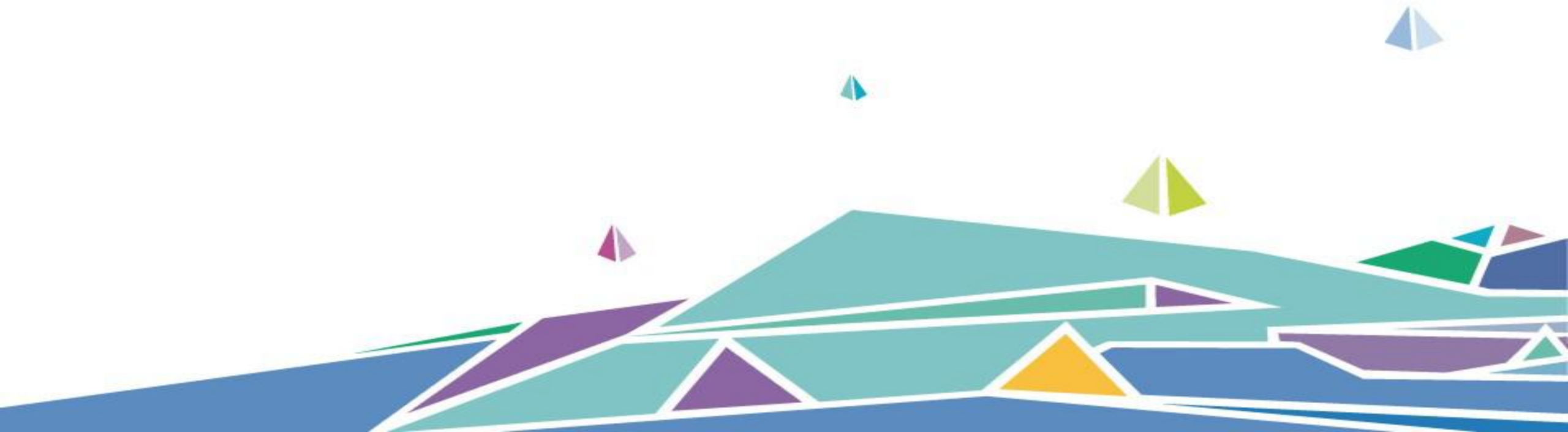
中亦科技
China Etek Service

Perfecting IT service and favoring clients 'success
锻造凝练IT服务 助推用户事业发展

图数据库的基本知识

by中亦大数据团队



目录 Content

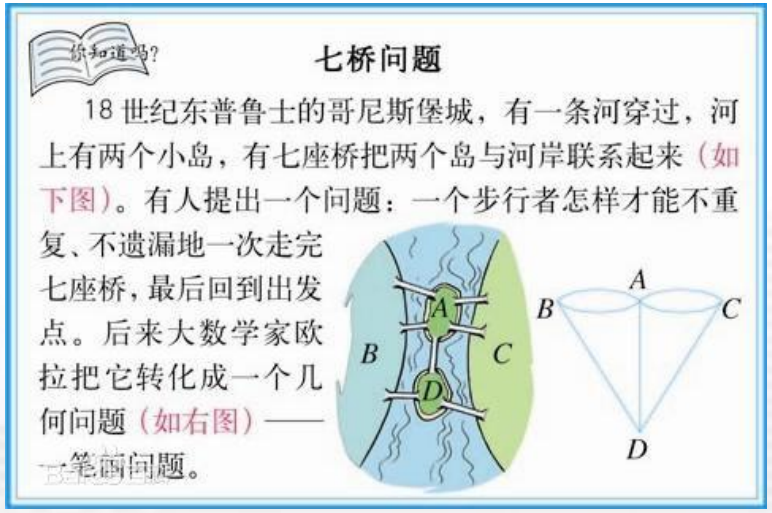
目录

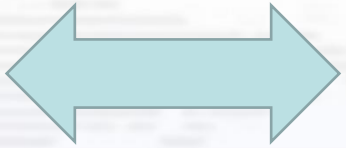
01

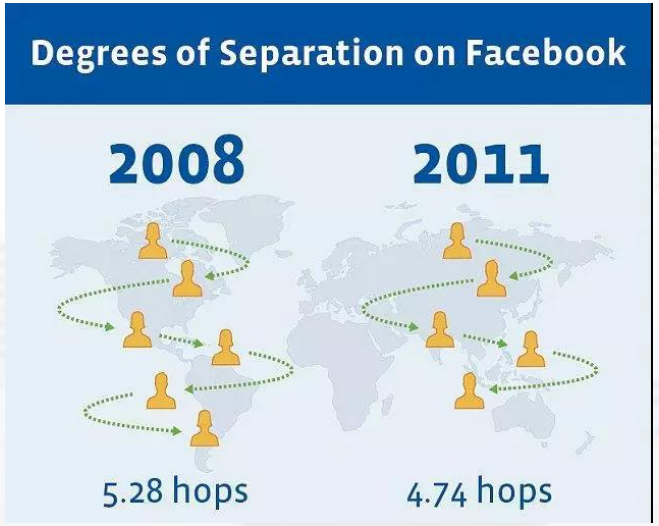
PART 01

图计算基本知识

1.1 图的概念



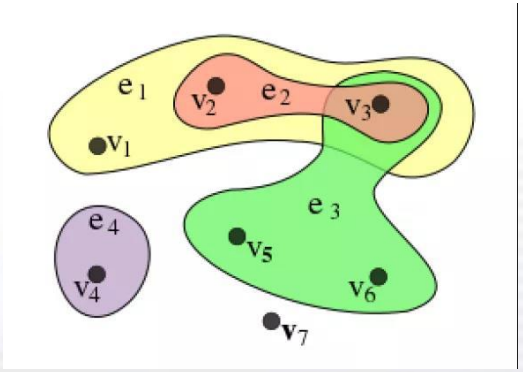


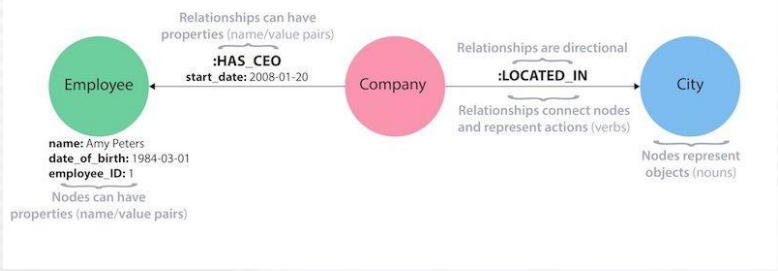


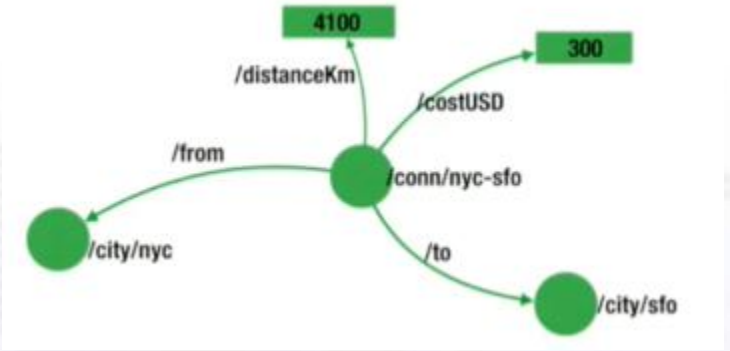
graph(数据结构)　　network（真实网络）

图的定义：由若干个不同顶点与连接其中某些顶点的边所组成的图形就称为图。通常用大写字母G表示图，V表示所有顶点的集合，E表示边的集合，记作G = (V, E)

