# **SCQL最佳实践**

## **使用SCQL**

目前只开放了API：

1. 通过SCDBClient体验（一个类似mysql的client工具）
2. 基于SCQL API封装白屏产品，或者集成到业务中

所有的请求都是query形式，只有两类接口：scql/api/scdb\_api.proto

1. 用户异步提交查询，然后轮询结果（Submit & Fetch）
2. 同步提交query等待返回(SubmitAndGet)

2适合执行时间短的query，例如ddl，dcl以及小规模的dql语句

1适合大规模dql查询

流程：

部署系统 -> 项目设置 -> 联合分析

项目设置：

创建用户和DB（项目）

授权用户

创建表

设置CCL

## **部署**

镜像：secretflow/scql

docker pull secretflow-registry.cn-hangzhou.cr.aliyuncs.com/secretflow/scql:0.8.1b1

启动SCDB：/home/admin/bin/scdbserver -config=/path/to/config.yml

启动SCQLEngine: /home/admin/bin/scqlengine -flagfile=/path/to/gflags.conf

## **使用方案**

假设有这么一个场景，金融机构alice想要联合一个电商平台bob做联合用户的画像分析

alice希望能够统计不同信用等级，年龄在20岁到30岁之间，并且是电商平台bob的活跃用户的人数、以及平均收入和平均交易金额。

Alice有表格user\_credit(id, credit\_rank, income, age)

Bob有表格user\_stats(id, order\_amout, is\_active)

### **SCQL概念**

用户每次执行query，都需要提供这些信息：

project: 多个参与方协商后加入同一个项目

参与方身份认证：query执行时会进行

数据表管理：管理参与分析的数据表的schema信息

权限管理：表字段的权限，特别是CCL

为了降低上手成本，sf开发了SCDB(SCQL as a Database)，遵循传统dbms的概念

SCDB的相关概念：

Database: 对应一个项目。

创建/删除项目就CREATE/DROP DATABASE db\_test

User: 每个用户唯一对应一个参与方

CREATE USER alice PARTY\_CODE ‘ALICE’ IDENTIFIED BY ‘alice123’

Table: 物理表在db的映射（别名）

CREATE TABLE db\_test.ta(id string, col1 int64)

REF\_TABLE=db1.tbl\_1 DB\_TYPE=’mysql’

Privilege: 普通权限和CCL管理

GRANT SELECT PLAINTEXT\_AFTER\_JOIN(id) on db\_test.ta to

### **具体流程**

SCDB内置了一个权限受限的root用户，方便系统进行bootstrap

只能做CREATE database/user的操作

无法修改普通用户设置的ccl，无法发起查询(DQL)

权限滥用也不会导致数据泄露。

以下五部通过SCQLClient完成。也可以尝试postman、curl发起请求

第一步是创建用户，也就是建立用户和机构身份的绑定关系

通过party code字段建立机构身份的映射

两个机构分别创建用户：

可以通过root账号创建

可以通过委托root账号管理者创建（可信第三方，自己可以改密码）

案例：(公钥和签名可以防止伪造身份攻击)

root> CREATE USER `alice` PARTY\_CODE ‘Alice’ IDENTIFIED BY ‘alice123’

WITH ‘2023-08-23T29:03:34.26…+08:00’

Signature

Public key

ENDPOINT ‘alice-engine-host:port’

第二步：创建项目&用户授权。

创建一个名为db\_test的项目（数据库）

CREATE DATABASE db\_test

把合作方添加到项目中(alice和bob可以在项目中创建、删除表，进行授权)

root> GRANT CREATE, GRANT OPTION, DROP ON db\_test.\* TO alice

第三步：创建表，方便导入联合分析

alice> CEREATE TABLE db\_test.ta

(ID string,

credit\_rank int,

income int,

age int)

REF\_TABLE=alice.user\_credit DB\_TYPE=’mysql’;

bob> CEREATE TABLE db\_test.tb

(ID string,

order\_amount double,

is\_active int)

REF\_TABLE=bob.user\_stats DB\_TYPE=’mysql’

第四步：设置CCL

Alice的设置：

alice> GRANT SELECT PLAINTEXT(ID, credit\_rank, income, age) ON db\_test.ta TO alice;

alice> GRANT SELECT PLAINTEXT\_AFTER\_JOIN(ID) ON db\_test.ta TO bob;

alice> GRANT SELECT PLAINTEXT\_AFTER\_GROUP\_BY(credit\_rank) ON db\_test.ta TO bob;

alice> GRANT SELECT PLAINTEXT\_AFTER\_AGGREGATE(income) ON db\_test.ta TO bob;

alice> GRANT SELECT PLAINTEXT\_AFTER\_COMPARE(age) ON db\_test.ta TO bob;

bob的设置：

bob> GRANT SELECT PLAINTEXT(ID, order amount, is\_active) ON db\_test.tb TO bob;

bob> GRANT SELECT PLAINTEXT\_AFTER\_JOIN(ID) ON db\_test.tb TO alice,

bob> GRANT SELECT PLAINTEXT\_AFTER\_COMPARE(is\_active) ON db\_test.tb TO alice;

bob> GRANT SELECT PLAINTEXT\_AFTER\_AGGREGATE(order\_amount) ON db\_test.tb TO alice,

