交易前台ACE---OMS（v0.0.2）

# 目录

[目录 1](#_Toc40558678)

[修订（开发）记录 4](#_Toc40558679)

[一． OMS总体设计 5](#_Toc40558680)

[1.1 整体架构 5](#_Toc40558681)

[1.2 术语解释 6](#_Toc40558682)

[二．OMS周边模块交互图 7](#_Toc40558683)

[三．OMS内部组件设计 8](#_Toc40558684)

[3.1 状态机 8](#_Toc40558685)

[3.2 数据完整性 8](#_Toc40558686)

[3.3 数据一致性 9](#_Toc40558687)

[3.3.1 mysql同步到ignite 9](#_Toc40558688)

[3.3.2 ignite同步到mysql 9](#_Toc40558689)

[3.4 线程池 10](#_Toc40558690)

[3.5 ignite内存设计 10](#_Toc40558691)

[四．OMS各模块功能设计 11](#_Toc40558692)

[4.1 通用模块 11](#_Toc40558693)

[4.1.1 服务初始化（日切、服务重启） 11](#_Toc40558694)

[4.1.2 异步持久化（待讨论） 11](#_Toc40558695)

[4.1.3 数据迁移备份 11](#_Toc40558696)

[4.1.4 异常恢复 11](#_Toc40558697)

[4.2 OMS-TRADE模块 12](#_Toc40558698)

[4.2.1 委托/撤单 12](#_Toc40558699)

[4.3 OMS-APC模块 14](#_Toc40558700)

[4.3.1 冻结资金/冻结股份—买单/卖单委托 14](#_Toc40558701)

[4.3.2 解冻资金/解冻股份—卖单/卖单撤销委托 15](#_Toc40558702)

[4.3.3 扣减（冻结中）资金并增加股份—买单成交 15](#_Toc40558703)

[4.3.4 增加资金并扣减（冻结中）股份—卖单成交 15](#_Toc40558704)

[4.3.5 增加资金/减少资金 16](#_Toc40558705)

[4.3.6 修改/新增账户 17](#_Toc40558706)

[4.3.7 冻结/解冻账户 18](#_Toc40558707)

[五．对外服务接口 18](#_Toc40558708)

[六．数据库设计 20](#_Toc40558709)

[6.1 OMS通用表 20](#_Toc40558710)

[6.1.1异常任务表（ace\_exception\_task） 20](#_Toc40558711)

[6.2 OMS-TRADE数据表 21](#_Toc40558712)

[6.2.1 订单表（ace\_trade\_order） 21](#_Toc40558713)

[6.2.2 订单历史表（ace\_trade \_order\_his） 22](#_Toc40558714)

[6.2.3 成交回报表（ace\_trade \_execution） 23](#_Toc40558715)

[6.2.4 成交回报历史表（ace\_trade \_execution\_his） 23](#_Toc40558716)

[6.3 OMS-APC表 23](#_Toc40558717)

[6.3.1 交易账户信息表（ace\_apc\_account） 23](#_Toc40558718)

[6.3.2 持仓信息表(ace\_apc\_position) 24](#_Toc40558719)

[6.3.3 资金信息表(ace\_apc\_cash) 25](#_Toc40558720)

[6.3.4 持仓详细流水表（ace\_apc\_position\_statement） 25](#_Toc40558721)

[6.3.5 资金详细流水表（ace\_apc\_cash\_statement） 26](#_Toc40558722)

# 修订（开发）记录

| 日期 | 版本 | 修订说明 | 修订者 |
| --- | --- | --- | --- |
| 2020.05.09 | v0.0.1 | 1. ACE-OMS详细设计起草 | 姜瀚 |
| 2020.05.16 | v0.0.1 | 将账户、持仓、资金管理放到OMS中实现 | 姜瀚 |

