额度管理需求分析与设计文档

需求分析

名词解释：null

PRD: null

额度管理模块

1. 支持多个额度账户管理，额度初始化
   用户或管理员可以创建额度，创建的额度包含内容：额度类型（如美元额度，人民币额度等等），额度总值，额度可用值，额度所属用户。
   同时用户或管理员可以删除额度，删除额度的前提是该用户没有使用该额度，即额度总值=额度可用

值。
限制：一个用户有多种类型的额度, 但同一种类型的额度不可以重复创建。

2. 支持额度增加
   用户可以恢复额度，此时增加可用额度值。（暂不支持一键调增额度总值，但可以通过先删除额度，再重新创建新的额度来实现调增额度总值的功能）。
   限制：增加后的额度可用值不超过额度总值。
3. 支持额度扣减
   用户可以使用额度，此时会扣减可用额度值。
   限制：用户不可以在可用额度值为0时使用额度。
4. 支持额度查询
   用户可以查询自己名下的所有额度信息。

性能目标

1. 4C8G支持xxx qps操作, xxx tps操作
2. openapi timeout最大值应在2秒内

技术/组件依赖

语言：Java15(openjdk version "15.0.2")
框架：springboot，mybatis, jpa(单测用例使用)
数据库：mysql8，h2(单测用例使用)

依赖组件：null

详细设计

时序图

null

OpenApi

openApi统一返回值：

```
{
    "code": 2004,
    "data": "",
    "msg": "user has the type quota",
    "logId": "4dab2c5f-bd0e-480f-9cc5-71b6501a16bf"
}
```

说明：

- code：请求状态码，code=0表示请求成功执行
- msg：请求失败时，会返回错误信息
- logId：全局logID，客户端若未生成，则服务端生成
- data：接口成功时返回的结构体，该系统返回三种data：
  - 无数据: ""
  - 额度:

```
{
    "id": 4,  // 额度ID
    "userId": 1, // 用户ID
    "type": "1", // 额度类型
    "avail": 1.1039974440924007, // 额度可用值
    "total": 10.0, // 额度总值
    "status": "Available", // 额度状态
    "createAt": "2024-07-11T15:08:17.000+00:00", // 额度创建时间
    "updateAt": "2024-07-13T23:03:36.000+00:00"  // 额度更新时间
}
```

  - 额度列表:

```
  [
{
    "id": 4,  // 额度ID
    "userId": 1, // 用户ID
    "type": "1", // 额度类型
    "avail": 1.1039974440924007, // 额度可用值
    "total": 10.0, // 额度总值
    "status": "Available", // 额度状态
    "createAt": "2024-07-11T15:08:17.000+00:00", // 额度创建时间
    "updateAt": "2024-07-13T23:03:36.000+00:00"  // 额度更新时间
},
    ...
]
```

查询用户配额

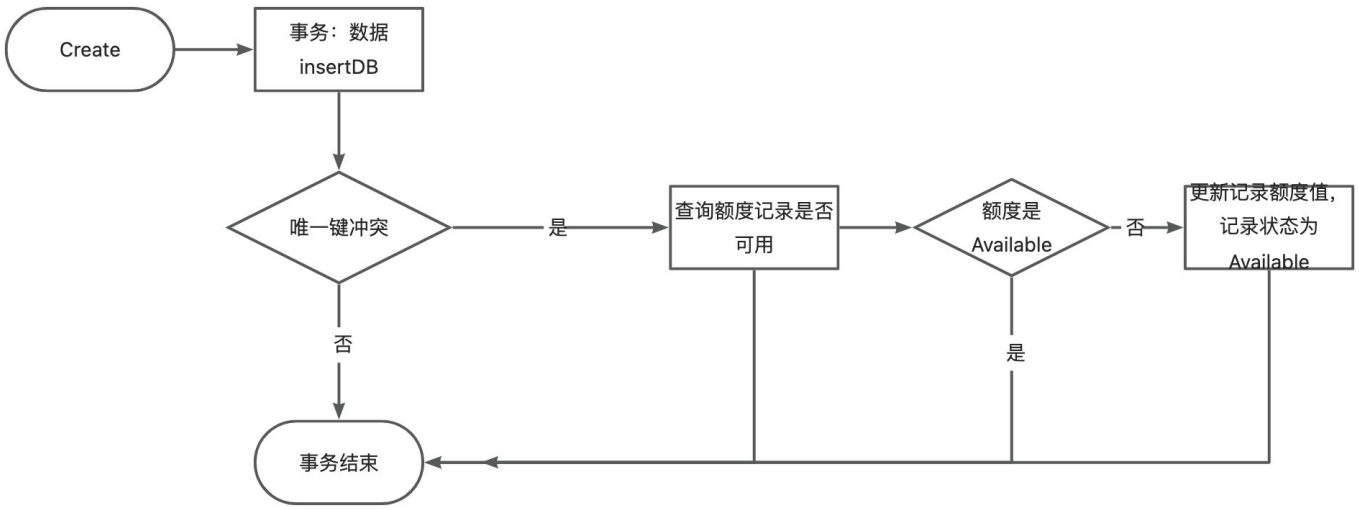
path: /get/{userId}/{type}, /get/{userId}。userId：用户ID，type：额度类型
type为空时返回用户的全部额度。
data返回值示例:

```
    [
        {
            "id": 4,  // 额度ID
            "userId": 1, // 用户ID
            "type": "1", // 额度类型
            "avail": 1.1039974440924007, // 额度可用值
            "total": 10.0, // 额度总值
            "status": "Available", // 额度状态
            "createAt": "2024-07-11T15:08:17.000+00:00", // 额度创建时间
            "updateAt": "2024-07-13T23:03:36.000+00:00"  // 额度更新时间
        }
       ]
```

