zadd 'zuser1' 1 'dq' 5 'xq' 8 'lb' 3 'bq'
```

获取

- 返回指定范围内的元素
  - start、stop为元素的下标索引
  - 索引从左侧开始，第一个元素为0
  - 索引可以是负数，表示从尾部开始计数，如-1表示最后一个元素

```
zrange key start stop
```

- 例2：获取键'zuser1'的集合中所有元素

```
zrange 'zuser1' 0 -1
```

- 返回score值在min和max之间的元素

```
zrangebyscore key min max
```

- 例3：获取键'zuser1'的集合中分数在4和9之间的元素

```
zrangebyscore 'zuser1' 4 9
```

- 返回成员member的score值

```
zscore key member
```

- 例4：获取键'zuser1'的集合中元素'lb'的分数

```
zscore 'zuser1' 'lb'
```

删除

- 删除指定元素

```
zrem key member1 member2 ...
```

- 例5：删除集合'zuser1'中元素'dq'

```
zrem 'zuser1' 'dq'
```

- 删除分数在指定范围的元素

```
zremrangebyscore key min max
```

- 例6：删除集合'zuser1'中分数在4、9之间的元素

```
zremrangebyscore 'zuser1' 4 9
```