# OMS总体设计

## 1.1 整体架构

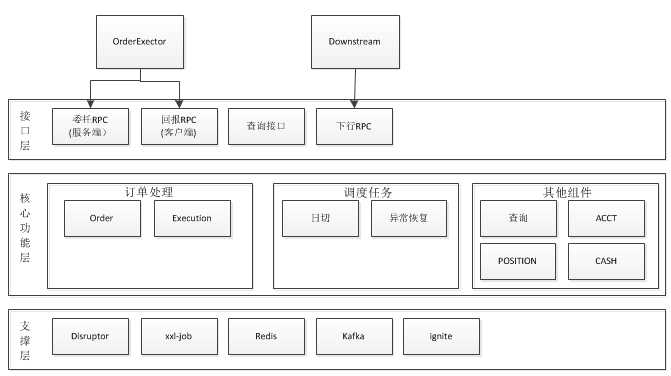


图1.1 ACE-OMS总体设计

Order Manager Service（OMS）是一个基于分布式内存数据库ignite的内存交易系统，其包含三大组件：订单引擎/账户持仓资金引擎/查询类引擎。

OMS可以工作在有状态模式、无状态模式和特定用户模式：当OMS处于无状态模式时，会处理交易和查询两种事务；当OMS处于有状态模式时，需要设定其是查询类OMS还是交易类OMS。有状态OMS可以和无状态OMS混用，但应尽量避免此种做法；当OMS处于特定用户模式时，其ignite将工作在local模式，即这个OMS没有加入到集群，它只为特殊用户进行服务。

一个无状态OMS担负以下职责：

1. 委托管理：交易中台的OrderExector与StrategyContainer是一对一关系的，所以OrderExector没有全局的视角；而OMS与OrderExector是多对一的，所有的订单都要经过OMS集群发送给下游
2. 回报管理：同上。
3. 管理持仓和现金
4. 委托查询/成交查询/账户查询/持仓查询/资金查询

## 1.2 术语解释

|  |  |
| --- | --- |
| **缩写、术语** | **解释** |
| OMS | Order Managerment System，负责处理订单逻辑 |
| OMS-APC | OMS的账户持仓资金子模块 |
| OMS-TRADE | OMS的委托回报子模块 |
| OMS-QUERY | OMS的对外查询暴露接口子模块 |
| OrderExecutor | KING的委托订单组件 |
| TradeExecutor | KING的成交回报处理组件 |

# 二．OMS周边模块交互图

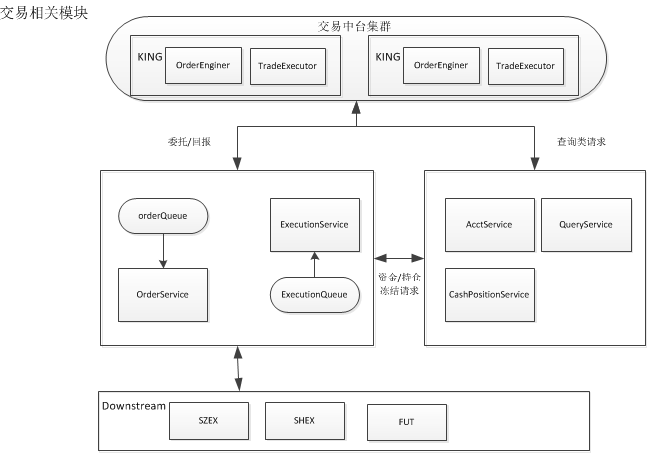


图2.1 OMS与周边模块的交互

# 三．OMS内部组件设计

## 3.1 状态机

订单状态机，包括Init(初始化订单)、New（新订单未报）、Pending(新订单已报)、Part-Exec（部分成交）、Full-Exec（完全成交）、Rejected（订单被拒绝）、Withdrawing-trade（已报交易所的订单撤单）、Withdraw（订单未成交，全部撤回）。