第五步：发起联合分析查询：

alice> use db\_test;

alice> SELECT ta.credit\_rank, COUNT(\*) as cnt, AVG(ta.income) as avg\_income,AVG(tb.order\_amount) as avg\_amount FROM ta INNER JOIN tb ON ta.ID = tb.ID WHERE ta.age>=20 AND ta.age<=30 AND tb.is\_active=1 GROUP BY ta.credit\_rank;

# **SCQL工作原理**

工作过程：SCDB把用户的query翻译成执行图，下发给SCQLEngine执行。

SCQLEngine如何执行、和GPU的关系：

## **SCQL和SPU**

SPU的前端把python的机器学习应用程序（pytorch，tensorflow，jax）直接转变为MPC的程序

SCQL使用了SPU的库文件：

SCQL采用SPU架构中的PPHLO C++ API进行接入

在SPU的autosourcing和colocatedIO模式中选择了后者

colocatedIO特性是数据拥有方同时是计算方

autosourcing特性是数据拥有方把数据分片给计算方计算

## **案例：aggregation w/ group by**

SCQL:

SELECT k, sum(v)

FROM virtual\_table

GROUP BY k;

### **明文计算**

排序思路（当然哈希思路也行）

原始数据 ->排序-> 按K排序的数据 ->分组聚合-> 聚合数据

k v k v b sum(v)

a 3 a 3 0 3

b 2 a 10 1 13

c 1 b 2 1 2

a 10 c 1 0 1

c 20 c 20 0 21

c 5 c 5 1 26

其中b[i]为1表示下标i为该分组的最后一个元素

->过滤-> 结果

k sum(v)

a 13

b 2

c 26

### **密态计算**

提供了四种实现方式：

private groupby 某一参与方的本地明文计算，key和聚合类属于同一方。

HESum groupby

Vertical groupby

Oblivious groupby

性能依次降低，但是适用范围越来越大。

假设K在alice，V在bob. 使用半同态，只能获得sum/avg，没法比较大小

1、Alice：给每一个元素赋予一个分组ID，例如a->0, b->1, c->2(hash)

2、BOB把value加密后发给alice

3、Alice把属于同一个分组的密文加起来(aggregate)，发给bob

4、bob解密后获取同态密文的明文

5、完成计算，但是alice不知道每个分组的结果，bob不知道哪些元素在一组。

# **SecretNote上手SCQL**

secretnote + P2P模式

任何一方都可以主动创建项目，邀请其他人加入

## **安装和部署（secretnote方式）**

0、安装：

pip install secretnote -i <https://pypi.tuna.tsinghua.edu.cn/simple>

…/scql$ git clone https://github.com/secretflow/scql.git . (0.8.1b1)

1、在scql/examples/p2p-tutorial执行bash setup.sh

生成alice和bob的公钥、私钥

2、然后执行docker compose -p tk-sn up -d

-p是指定项目名称用的，-d是后台启动

查看容器集群的状态：

docker compose -p tk-sn ps

如果mysql的STATUS为healthy就好了

3、检查端口号：

cat .env

看到如下输出：

ALICE\_PORT=8081

BOB\_PORT=8082

SCQL\_IMAGE=secretflow/scql:latest

4、启动alice

执行secretnote --mode=scql --party=alice --host=http://127.0.0.1:8081

终端看到了这个：<http://localhost:8888/secretnote/scql>

浏览器输入：<http://xx.yy.zz.ww:8888/secretnote/scql/>，看到用户为alice

（

区分一下secretflow：[http:// xx.yy.zz.ww:8888/secretnote/secretflow](http://172.25.172.56:8888/secretnote/secretflow)

这里也有scql，但是没有用户

）

5、启用bob

secretnote --mode=scql --party=bob --host=http://127.0.0.1:8082

输出：<http://127.0.0.1:8889/secretnote/scql>

注意：4、5中Secretnote的启动设定中的host，必须和3给的端口一致，否则项目无法创建

## **使用SCQL(基于brokerctl工具)**

需要完成3.1中的1-3

1. 构建brokerctl:

…/scql$ go build -o brokerctl cmd/brokerctl/main.go

1. 创建项目：

./brokerctl create project --project-id "demo" --host <http://localhost:8081>

1. 查看项目：Members只有alice

./brokerctl get project --host <http://localhost:8081>

1. 添加项目参与方
2. alice邀请bob参与项目

./brokerctl invite bob --project-id "demo" --host http://localhost:8081

1. Bob查看invite：

./brokerctl get invitation --host <http://localhost:8082>

1. Bob加入：

./brokerctl process invitation 1 --response "accept" --project-id "demo" --host <http://localhost:8082>

1. 检查参与方：Members出现alice和bob

./brokerctl get project --host <http://localhost:8081>

1. 创建数据表
2. Alice数据初始化

./brokerctl create table ta --project-id "demo" --columns "ID string, credit\_rank int, income int, age int" --ref-table alice.user\_credit --db-type mysql --host <http://localhost:8081>