创建用户配额

path: /create/{userId}/{type}/{total} userId: 用户ID，type：额度类型，total：额度总值。 参数均不可为空。

细节:



API接口返回错误码:

code	msg	data	出现场景

1000	internal err, please contact the admin		系统问题
0	""	{"id":4,"userId":1,"type":"1","avail":1.1039974440924007,"total":10.0,"status":"Available","createAt":"","updateAt":""}	成功创建额度
2001	the parameter is invalid, please check and try again		userId，type，total 参数不合法或为空
2004	user has the type quota		重复创建额度

删除用户配额

path: /delete/{id}, id为额度id。

细节：删除配额在db中软删除。

```
1   UPDATE quota SET `status` =  'Deleted' WHERE id = #{id} and avail = total
```

API接口返回错误码：

code	msg	data	出现场景
1000	internal err, please contact the admin		系统问题

0	""	""	成功删除额度
2001	the parameter is invalid, please check and try again		id参数不合法或为空
2002	the available quota has been used		用户已经使用quota，无法删除

增加用户额度

path: /add/{id}/{count} id: 配额id，count：增加的额度值

细节：增加额度要求不超过额度上限，使用db update 排他锁实现。

```
UPDATE quota SET avail = avail + #{count} WHERE id = #{id}  and `status` = 'Available' and total >= avail + #{count}
```

返回值：

code	msg	data	出现场景
1000	internal err, please contact the admin		系统问题
0	""	""	成功
2001	the parameter is invalid, please check and try again		id，count参数不合法或为空
2005	the qualifying quota does not exist		符合条件的额度不存在

减少用户额度

path: /reduce/{id}/{count} id: 额度id，count：扣减数量

细节：增加可用额度不会扣成负数，使用db update 排他锁实现。

```
UPDATE quota SET avail = avail - #{count} WHERE id = #{id} and avail >= #{count}  and `status` = 'Available
```