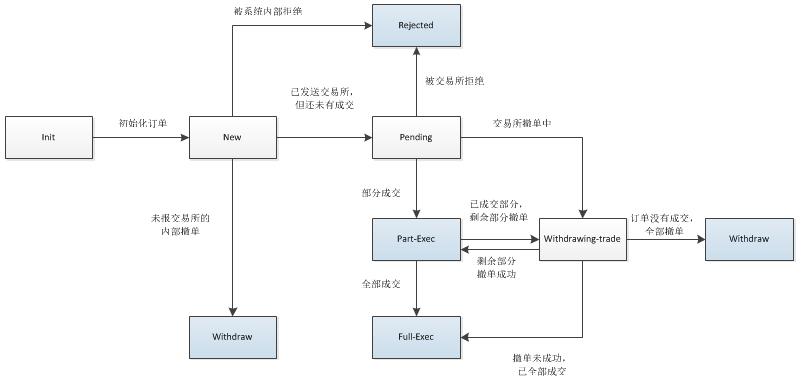


图3.1 订单状态机

深色为终态，除此以外都是中间态。

## 3.2 数据完整性

OMS是内存交易系统，所有的状态保存在内存ignite中，以此来支持极速交易。但内存交易系统在故障重启后数据会丢失，为了解决这个问题，我们采用异步持久化的方式将内存状态持久化到MySql中。是利用ignite自身特性进行持久化，还是另起线程做持久化mysql的操作，待进一步讨论以及性能验证。

异步持久化会引入另一个问题，系统故障瞬间，异步队列中的数据没有被成功持久化。目前只得通过手工恢复的方式解决，虽然会引入一些运维工作，但可以提升系统性能，简化系统设计。具体处理方式是，从委托或回报日志（如Downstream或OrderExecutor），重启前运维人员通过日志和数据库中记录的记录进行对比，如果有漏单或回报，则通过手工将这些数据导入到MySql中，确认OK后再重启系统。OMS的启动后，会把该笔订单重新加载到ignite

## 3.3 数据一致性

保证数据一致性，即mysql数据库与ignite的数据一致性。其一致性是双向的。即mysql数据发生变化，推送给ignite；和ignite数据变化，持久化到数据库。

### 3.3.1 mysql同步到ignite

分为启动时的同步，和运行时的热同步。正常情况，是不应该出现热同步的，这意味着发生了重大事故，运维人员不得不更改数据库的数据

1.启动时的同步：

会通过FAMS自身的特性将order/execution/account/cash/position加载到ignite

2.系统运行中的热同步：

方案1：trigger

方案2：在order/execution/account/cash/position中添加一个字段，用于标识此条记录需要同步

### 3.3.2 ignite同步到mysql

即内存的异步持久化：

方案1：通过ignite自身的特性

方案2：统一设计持久化线程池。order/execution/account/cash/position设计一个version字段，整个OMS集群在一个时间区段，所有对ignite进行的操作都要标上version；当线程池进行持久操作时，判断当前version是否需要持久化。例如：

当前线程池维护的需要持久化的version=16，此时ignite中有两条order一条position1记录：

|  |  |
| --- | --- |
| Ignite内存 | Version |
| order1 | 15 |
| order2 | 16 |
| position1 | 14 |

即表示order1在当前持久化周期内发生了变化，需要持久化到数据库。故在当前持久化的时间节点时，该条记录会update/insert进入mysql。同时ignite的当前version增加为17；如果当前周期没有需要持久化的数据，ignite-version不发生变化

理论上不会出现记录的version>ignite-verison的情况

## 3.4 线程池

需要一个可配置容量的线程池：

线程1：持久化数据库数据（也可以依赖ignite自己持久化，到时候验证性能再决定）

线程组2：读取order队列，RPC downstream进行委托下单

线程组3：读取executor队列，并RPC交易中台发送交易回报

线程4：异常恢复

线程组5：处理查询请求

## 3.5 ignite内存设计

每个OMS作为一个ignite实例，都将创建order/execution/account/cash/position五个cache；同时由于使用ignite的PARTITIONED模式，所以每一个OMS节点，都是ignite集群中的节点，对cache的操作，会同步到各个节点，从而保证OMS是一个无状态的系统。

即使是工作在有状态模式的OMS，将堵塞查询或交易通道，使得其内部线程池全部用于单一功能，提高OMS的处理能力

# 四．OMS各模块功能设计

## 4.1 通用模块

### 4.1.1 服务初始化（日切、服务重启）

MonitorService每日日终会强制所有服务下线重启（或脚本定时任务），OMS重启后，做以下事情：

1. 重新从全局配置表获取BussinessDate
2. 将ignite中的order和execution表的数据非当日的交易全部删除
3. 将ignite中的当天的order和execution表导入内存（即重启恢复）
4. 从mysql加载账户信息/持仓信息/资金信息到ignite