1.2 图的分类







超图-HyperGraph
在超图中，实体之间是通过超边进行连接的。一条超边对应一类关系。
如图所示，相同颜色区域中的实体之间都是该类边。

属性图-PropertyGraph
每个实体有自己的属性信息，比如公司的职员实体，具有属性：姓名、生日、工号等，职员和公司之间雇佣关系，雇佣关系也有自己的属性：入职日期等。

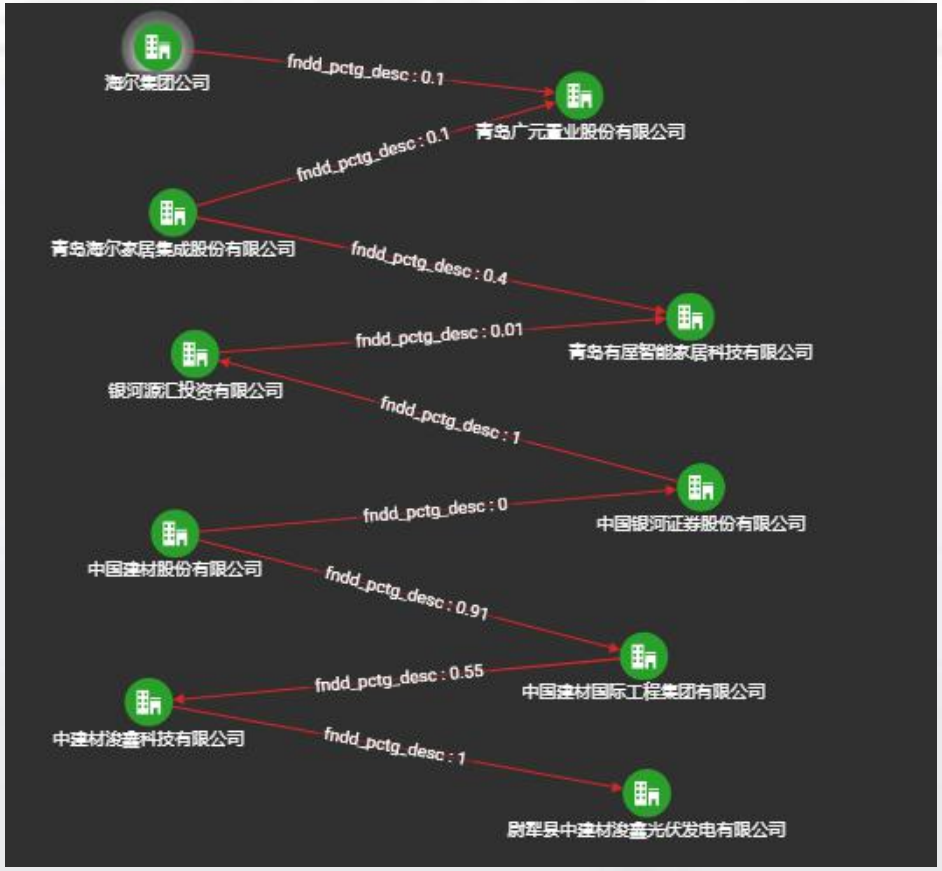
三元组-RDF
三元组是对RDF格式的一种形式化说法。因为RDF的每一条记录有三个部分的内容：资源信息、属性、属性值。因此而得名三元组。

超图、属性图、RDF虽是定义图的拓扑关系的三种不同范式，但是都可以正确表征图的拓扑特征，而且三者之间是可以相互转换的。

1.3 图算法

路径（Path）（找到最短路径或评估某条路径的可行性或质量）

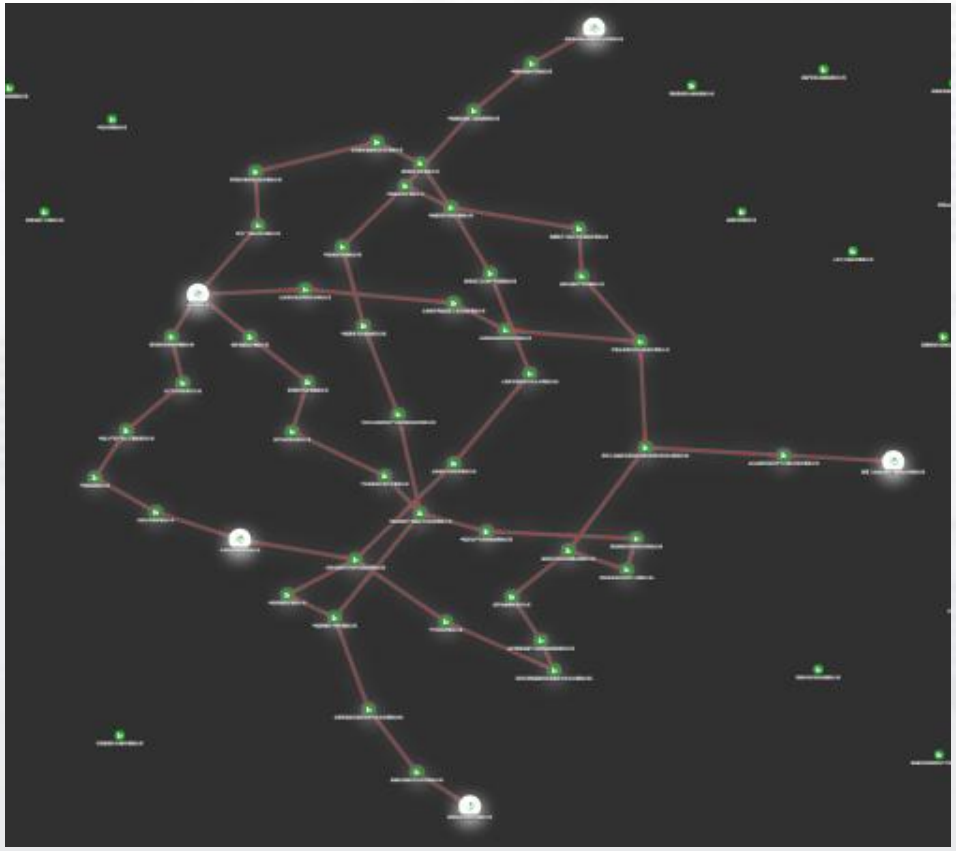
单起点最短路径算法（Single-Source Shortest Path），无权重
单起点最短路径算法（Single-Source Shortest Path），含权重
单顶点对最短路径算法（Single-Pair Shortest Path）
全顶点对最短路径算法（All-Pairs Shortest Path）

