Redis使用总结

**一、利用redis缓存数据创建与mysql一致的数据**

先演示下实现mysql表中与redis数据统一的效果，看一下mysql中数据结构设计和redis有些什么区别

1.有如下两张数据表

CREATE TABLE `user` (

  `id` int(10) NOT NULL AUTO\_INCREMENT,

  `username` varchar(20) NOT NULL COMMENT '用户名',

  `password` varchar(25) NOT NULL COMMENT '密码',

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8 COMMENT='用户信息表';

CREATE TABLE `archives` (

  `id` int(10) NOT NULL AUTO\_INCREMENT,

  `userid` int(10) NOT NULL COMMENT '用户id',

  `username` varchar(20) NOT NULL COMMENT '发布微博的用户名称',

  `post\_time` datetime NOT NULL COMMENT '发布时间',

  `content` varchar(200) NOT NULL COMMENT '微博内容',

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8 COMMENT='用户发布的微博信息表';

2.mysql表数据如下

user表数据如下

mysql> select \* from user;

+----+----------+----------+

| id | username | password |

+----+----------+----------+

|  1 | dongzi   | 123456   |

|  2 | lidong   | 123456   |

+----+----------+----------+

archives表数据如下

mysql> select \* from archives;

+----+--------+----------+---------------------+--------------------------------------------+

| id | userid | username | post\_time           | content                                    |

+----+--------+----------+---------------------+--------------------------------------------+

|  1 |      1 | dongzi   | 2016-06-22 16:00:40 | 我是用户东子，我的第一条微博               |

|  2 |      2 | lidong   | 2016-06-22 16:10:40 | 我是用户lidong，我的第一条微博             |

|  3 |      2 | lidong   | 2016-06-22 17:05:40 | 我是用户lidong，我的第二条微博             |

+----+--------+----------+---------------------+--------------------------------------------+

3.使用redis增加user表和archives数据

在user表中增加第一个用户

incr global:user    --让用户表id自增，此时id为1，为了实现mysql中的auto\_increment

set user:userid:1:username dongzi

set user:userid:1:password 123456

set user:username:dongzi:userid 1 --设置这个key是为了利用用户名来反查用户id

在user表中增加第二个用户

incr global:user                    --现在返回的值为2

set user:userid:2:username lidong

set user:userid:2:password 123456

set user:username:lidong:userid 2 --设置这个key是为了利用用户名来反查用户id

发布第一条微博

incr global:archives    ​    ​--记录archives表的id

hset archives:id:1 userid 1

hset archives:id:1 username dongzi

hset archives:id:1 post\_time '2016-06-22 16:00:40'

hset archives:id:1 content '我是用户东子，我的第一条微博'

lpush archives:userid:1 1 --记录用户id发布的所有微博id

发布第二条微博

incr global:archives

hset archives:id:2 userid 2

hset archives:id:2 username lidong

hset archives:id:2 post\_time '2016-06-22 16:14:34'

hset archives:id:2 content '我是用户lidong，我的第一条微博'

lpush archives:userid:2 2

发布第三微博

incr global:archives

hset archives:id:3 userid 2

hset archives:id:3 username lidong

hset archives:id:3 post\_time '2016-06-22 17:05:40'

hset archives:id:3 content '我是用户lidong，我的第二条微博'

lpush archives:userid:2 3

最终我们使用redis得到了与mysql一致的数据，只是redis中对应关系会有一些区别

**二、根据具体业务逻辑，合理设置失效时间**

**三、高频，低频数据要分离**

高频率查询的数据存入redis缓存，低频数据不要存入redis缓存

**四、合理使用redis的数据类型**

例如：

1.使用hash类型：

hset archives:id:1 userid 1

hset archives:id:1 username dongzi

我们使用hash类型来存储，可以达到类似mysql中一条数据中的信息，这样redis中每条信息就只有一个key，能够大量的减少key的数量  
  
2.使用list类型  
lpush archives:userid:2 3  
创建一个key，list类型来表示userid为2的所有发布的微博id，这样就可以将微博id与用户id绑定了

**五、尽量使用字符串格式**  
业务逻辑满足，尽量使用字符串格式，读取时也方便读取查看

**六、可以使用incr实现自增id来达到类似mysql的auto\_increment功能**

例如：

incr global:user

每次incr一下自增一次，插入新数据的时候带上这个自增后的id值

**七、合理设置key的格式**

优点：这样的好处就是方便体现数据之间的关系，有层次化，方便管理。

例如：

set user:userid:2:username lidong  
表名放在第一位，用户id放在第二位，需要查询的字段放在第三位，然后使用冒号拼接凸显出逻辑关系，这个只是根据上面的表设计的，具体业务逻辑具体实现。

**八、创建一个“冗余”的key来反查数据**  
​例如：  
set user:username:lidong:userid 2  
设置这个key是为了利用用户名来反查用户id，我们知道用户名为lidong就可以知道用户id为2，相应的就可以获取用户id为2的所有信息