### 4.1.2 异步持久化（待讨论）

### 4.1.3 数据迁移备份

使用FAMS框架的迁移插件（待开发）

### 4.1.4 异常恢复

## 4.2 OMS-TRADE模块

### 4.2.1 委托/撤单

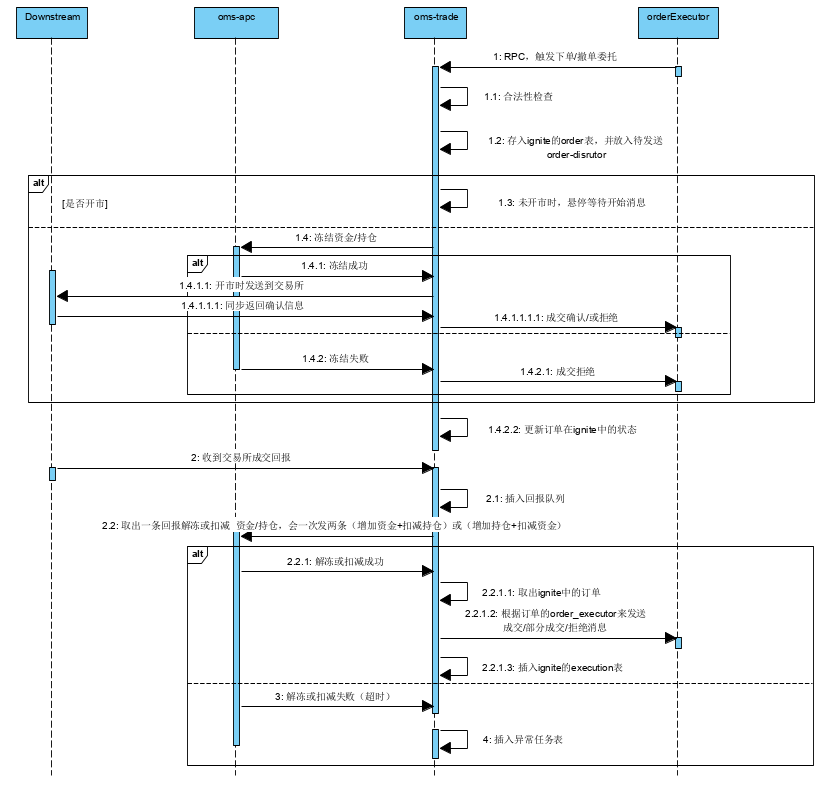


图4.1 委托/撤单

收到交易中台的订单后，做如下处理：

1. 写入order缓存队列
2. OMS判断当前是否是开市时间，如果已开市，轮询order线程持续等待；如果已开市，order线程调起缓存中的order。
3. 检查order是否已在ignite中存在，如果已经存在，即判断为重复单（可能是上游KING的bug），报警并丢弃；如果不存在的映射，调APC接口进行资金或持仓冻结操作；
4. 以便回报和交易确认的回调。
5. 调downstream，更改状态机为New，并写入ignite；如果发生异常，插入异常任务表，（对应order的状态变为exeception？，待讨论）

收到交易中台的撤单请求：

1. 写入order缓存队列
2. 找到内存中的订单（如果没有，要么是丢单，要么是服务重启，之前的数据持久化失败了，待讨论如何处理）
3. 如果是New状态，直接更新状态为Withdraw，内部撤单
4. 如果是Pending状态，向downstream发送撤单申请，如果发生异常，插入异常任务表（对应order的状态变为exeception？，待讨论）
5. 如果是完全成交，无需处理
6. 调交易中台的rpc通知撤单结果
7. 写入持久化的order队列

收到downstream的订单确认（已发送交易所，或者交易所拒绝，或部分成交）：

1. 写入execution缓存队列

2. execution线程调起queue缓存的结果

1. 如果是订单确认：更新状态为Pending，发送交易中台（也可不发送，待商量），更新ignite
2. 如果是订单拒绝：更新状态为Rejected，发送交易中台，更新ignite
3. 如果是成交报文：更新状态为full或part，发送交易中台，更新ignite
4. 写入持久化的execution队列