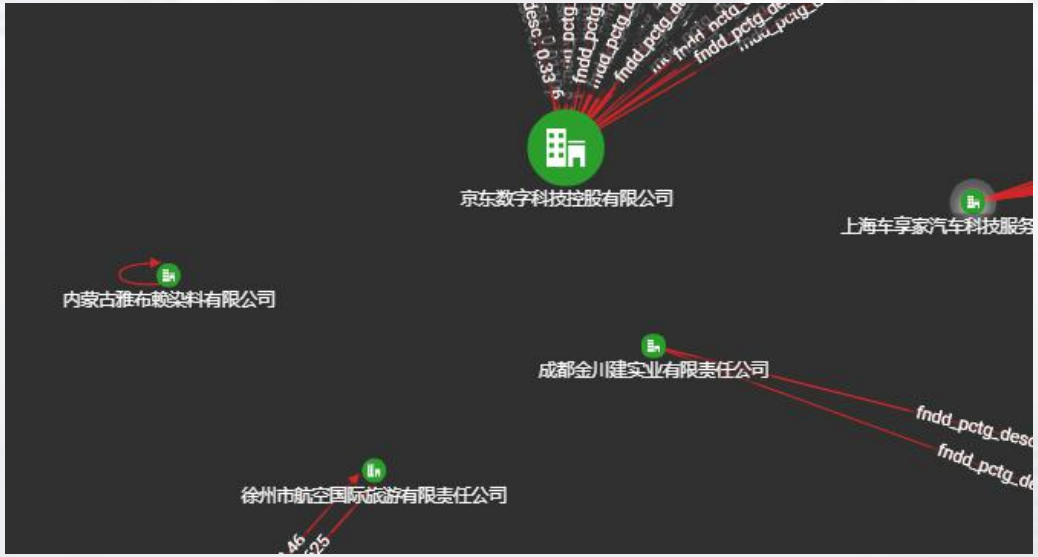


1.3 图算法

中心度（Centrality）（为了确定网络中某个顶点对于总体的重要性）

页面排名算法（PageRank）
接近中心度算法（Closeness Centrality）
中介中心度算法（Betweenness Centrality）





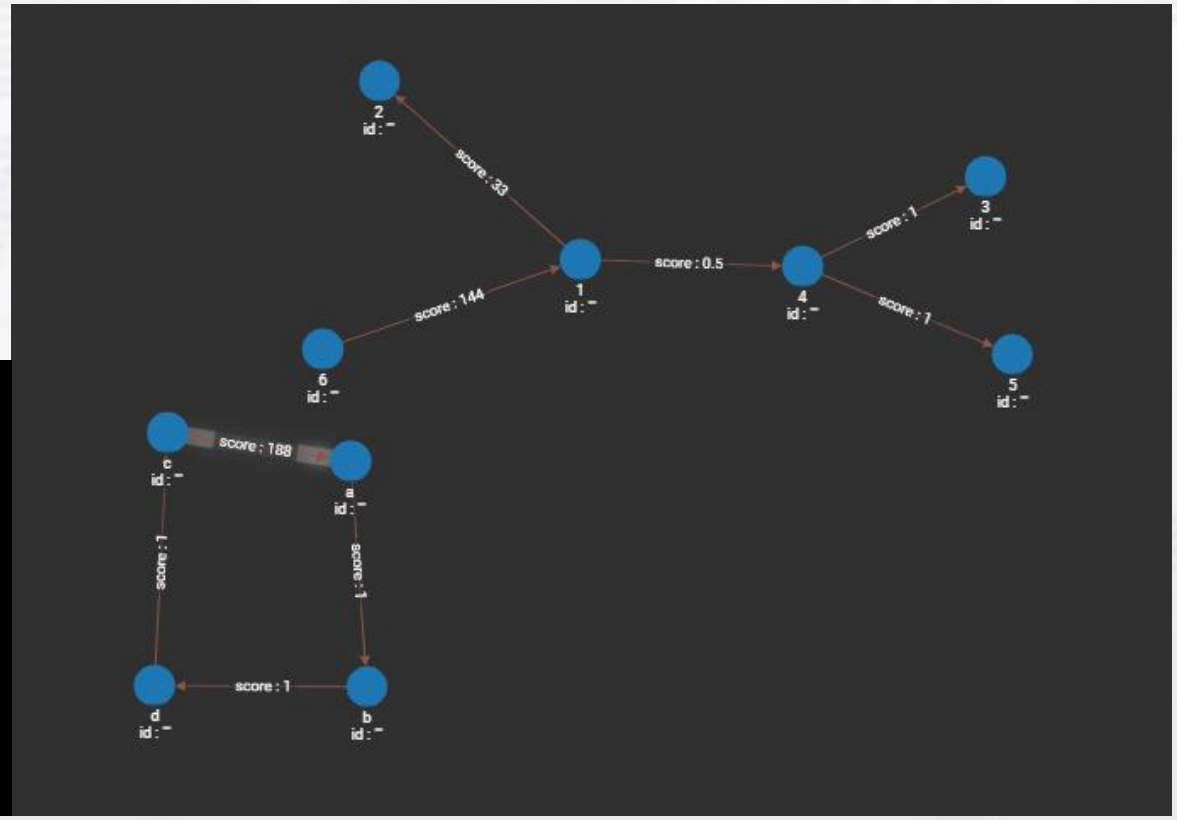
1.3 图算法

群体度（Community）

（评估一个网络结构中个体组合或分裂的程度，同时也能够得到网络的组织程度正在加强或削弱的趋势）

连通分量算法（Connected Components）
标签传播算法（Label Propagation）
鲁汶算法（Louvain Modularity）
三角计数算法（Triangle Counting）

```
Vertex_ID,community_id
c,391118848
a,391118848

5,517996544
3,517996544
4,517996544

6,571473920
2,571473920

b,560988160
d,560988160
1,571473920
```



1.3 图算法

衡量相似度（Similarity）

邻点的余弦相似度算法（Cosine Similarity），单起点
邻点的余弦相似度算法（Cosine Similarity），All Pairs
邻点的杰卡德相似度算法（Jaccard Similarity），单起点
邻点的杰卡德相似度算法（Jaccard Similarity），All Pairs

