连通分量算法练习

背景

对银联持卡人关系网络图，依据其连通关系，采用不同算法对其进行划分社区。其中点类型有银行卡、身份证、手机设备、手机号、信用卡、转账等，边类型为共同交易（绑定）信息、转账信息、在商户消费信息等。详细顶点及边如下图，每个顶点均有属性f_id，为唯一标识。

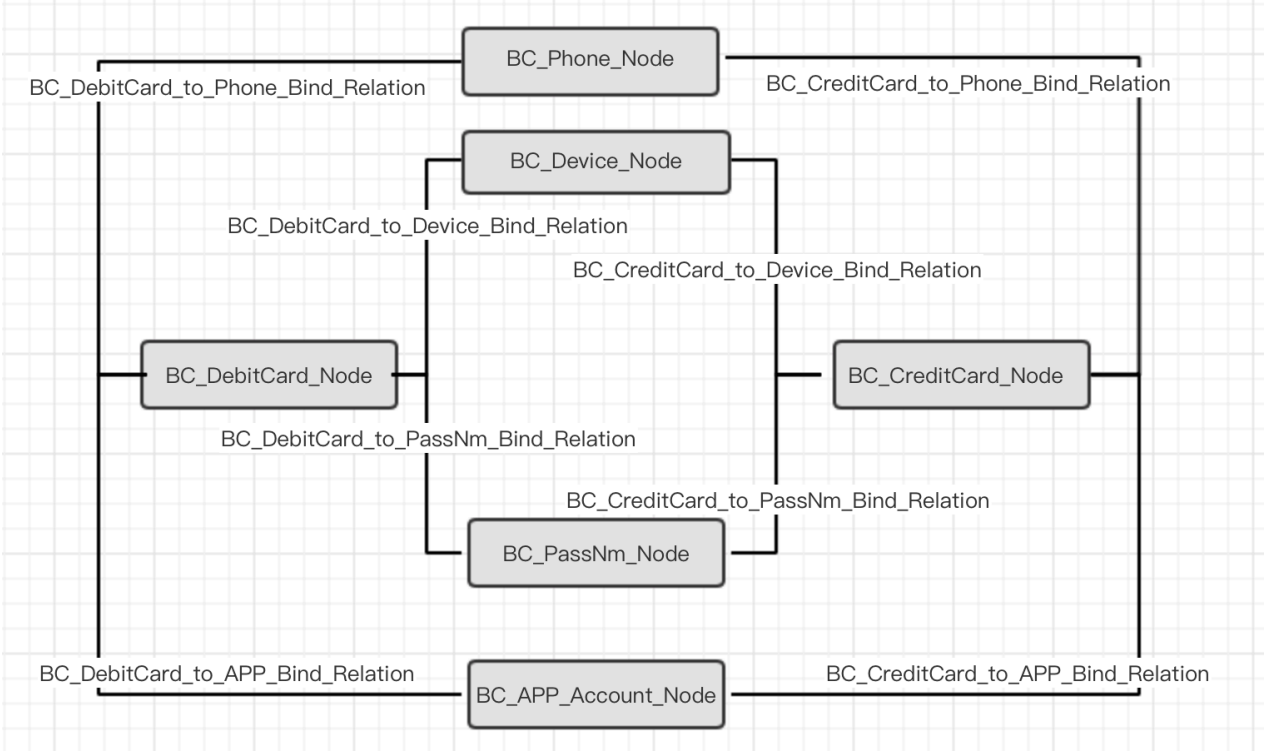

图 unionpay_basic

任务

在图unionpay_basic中，定义一个”社区“为互相连通的一组顶点和边的最大范围。也就是说，在社区内，可以从任何一个顶点（不论其实BC_Phone_Node还是BC_DebitCard_Node，或者其他顶点）到达任何另一个顶点。

请应用连通分量算法，对图unionpay_basic进行社区划分。要求输出每个顶点所属社区。

参考答案

```
CREATE DISTRIBUTED QUERY CONN_COMP_FILE (STRING d_cardfile, STRING phonefile,
STRING devicefile, STRING passfile, STRING appfile, STRING c_cardfile) FOR
GRAPH unionpay_basic {
# This query identifies the Connected Components (undirected edges)

MinAccum<int> @cc_id = 0;
SumAccum<int> @old_id = 0;
OrAccum<bool> @active;
MapAccum<int, int> @@compSizes;
FILE f1(d_cardfile);
FILE f2(phonefile);
FILE f3(devicefile);
FILE f4(passfile);
FILE f5(appfile);
FILE f6(c_cardfile);
Start1 = {BC_DebitCard_Node.*}; #借记卡
Start2 = {BC_Phone_Node.*}; #手机号
Start3 = {BC_Device_Node.*}; #设备
Start4 = {BC_PassNm_Node.*};
Start5 = {BC_APP_Account_Node.*}; #app账号
Start6 = {BC_CreditCard_Node.*}; #信用卡
Start = Start1 UNION Start2 UNION Start3 UNION Start4 UNION Start5 UNION
Start6;

# Initialize: Label each vertex with its own internal ID
LOG(TRUE, "init start");
S = SELECT x FROM Start:x
  POST-ACCUM x.@cc_id = x.f_id,
         x.@old_id = x.f_id;
LOG(TRUE, S.size());
# Propagate smaller internal IDs until no more ID changes can be DOne
WHILE (Start.size()>0) DO
Start = SELECT t FROM Start:s -
((BC_DebitCard_to_Device_Bind_Relation|BC_DebitCard_to_Phone_Bind_Relation|
BC_DebitCard_to_PassNm_Bind_Relation|BC_DebitCard_to_APP_Bind_Relation|
BC_CreditCard_to_Device_Bind_Relation|BC_CreditCard_to_Phone_Bind_Relation|
BC_CreditCard_to_PassNm_Bind_Relation|BC_CreditCard_to_APP_Bind_Relation):e)->
(BC_CreditCard_Node|BC_DebitCard_Node|BC_Phone_Node|BC_Device_Node|BC_PassNm_No
de|BC_APP_Account_Node):t
  ACCUM t.@cc_id += s.@cc_id // If s has a smaller id than t, copy the id to t
  POST-ACCUM
    CASE WHEN
      t.@old_id != t.@cc_id THEN // If t's id has changed
      t.@old_id = t.@cc_id,
      t.@active = true
    ELSE
```

```
      t.@active = false
    END
    HAVING
      t.@active == true;
    END;

LOG(TRUE, "while finished");
#Output
#output_file
Start1 = {BC_DebitCard_Node.*};
L1 = SELECT s FROM Start1:s
POST-ACCUM f1.println(s,s.@cc_id);

Start2 = {BC_Phone_Node.*};
L2 = SELECT s FROM Start2:s
POST-ACCUM f2.println(s,s.@cc_id);

Start3 = {BC_Device_Node.*};
L3 = SELECT s FROM Start3:s
POST-ACCUM f3.println(s,s.@cc_id);

Start4 = {BC_PassNm_Node.*};
L4 = SELECT s FROM Start4:s
POST-ACCUM f4.println(s,s.@cc_id);

Start5 = {BC_APP_Account_Node.*};
L5 = SELECT s FROM Start5:s
POST-ACCUM f5.println(s,s.@cc_id);

Start6 = {BC_CreditCard_Node.*};
L6 = SELECT s FROM Start6:s
POST-ACCUM f6.println(s,s.@cc_id);

LOG(TRUE,"query finished");

}
```