## OMS-APC模块

### 4.3.1 冻结资金/冻结股份—买单/卖单委托

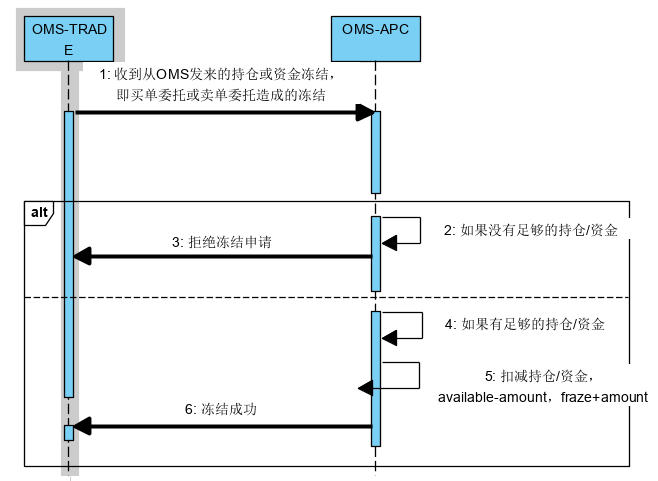


图4.3 APC的冻结资金/冻结股份接口

场景：APMS收到OMS的持仓或资金冻结请求：

1. 从ignite中取出对应账户的持仓或资金，如果没有对应的持仓和资金，直接返回失败
2. 冻结持仓：可用持仓 = 可用持仓–请求数量；冻结持仓 = 冻结持仓+请求数量

冻结资金：可用资金 =可用资金-请求金额；冻结资金 = 冻结资金+请求金额

1. 上述过程如无异常，更新ignite，并返回结果

### 4.3.2 解冻资金/解冻股份—卖单/卖单撤销委托

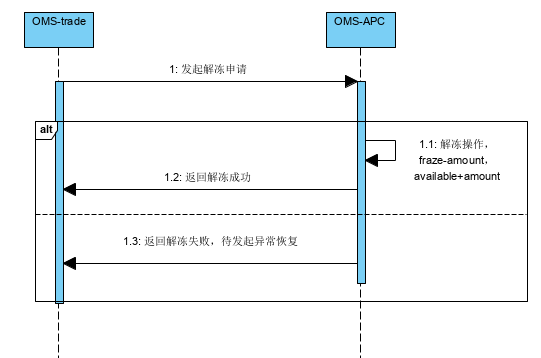


图4.4解冻资金/解冻股份—卖单/卖单撤销委托

场景：收到OMS的撤销委托时

解冻资金和解冻股份过程类似4.5过程。只是公式变为：

解冻持仓：可用持仓= 可用持仓+请求数量；冻结持仓=冻结持仓-请求数量

解冻资金：可用资金=可用资金+请求金额；冻结资金=冻结资金-请求金额

### 4.3.3 扣减（冻结中）资金并增加股份—买单成交

场景：OMS处理买单成交时

扣减（冻结中）资金并增加股份类似上述过程。只是公式变为：

持仓：冻结持仓=冻结持仓-请求数量；

T+1类：T1持仓= T1持仓+请求数量 T+0类：可用持仓=可用持仓+请求数量

资金：冻结资金=冻结资金-请求金额

### 4.3.4 增加资金并扣减（冻结中）股份—卖单成交

场景：OMS处理卖单成交时

增加资金并扣减（冻结中）股份类似上述过程。只是公式变为：

持仓：冻结持仓=冻结持仓-请求数量；

资金：可用资金=可用资金+请求金额；

### 4.3.5 增加资金/减少资金

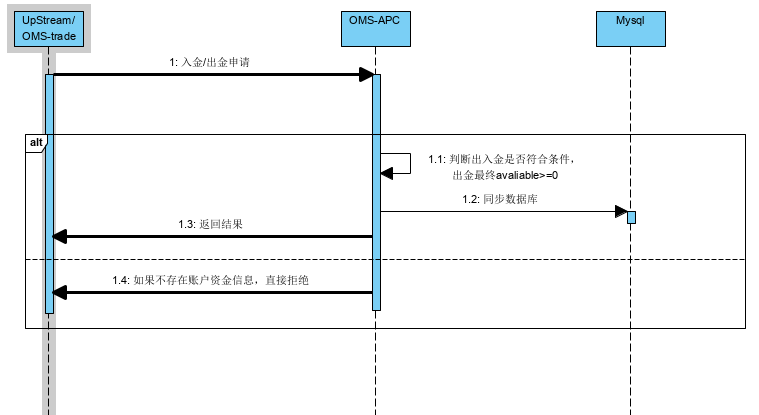


图4.5 入金/出金情景

场景：交易员从对交易账号的资金进行操作（交易账户和对公账户之间的交割）

1. 收到入金出金请求，检查ignite的账户资金信息
2. 如果存在对应的信息，如果是出金，需要判断资产是否足够；如果是入金直接在可用资金上增加。如果不存在资金信息，拒绝申请
3. 强制同步到mysql，通过mysql->ignite的机制最终把数据同步到ignite