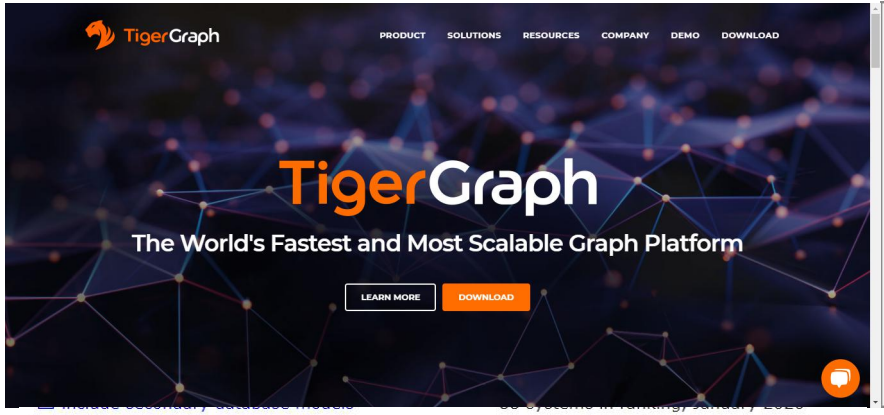
目录

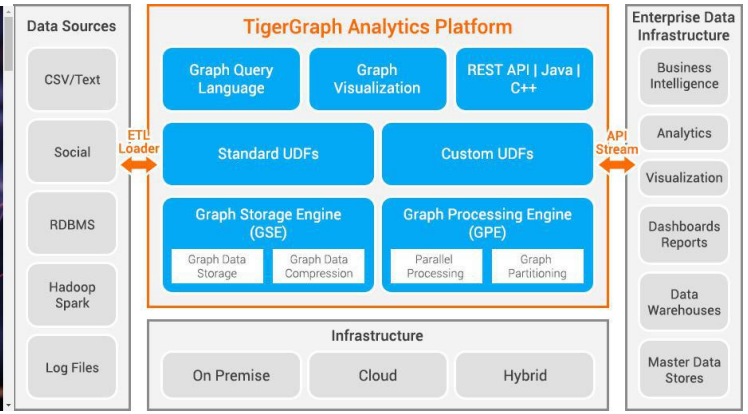
02

PART 02

图数据库基本使用

2.1 tigergraph简介



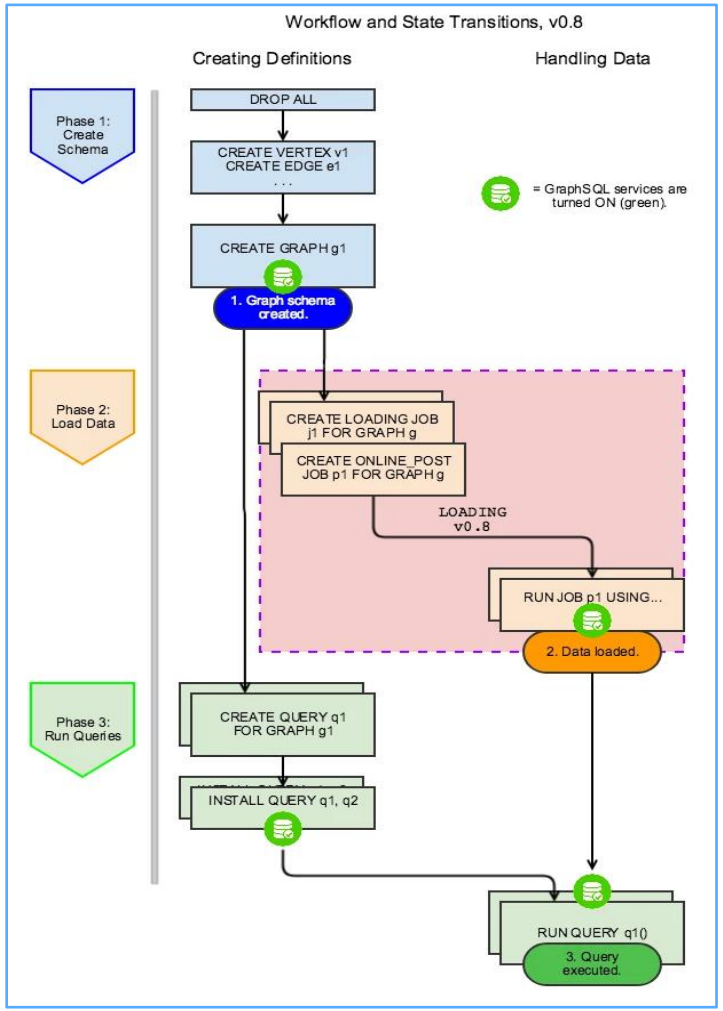


Rank Jan 2020	Rank Dec 2019	Rank Jan 2019	DBMS	Database Model	Score Jan 2020	Score Dec 2019	Score Jan 2019
1.	1.	1.	Neo4j	Graph	51.66	+1.10	+4.86
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	31.51	+0.07	+7.12
3.	↑ 4.	↑ 4.	ArangoDB	Multi-model	5.21	+0.33	+0.92
4.	↓ 3.	↓ 3.	OrientDB	Multi-model	5.11	+0.18	-0.63
5.	5.	5.	Virtuoso	Multi-model	2.65	+0.01	+0.00
6.	6.	6.	JanusGraph	Graph	1.77	+0.02	+0.51
7.	7.	7.	Amazon Neptune	Multi-model	1.73	+0.16	+0.67
8.	8.	↑ 10.	GraphDB	Multi-model	1.14	-0.01	+0.33
9.	↑ 11.	↑ 11.	Dgraph	Graph	1.04	+0.09	+0.39
10.	↓ 9.	↓ 8.	Giraph	Graph	1.02	-0.02	+0.02
11.	↓ 10.	↑ 13.	TigerGraph	Graph	1.00	+0.04	+0.41
12.	12.	↓ 9.	AllegroGraph	Multi-model	0.86	-0.02	-0.05
13.	13.	↓ 12.	Stardog	Multi-model	0.81	+0.03	+0.18
14.	14.	↑ 18.	FaunaDB	Multi-model	0.80	+0.11	+0.50
15.	15.	15.	Blazegraph	Multi-model	0.64	+0.02	+0.17
16.	16.	↓ 14.	Graph Engine	Multi-model	0.58	0.00	+0.01

深度关联分析
多维度数据表示
高级聚合及分析
增强的机器学习和人工智能

2.2 tigergraph工作流程

1. 定义/创建一个 Graph Schema --业务访谈、场景梳理
2. Load 少量数据 --数据分析、数据映射
3. 创建查询 --规则实现、代码调试
4. 运行查询 --样本验证、性能调优
5. 导入更多数据 --迭代优化、生产上线