1. 查看alice数据

./brokerctl get table ta --host http://localhost:8081 --project-id "demo"

1. 创建bob数据

./brokerctl create table tb --project-id "demo" --columns "ID string, order\_amount double, is\_active int" --ref-table bob.user\_stats --db-type mysql --host http://localhost:8082

1. 检查bob数据

./brokerctl get table tb --host http://localhost:8082 --project-id "demo"

1. CCL授权
2. Alice操作自己的数据表

./brokerctl grant alice PLAINTEXT --project-id "demo" --table-name ta --column-name ID --host http://localhost:8081

./brokerctl grant alice PLAINTEXT --project-id "demo" --table-name ta --column-name credit\_rank --host http://localhost:8081

./brokerctl grant alice PLAINTEXT --project-id "demo" --table-name ta --column-name income --host http://localhost:8081

./brokerctl grant alice PLAINTEXT --project-id "demo" --table-name ta --column-name age --host http://localhost:8081

1. Alice设置bob访问自己表的权限

./brokerctl grant bob PLAINTEXT\_AFTER\_JOIN --project-id "demo" --table-name ta --column-name ID --host http://localhost:8081

./brokerctl grant bob PLAINTEXT\_AFTER\_GROUP\_BY --project-id "demo" --table-name ta --column-name credit\_rank --host http://localhost:8081

./brokerctl grant bob PLAINTEXT\_AFTER\_AGGREGATE --project-id "demo" --table-name ta --column-name income --host http://localhost:8081

./brokerctl grant bob PLAINTEXT\_AFTER\_COMPARE --project-id "demo" --table-name ta --column-name age --host http://localhost:8081

1. Bob操作自己的表

./brokerctl grant bob PLAINTEXT --project-id "demo" --table-name tb --column-name ID --host http://localhost:8082

./brokerctl grant bob PLAINTEXT --project-id "demo" --table-name tb --column-name order\_amount --host http://localhost:8082

./brokerctl grant bob PLAINTEXT --project-id "demo" --table-name tb --column-name is\_active --host http://localhost:8082

1. Bob设置alice访问权限

./brokerctl grant alice PLAINTEXT\_AFTER\_JOIN --project-id "demo" --table-name tb --column-name ID --host http://localhost:8082

./brokerctl grant alice PLAINTEXT\_AFTER\_COMPARE --project-id "demo" --table-name tb --column-name is\_active --host http://localhost:8082

./brokerctl grant alice PLAINTEXT\_AFTER\_AGGREGATE --project-id "demo" --table-name tb --column-name order\_amount --host http://localhost:8082

1. 查看alice能访问的东西（自己表直接访问，另一方的数据操作后可以访问）

./brokerctl get ccl --project-id "demo" --parties alice --host <http://localhost:8081>

1. 查看bob可以访问的东西

./brokerctl get ccl --project-id "demo" --parties bob --host http://localhost:8081

1. 联合SQL查询：

./brokerctl run "SELECT ta.credit\_rank, COUNT(\*) as cnt, AVG(ta.income) as avg\_income, AVG(tb.order\_amount) as avg\_amount FROM ta INNER JOIN tb ON ta.ID = tb.ID WHERE ta.age >= 20 AND ta.age <= 30 AND tb.is\_active=1 GROUP BY ta.credit\_rank;" --project-id "demo" --host http://localhost:8081 --timeout 3

结果如下：

2 rows in set: (0.200944026s)

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

| credit\_rank | cnt | avg\_income | avg\_amount |

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

| 5 | 6 | 18069.72427368164 | 7743.392951965332 |

| 6 | 4 | 336018.1408996582 | 5499.425048828125 |

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

## **使用SCQL(基于secretnote)**

需要完成3.1中的1-5

1. 创建项目：alice网页上点击add new project按钮，新建项目test
2. 邀请bob：点击alice右上角的+发送请求

在bob的主页右上角的铃铛会刷新出提示，点击accept

1. 添加alice数据：（需要在alice的界面添加）

表名称为ta，关联表alice.user\_credit，数据列ID string, credit\_rank int, income int, age int

1. 添加bob数据（需要在bob的界面添加）

表名tb，关联表bob.user\_stats，字段ID string, order\_amount double, is\_active int

1. Alice配置ccl, 给自己的全是plaintext，给bob的如下

ID after join

credit\_rank after group by

income after aggregate

age after compare

1. Bob配置ccl, 给alice的如下：

ID

order\_amount

is\_active

1. Alice端查询，新建一个notebook，执行下面三块代码：

select \* from ta

-- 成功输出

SELECT \* from tb

-- not visiable to alice

SELECT ta.credit\_rank, COUNT(\*) as cnt, AVG(ta.income) as avg\_income, AVG(tb.order\_amount) as avg\_amount FROM ta INNER JOIN tb ON ta.ID = tb.ID WHERE ta.age >= 20 AND ta.age <= 30 AND tb.is\_active=1 GROUP BY ta.credit\_rank;

1. 实验结果如下：