### 4.3.6 修改/新增账户

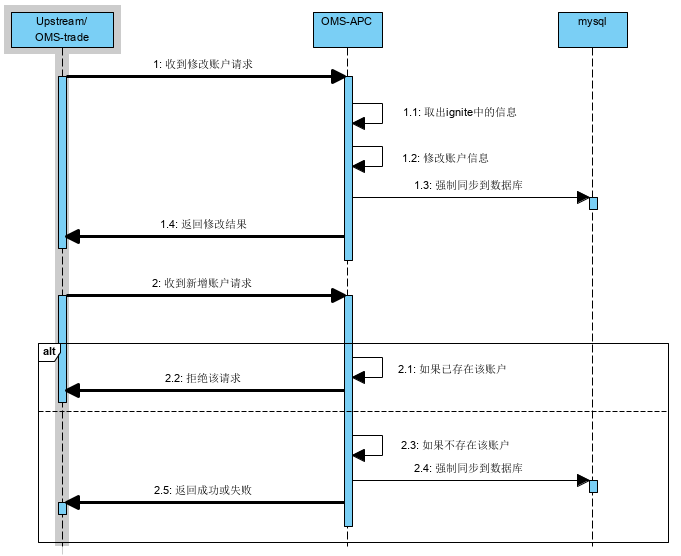


图4.6 修改/新增交易账户

场景：前台Upstream发起的修改交易账户或新增账户请求

修改：

1. 取出ignite中的账户信息。
2. 修改账户信息后，同步到mysql
3. 返回前台结果
4. 通过mysql->ignite的机制最终把数据同步到ignite

新增：

1. 检查是否有重复账户
2. 若没有，新建账户对应的资金信息，新建账户信息，同步到mysql
3. 返回前台结果
4. 通过mysql->ignite的机制最终把数据同步到ignite