2.3 图数据模型vs关系型数据模型

1. 在关系模型中，操作的对象和结果都是一张二维表；

2. 关系型数据库可用于表示实体之间的多对多的关系，只是此时要借助第三个关系---表，来实现多对多的关系；

3. 关系必须是规范化的关系，即每个属性是不可分割的实体，不允许表中表的存在；

学生

stu_id	stu_name	sex	age

课程

cour_id	cour_name	xuefen

教师

tea_id	tea_name	sex	age

选课

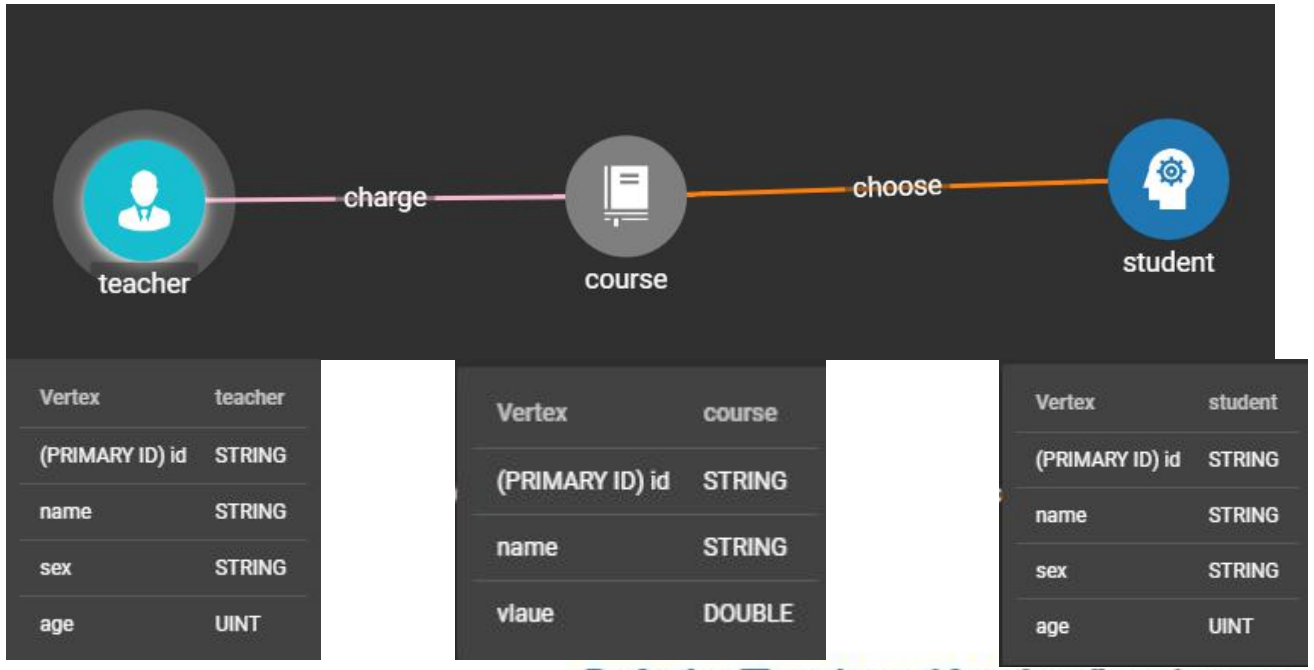
stu_id	cour_id	chengji

教课

tea_id	cour_id

强调关系/关系 - 非常适合社交/交易/网络分析

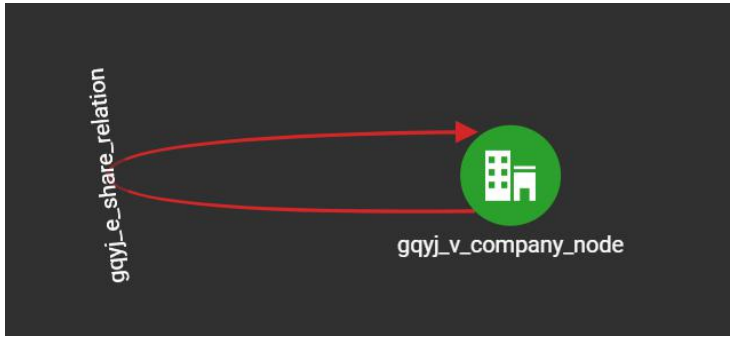
1. **点（vertex）** - 现实世界或抽象实体的模型
2. **边 （edge）** - 两个顶点之间的关系
3. **图（Schema ）** - 定义完整“世界”的相关顶点和边缘的集合



2.4 SCHEMA DDL

```
CREATE VERTEX gqyj_v_company_node(PRIMARY_ID id STRING,
                cust_name STRING,
                unn_soc_cr_cd STRING,
                 entp_inf_idr STRING,
                 status STRING,
                 data_version DATETIME,
                 level INT,
                 mu_c STRING)
```

Vertex	gqyj_v_company_node
(PRIMARY ID) id	STRING
cust_name	STRING
unn_soc_cr_cd	STRING
entp_inf_idr	STRING
status	STRING
data_version	DATETIME
level	INT
mu_c	STRING




```
CREATE  DIRECTED EDGE gqyj_e_share_relation(
                FROM gqyj_v_company_node, TO gqyj_v_company_node,
                fndd_pctg_desc DOUBLE,
                data_version DATETIME,
                mu_set SET<STRING>) WITH REVERSE_EDGE="gqyj_e_share_relation_by"
```

Edge	gqyj_e_share_relation
reverse edge	gqyj_e_share_relation_by
fndd_pctg_desc	DOUBLE
data_version	DATETIME
level	INT
mu_c	STRING
mu_set	SET<STRING>

```
CREATE  GRAPH  gqyj(gqyj_v_company_node , gqyj_e_share_relation)
```