返回值：

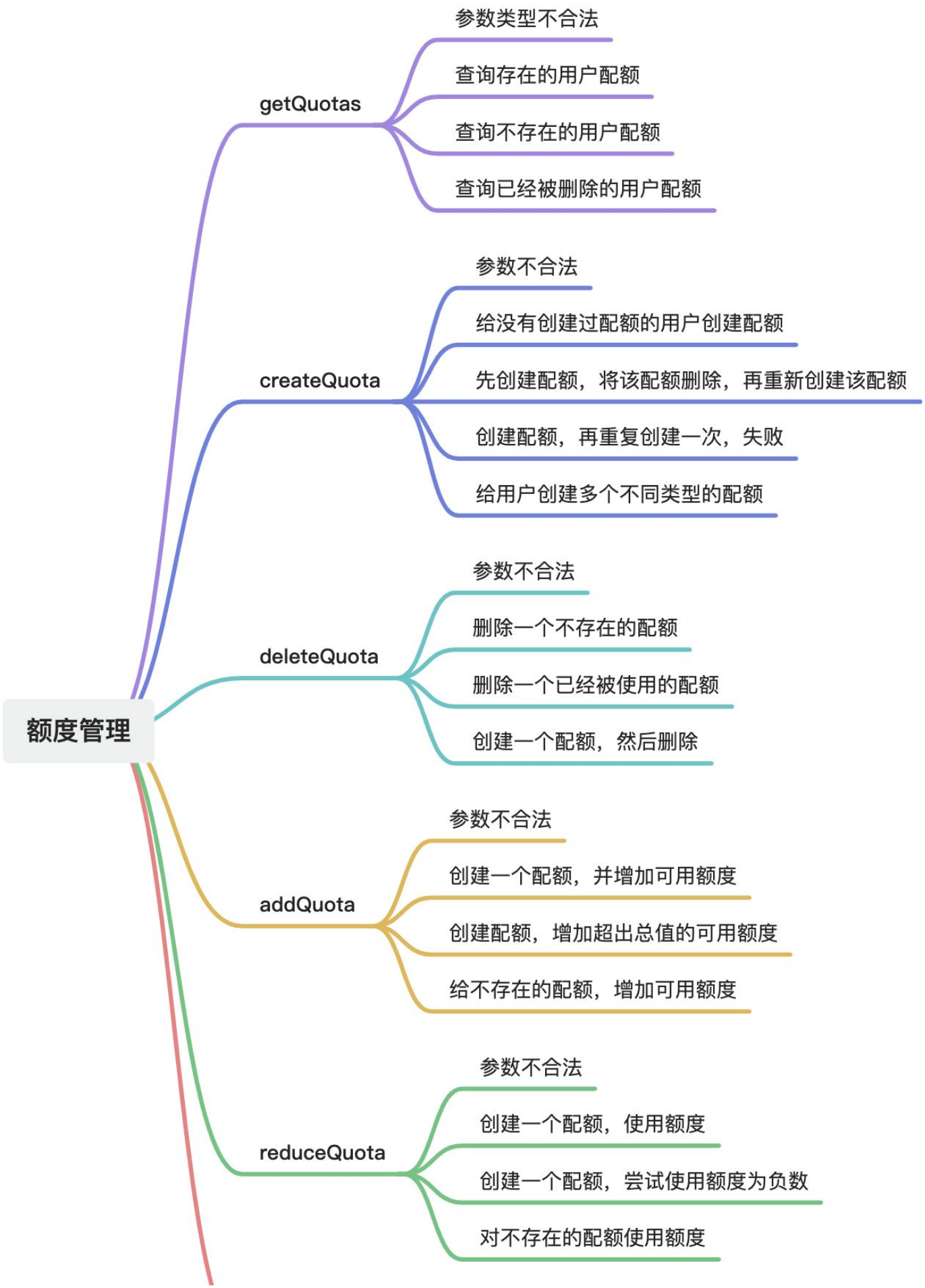
code	msg	data	出现场景
1000	internal err, please contact the admin		系统问题
0	""	""	成功
2001	the parameter is invalid, please check and try again		id，count参数不合法或为空
2003	the available quota is not enough		可用的额度不足

DB设计

```
CREATE TABLE `quota` (
                         `id` bigint unsigned NOT NULL AUTO_INCREMENT, -- 自增主键
                         `user_id` bigint NOT NULL, -- 用户id
                         `type` varchar(255) NOT NULL DEFAULT '', -- 额度类型
                         `avail` double NOT NULL, -- 可用额度值
                         `total` double NOT NULL, -- 额度总值
                         `status` varchar(255) NOT NULL DEFAULT '', -- 额度状态: Available, Deleted
                         `create_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
                         `update_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
                         PRIMARY KEY (`id`),
                         UNIQUE KEY `uni_user_type` (`user_id`,`type`)
);
```

测试用例

限制：使用h2数据库避免污染测试环境db。


额度管理
getQuotas
参数类型不合法
查询存在的用户配额
查询不存在的用户配额
查询已经被删除的用户配额
createQuota
参数不合法
给没有创建过配额的用户创建配额
先创建配额，将该配额删除，再重新创建该配额
创建配额，再重复创建一次，失败
给用户创建多个不同类型的配额
deleteQuota
参数不合法
删除一个不存在的配额
删除一个已经被使用的配额
创建一个配额，然后删除
addQuota
参数不合法
创建一个配额，并增加可用额度
创建配额，增加超出总值的可用额度
给不存在的配额，增加可用额度
reduceQuota
参数不合法
创建一个配额，使用额度
创建一个配额，尝试使用额度为负数
对不存在的配额使用额度

定时任务	每10s，10个用户并发操作随机额度

可运维

1. 请求生成logid：用于追踪和幂等
2. api response带上logid
3. 全局异常捕获，结果统一返回ApiResult
4. 日志可归档
5. 接口错误配置告警监控
6. 发布前完成api设置qps限制

回滚方案

null

参考文档

null