### 4.3.7 冻结/解冻账户

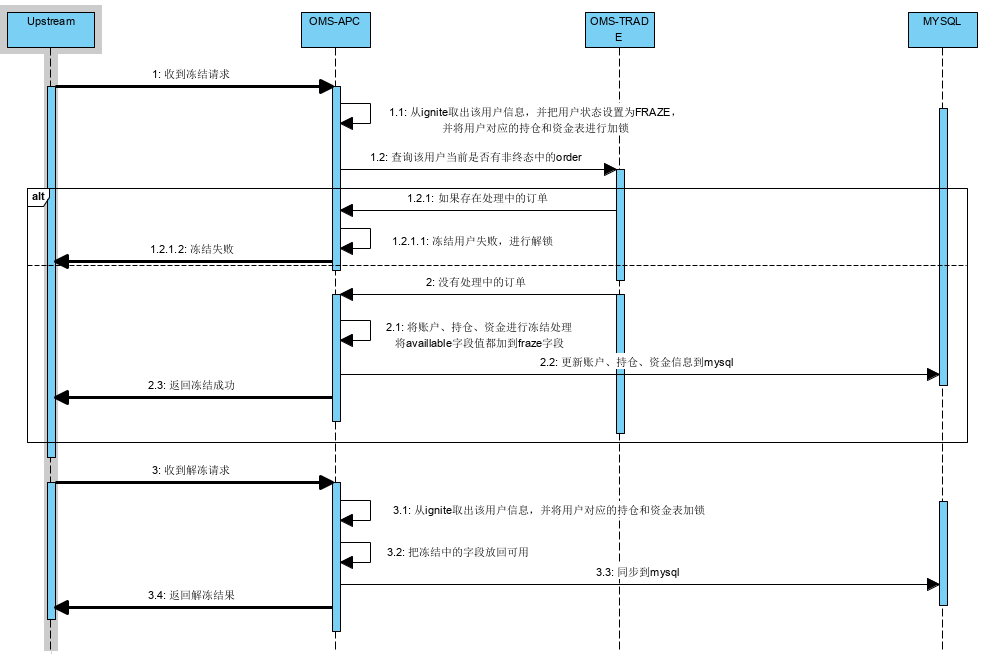


图4.7 冻结/解冻账户

场景：前台发起的冻结解冻交易账户申请

冻结：

1. 从ignite中取出账户信息，并将其状态设置为freeze
2. 向OMS发起查询，查询当前账户是否有处于非终态的委托，如果有，则返回前台冻结失败，将其状态改为active
3. 如果所有委托都已完成，将账户信息的status标志设置为freeze，将资金、持仓表所有数据都从available状态移到freeze状态。同步到mysql的账户、资金、持仓表

# 五．对外服务接口

|  |  |  |  |
| --- | --- | --- | --- |
| 方法名 | 描述 | 调用方 | 参数描述 |
| NewOrder | 提供给上游的普通委托回调接口 | KING | 参考order表 |
| ExecutionOrder | 提供给下游的成交回报回调接口 | Downstream | 参考execution表 |
| QueryOrder | 查询当日委托、历史委托 | Upstream、KING | 自行设计 |
| QueryExecution | 查询成交、历史成交 | Upstream、KING | 自行设计 |
|  |  |  |  |

Ps：未来还需要支持组合单

# 六．数据库设计

## 6.1 OMS通用表

### 6.1.1异常任务表（ace\_exception\_task）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| order\_id | varchar(64) | 是 | 订单编号 |
| account | varchar(32) | 是 | 交易账户 |
| security\_code | varchar(16) | 是 | 证券标的 |
| exch | varchar(8) | 是 | 交易市场 |
| exception\_type | varchar(8) | 是 | 需要讨论 |
| order\_status | varchar(16) | 否 | 订单状态机中的状态：  Init(初始化订单)、New（新订单未报）、Pending(新订单已报)、Part-Exec（部分成交）、Full-Exec（完全成交）、Rejected（订单被拒绝）、Withdrawing-trade（已报交易所的订单撤单）、Withdraw（订单未成交，全部撤回） |
|  |  |  |  |
| count | int | 是 | 重试次数，初始为0 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

## 6.2 OMS-TRADE数据表

### 6.2.1 订单表（ace\_trade\_order）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| order\_id | varchar(64) | 是 | 订单编号 |
| related\_order\_id | varchar(64) | 否 | 关联订单单号，撤单时填写 |
| exchange\_order\_id | varchar(64) | 否 | 交易所订单编号 |
| account | varchar(32) | 是 | 交易账户 |
| security\_code | varchar(16) | 是 | 证券标的 |
| exch | varchar(8) | 是 | 交易市场 |
| order\_status | varchar(16) | 是 | 订单状态机中的状态：  Init(初始化订单)、New（新订单未报）、Pending(新订单已报)、Part-Exec（部分成交）、Full-Exec（完全成交）、Rejected（订单被拒绝）、Withdrawing-trade（已报交易所的订单撤单）、Withdraw（订单未成交，全部撤回） |
| order\_reject\_code | varchar(16) | 否 | 填写拒单代码：  交易所拒单：EXCH\_REJECT  OMS拒单：OMS\_REJECT |
| order\_reject\_reason | varchar(255) | 否 | 拒单具体原因 |
| trade\_time | datetime | 是 | 交易时间 |
| order\_executor | varchar(32) | 是 | 订单来源名称，对应king的orderexecutor组件。用于回报等信息 |
| strategy\_container | varchar(32) | 否 | 如果是策略单，需要 |
| order\_price | bigdecimal(7,2) | 是 | 委托价格 |
| order\_volume | long | 是 | 委托量，以股为单位 |
| order\_type | varchar(16) | 是 | 订单类型：limit、fak、fok等 |
| order\_multiply | long | 是 | 证券乘数，例如1手=100股，此处为100 |
| order\_direction | varchar(8) | 是 | 委托方向：BUY/SELL/LONG/SHORT |
| exec\_price | bigdecimal(7,2) | 否 | 成交均价 |
| exec\_volume | long | 否 | 成交数量 |
| exec\_status | varchar(16) | 是 | 订单执行状态：  OPEN：等待执行，处于new、pengding、part-exec、withdraw-trade的订单，即还未完全执行结束的委托  CLOSE：执行完成，处于full-exec、Rejected、withdraw的订单，即已经结束的委托 |
| withdraw\_flag | varchar(2) | 否 | 撤单标志 |
| withdraw\_volume | long | 否 | 撤单数量 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