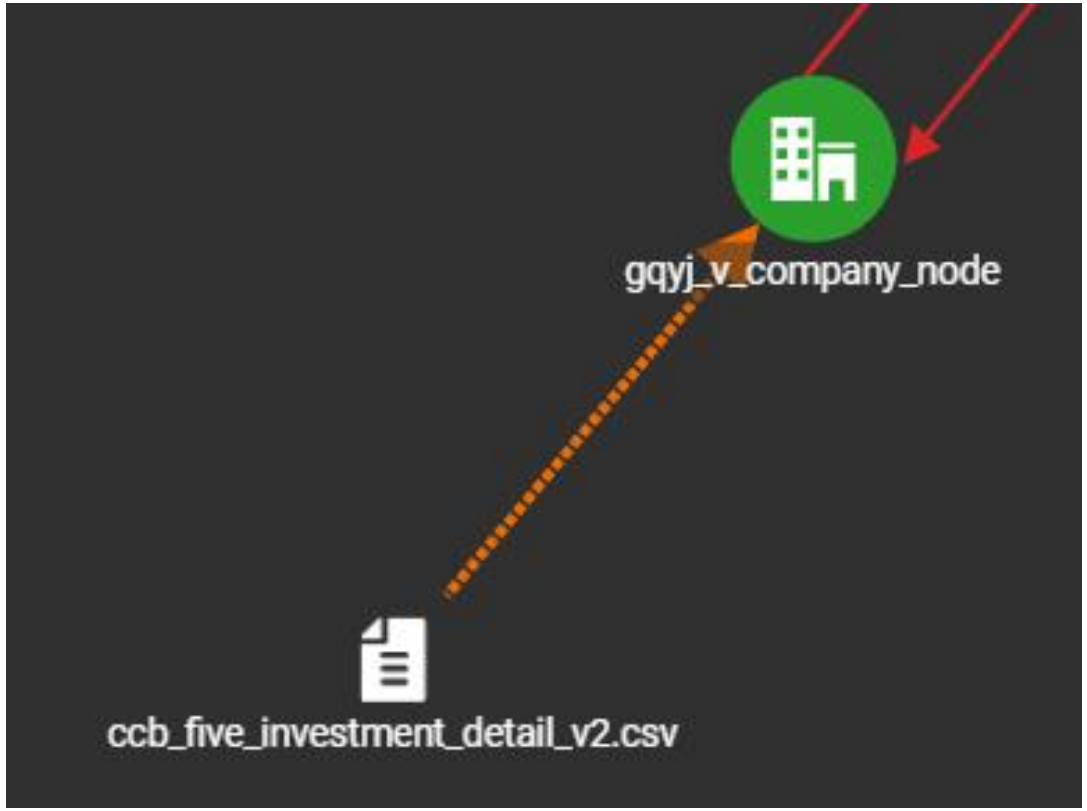
2.4 SCHEMA DDL

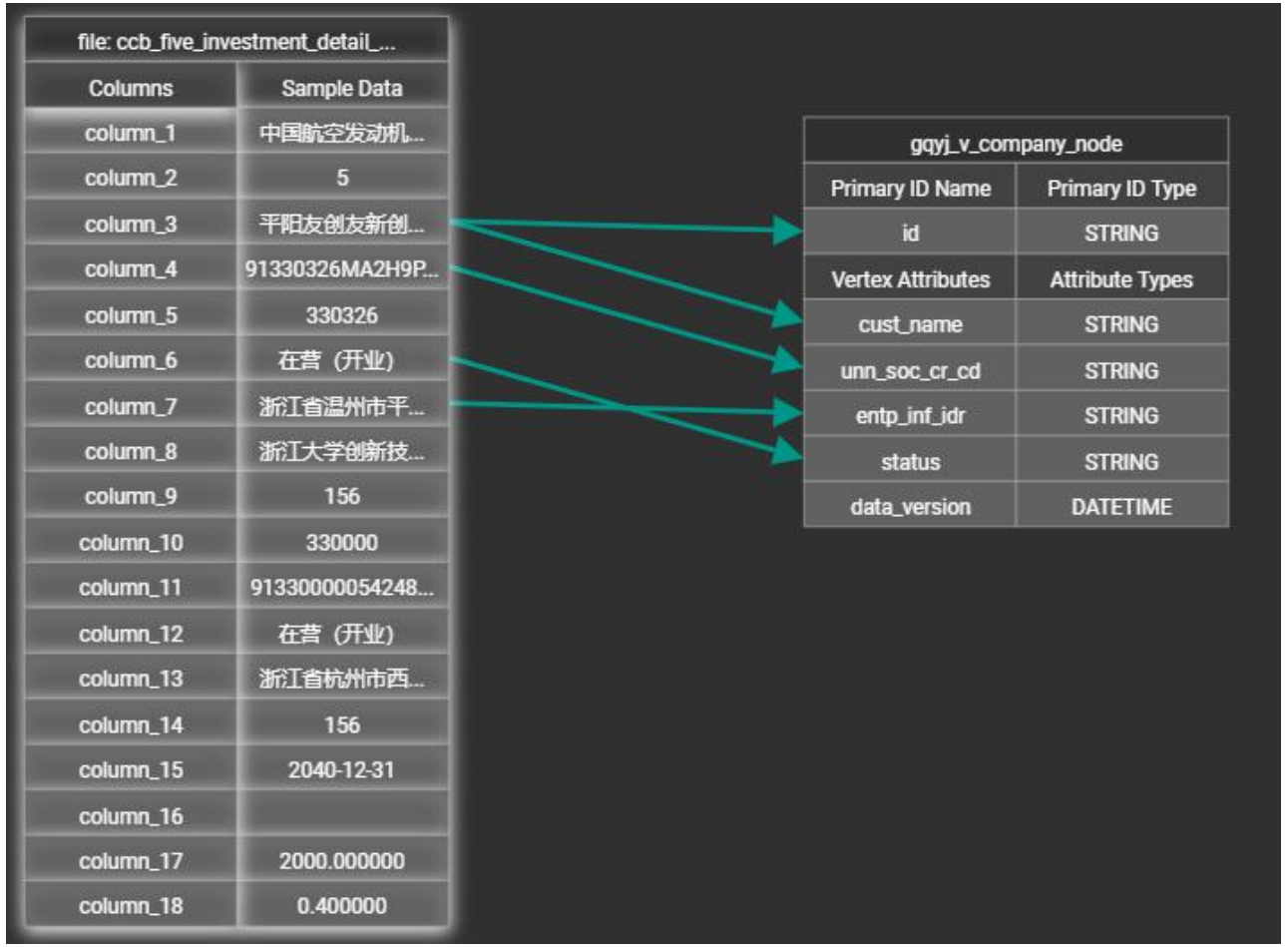
数据类型	属性类型
Numeric	INT, UINT, FLOAT, DOUBLE
Boolean	BOOL
Text	STRING, STRING COMPRESS
Time	DATETIME
Collection	SET<t>, LIST<t>, MAP<k,v>

