**Lock设计草案**

# 接口类

Namespace Exist

{

Class Mutex

{

Mutex( char \*name );

Virtual ~Mutex();

Void Lock();

Void Unlock();

Uint64 m\_key;//锁在exist上也是一个数据，既然是数据，那么在key-value系统中所有数据都需要拥有一个key，由name通过hash计算得到

}

}

# 实现方案1——Exist实现方式

## 锁的创建

构造函数，使用name计算出Mutex对象的key即可

不必发通知到exist上创建数据，等待第一次要Lock()时Exist自己会根据数据类型创建

## 加锁过程

## Lock()

向exist请求，将m\_key的Mutex类型数据设置为1

并以阻塞方式接收回应

## Exist响应

检查数据类型，进入对应数据类型的处理，这里会进入Mutex类型数据处理流程

先取出数据，检查数据是否为0

If 数据不存在或者为0，表示可以加锁，将数据设置未1，立刻回复

Else 将链接保存到等待列表，Lock()由于是阻塞方式接收回应，所以会一直等待，

也就是被Lock()住了

## 解除过程

## UnLock()

向exist请求，将m\_key的Mutex类型数据设置为1

立刻返回，不接收回应，解锁完成

## Exist响应

检查数据类型，进入对应数据类型的处理，这里会进入Mutex类型数据处理流程

检查等待列表中有没有线程在等待Lock()，

If 没有,则将数据设置为0，返回

Else 从列表头（最早开始等待的线程）拿出链接，回应，Lock()成功，数据就不必修改了（避免了数据查询和修改），因为又加锁了，又要设置为1

## 对象释放

析构函数，向exist请求删除key=m\_key的数据

## 实现方案2——Redis实现方式锁的创建

构造函数，使用name计算出Mutex对象的key即可

向redis请求查询key=m\_key的数据，

If 成功，什么也不敢

Else 设置0？不行，如果这时另外一个线程先完成了lock()将数据设置为了1，这里不是=解锁了。那么之后的Lock()不是可以成功？不是并发了？

Redis问题1：想不到好办法创建数据

此问题暂且搁置，假设设个问题解决了，下面继续Lock()/Unlock()的实现

## 加锁过程

## Lock()

* 方式一

向exist请求，将m\_key的Mutex类型数据设置为1

以阻塞方式接收回应？

不行，不管数据是否存在，值是否为0，redis是一定会立刻回复的，所以此方法不能将线程阻塞住

* 方式2

先查询数据

If =0，设置为1，加锁成功？

多个线程一起查询都查到=0，不是并发了？redis 的这个过程本身不是原子的

Else 用一个信号把自己挂起，等待解锁完成？

不行redis就没有主动通知的功能

结论加锁过程也没办法实现,其它的也会遇到同样的问题，

就其原因就是redis的数据操作太简单，不能支持lock操作

## 实现方案——Exist&Redis搭配实现方式

将所有Mutex类型的数据保存在Exist上

其他类型redis可支持类型的数据保存在redis上

map vector 等各种类型redis是否可支持，待设计