### 6.2.2 订单历史表（ace\_trade \_order\_his）

同订单表（ace\_trade\_order）

### 6.2.3 成交回报表（ace\_trade \_execution）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| order\_id | varchar(64) | 是 | 订单编号 |
| execution\_id | varchar(64) | 是 | 成交编号 |
| account | varchar(32) | 是 | 交易账户 |
| security\_code | varchar(16) | 是 | 证券标的 |
| exch | varchar(8) | 是 | 交易市场 |
| exec\_time | datetime | 是 | 成交回报时间 |
| exec\_price | bigdecimal(7,2) | 否 | 执行价格，如果是撤单的成交，可能没有价格 |
| exec\_volume | long | 是 | 成交数量 |
| withdraw\_flag | varchar(2) | 是 | 撤单标志。默认为N |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

### 6.2.4 成交回报历史表（ace\_trade \_execution\_his）

同成交回报（ace\_trade\_execution）

## 6.3 OMS-APC表

### 6.3.1 交易账户信息表（ace\_apc\_account）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| account | varchar(32) | 是 | 交易账户 |
| account\_type | varchar(16) | 是 | 交易账户类型：common普通账户 |
| account\_name | varchar(16) | 是 | 交易账户持有人姓名 |
| account\_jobid | varchar(16) | 是 | 交易账户持有人工号 |
| account\_level | int | 是 | 交易账户等级，不同等级对应不同的交易品种 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

### 6.3.2 持仓信息表(ace\_apc\_position)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| account | varchar(32) | 是 | 交易账户 |
| security\_id | varchar(16) | 是 | 证券id |
| security\_name | varchar(32) | 是 | 证券名称 |
| exch | varchar(8) | 是 | 交易所 |
| total\_qty | long | 是 | 持有总量=可用数量+冻结数量+在途数量 |
| available\_qty | long | 是 | 可用数量 |
| freeze\_qty | long | 是 | 冻结数量(卖出时冻结) |
| Onway\_qty | Long | 是 | 在途数量（买入时在途 T+1） |
| ave\_cost\_price | decimal(7,2) | 是 | 持仓成本均价 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

### 6.3.3 资金信息表(ace\_apc\_cash)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| account | varchar(32) | 是 | 交易账户 |
| currency | varchar(3) | 是 | 资金币种 |
| total\_amount | decimal(7,2) | 是 | 总金额=可用金额+冻结金额+在途金额 |
| available\_ amount | decimal(7,2) | 是 | 可用金额 |
| freeze\_ amount | decimal(7,2) | 是 | 冻结金额 |
| Onway\_amount | decimal(7,2) | 是 | 在途金额 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

### 6.3.4 持仓详细流水表（ace\_apc\_position\_statement）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| account | varchar(32) | 是 | 交易账户 |
| related\_order\_id | varchar(32) | 是 | 动作发生的关联订单号 |
| security\_id | varchar(16) | 是 | 证券id |
| security\_name | varchar(32) | 是 | 证券名称 |
| exch | varchar(8) | 是 | 交易所 |
| change\_type | varchar(16) | 是 | 包括冻结freeze、解冻unfreeze、出库out、入库in(T+0)、freeze-in入库但不可用(T+n) |
| change\_qty | long | 是 | 变动数量 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |

### 6.3.5 资金详细流水表（ace\_apc\_cash\_statement）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 是否必填 | 描述 |
| id | bigint | 是 | 主键 |
| account | varchar(32) | 是 | 交易账户 |
| currency | varchar(3) | 是 | 资金币种 |
| change\_type | varchar(16) | 是 | 包括冻结freeze、解冻unfreeze、出库out、入库in(T+0)、freeze-in入库但不可用(T+n) |
| change\_amount | decimal(7,2) | 是 | 变动金额 |
| modify\_time | datetime | 是 | 修改时间 |
| create\_time | datetime | 是 | 创建时间 |