1、PRIMARY_ID 可以是 UINT 或者 STRING，但是要唯一
2、属性有标准默认值，也可以定制默认值

2.5 导入数据（loading|etl）

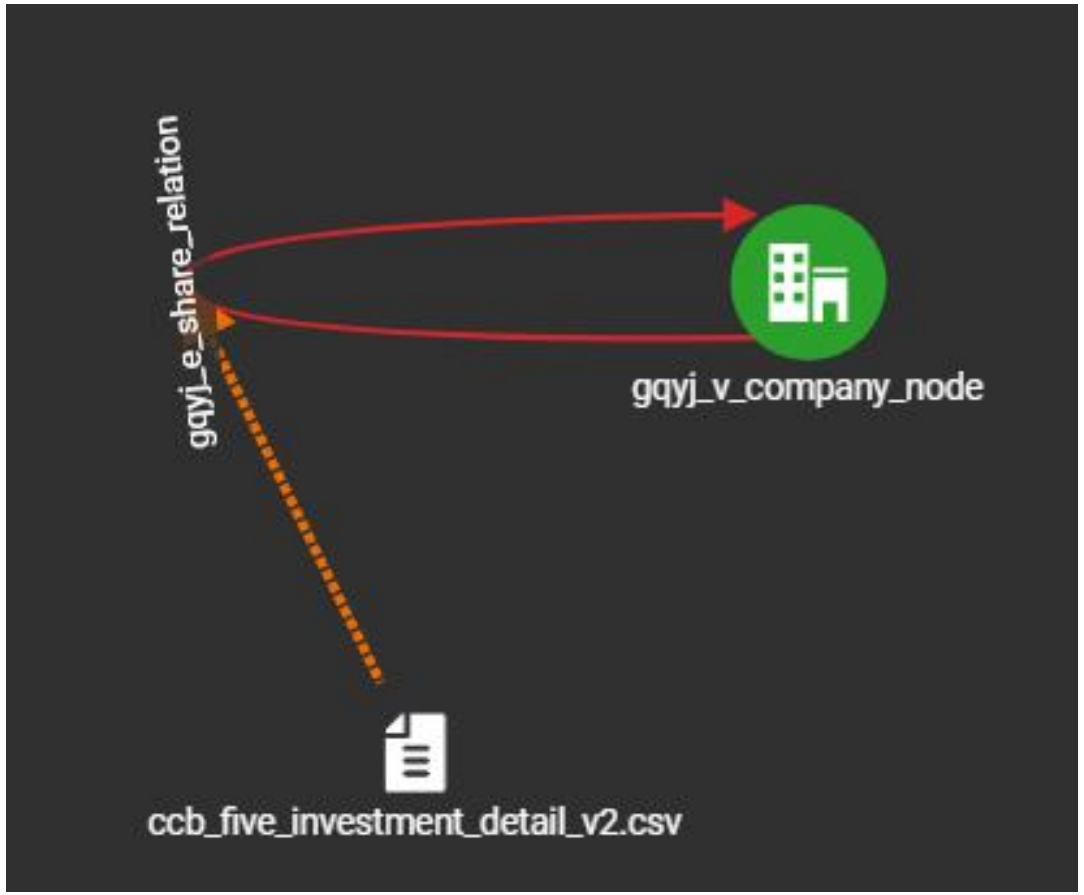
- 定义数据映射

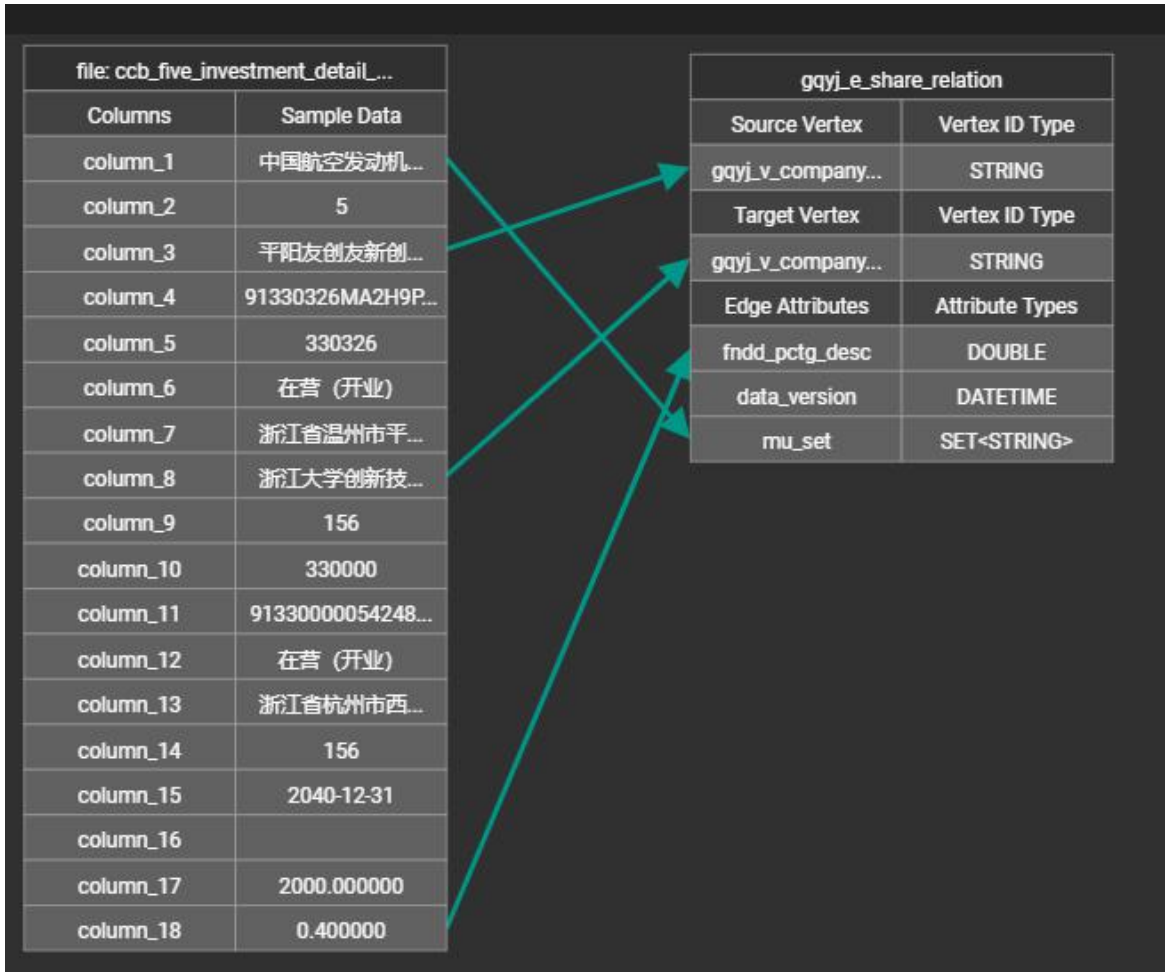




2.5 导入数据（loading|etl）

- 定义数据映射





2.5 导入数据（loading|etl）

- 定义数据映射
- 定义loading逻辑

```
CREATE LOADING JOB load_job_ccb FOR GRAPH gqyj {
     DEFINE FILENAME MyDataSource;
     LOAD MyDataSource TO EDGE gqyj_e_share_relation VALUES
      ($7, $2, _, _, _, _, $0)
      USING SEPARATOR=",", HEADER="false", EOL="\n";
   }
```

- 开启loading任务

```
RUN LOADING JOB jobName using f=””
```

2.6 编写查询（DML）

```
GSQL > select * from gqyj_v_company_node limit 1
[{
  "v_id": "海尔集团公司",
  "attributes": {
    "unn_soc_cr_cd": "91370200163562681G",
    "data_version": "1970-01-01 00:00:00",
    "level": 0,
    "cust_name": "海尔集团公司",
    "entp_inf_idr": "",
    "status": "在营（开业）",
    "mu_c": ""
  },
  "v_type": "gqyj_v_company_node"
}]
```

```
GSQL > delete from gqyj_v_company_node where primary_id == "calvin"
{
  "error": false,
  "message": "",
  "version": {
    "schema": 97,
    "edition": "enterprise",
    "api": "v2"
  },
  "results": {
    "deleted_vertices": 1,
    "v_type": "gqyj_v_company_node"
  }
}
GSQL >
```

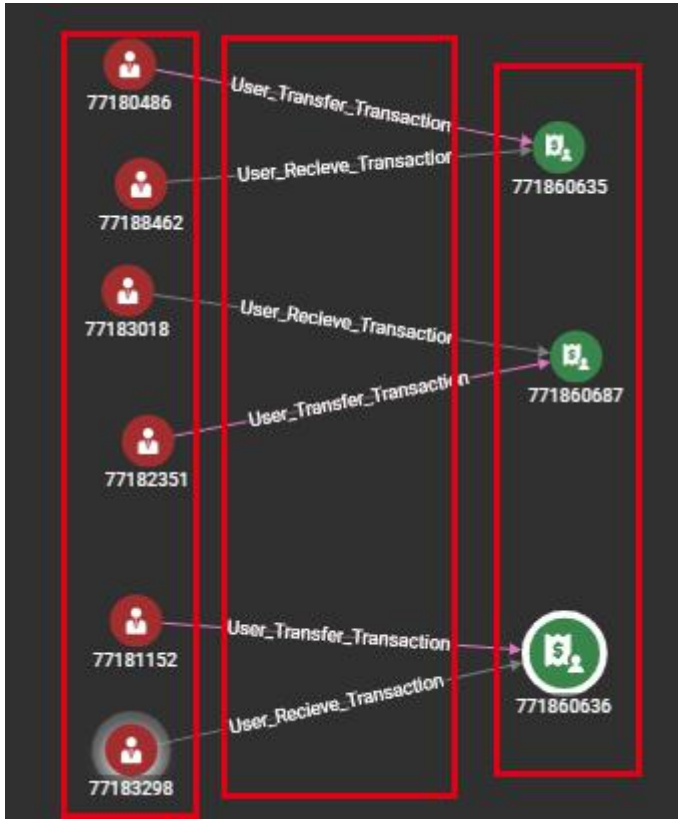
GraphStudio Admin

```
1  CREATE QUERY calvin_betweenness(vertex target_vertex,int step,bool debug) FOR GRAPH BCM_07 {
2    /*
3    从目标公司 通过担保关系扩展 n 度出去;
4    在形成子图中 计算出 任意顶点对 无加权最短路径 y 条;
5    计算有经过 目标公司的最短路径有 x 条;
6    中介度 = x/y
7
8    */
9    OrAccum @visited;
10
11   MapAccum<vertex,MinAccum<int>>@short_info;
12   MapAccum<vertex,MapAccum<vertex,MinAccum<int>>>@recive_short_info;
13
14   MapAccum <vertex,ListAccum<ListAccum<vertex>>> @short_path;
15   MapAccum<vertex,MapAccum <vertex,ListAccum<ListAccum<vertex>>>>@recv_short_path;
16   SumAccum<int> @@all_short_path,@@vis_short_path;
17
18   #MapAccum<vertex,MapAccum <vertex,ListAccum<ListAccum<vertex>>>> @all_short_path;
19   int c_step;
20
21
22   # 构建子图
23   c_step = 2* step;
24   start = {target_vertex};
25
```

2.6 编写查询（DML）

select 语法范式

```
resultSet = SELECT vSet
            FROM startSet:s  -(edgeTypes:e)-> targetTypes:t
             [sampleClause]                          ----随机采样
            [whereClause]                            ----条件过滤
            [accumClause]                            ----沿边并行计算
            [postAccumClause]                        ----基于点 并行计算
            [havingClause]                           ----条件过滤
            [orderClause]                            ----排序
            [limitClause]                            ----限制输出
    ;
```



2.6 编写查询（DML）

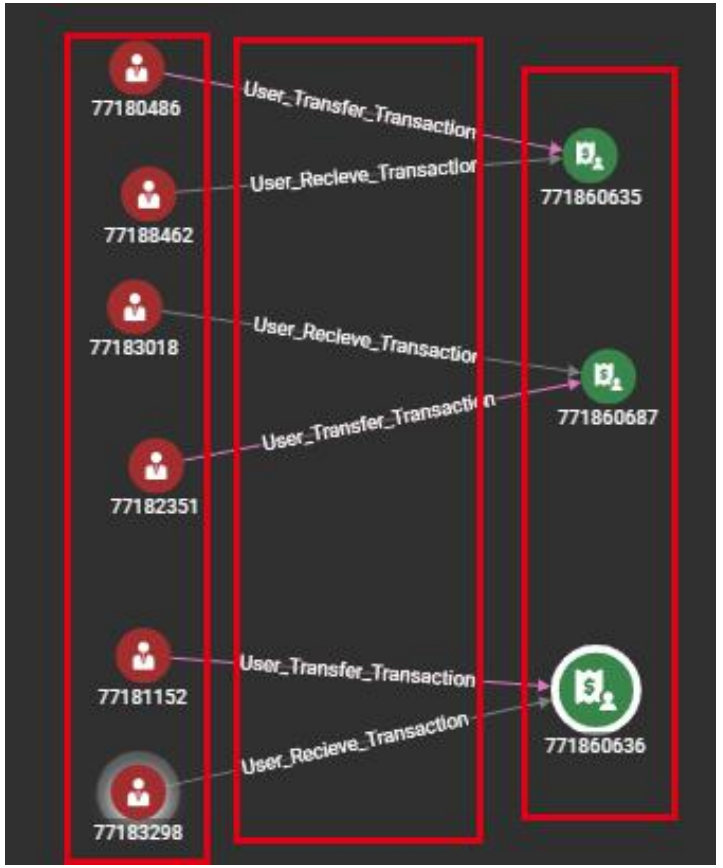
from 目标结果

可以从顶点集或者边集选择

```
FROM startSet:s
```

可以从定义的模板选择

```
FROM startSet:s –(edgeTypes:e)-> targetTypes:t
```

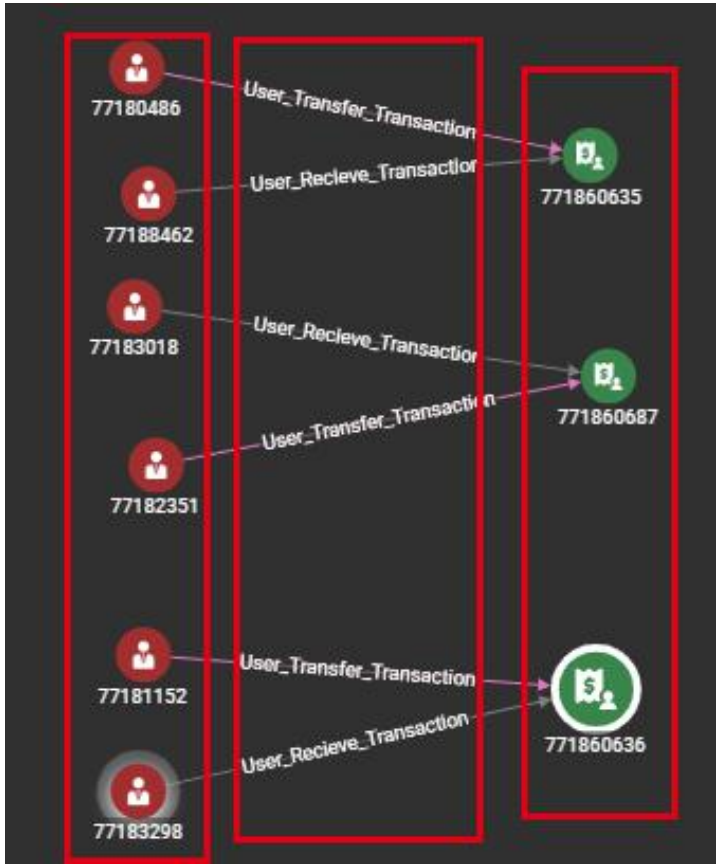


2.6 编写查询（DML）

seed （起点）

```
SO = seed;
S1 = SELECT t FROM S0:s -(:e)-> :t …;
S2 = SELECT t FROM S1:s -(:e)-> :t …;
```

```
# vertex or vertex set parameter
CREATE QUERY seedEx(VERTEX<person> seed, SET<VERTEX<person> seedSet)
FOR GRAPH socialNet {
    S1 = {seed};    # curly braces package the vertex into a set
    S2 = seedSet;
    S3 = ANY;       # all vertices, for global computations like
PageRank
    S4 = person.*;  # all person vertices
```



2.6 编写查询（DML）

Accumulators（累加器）种类

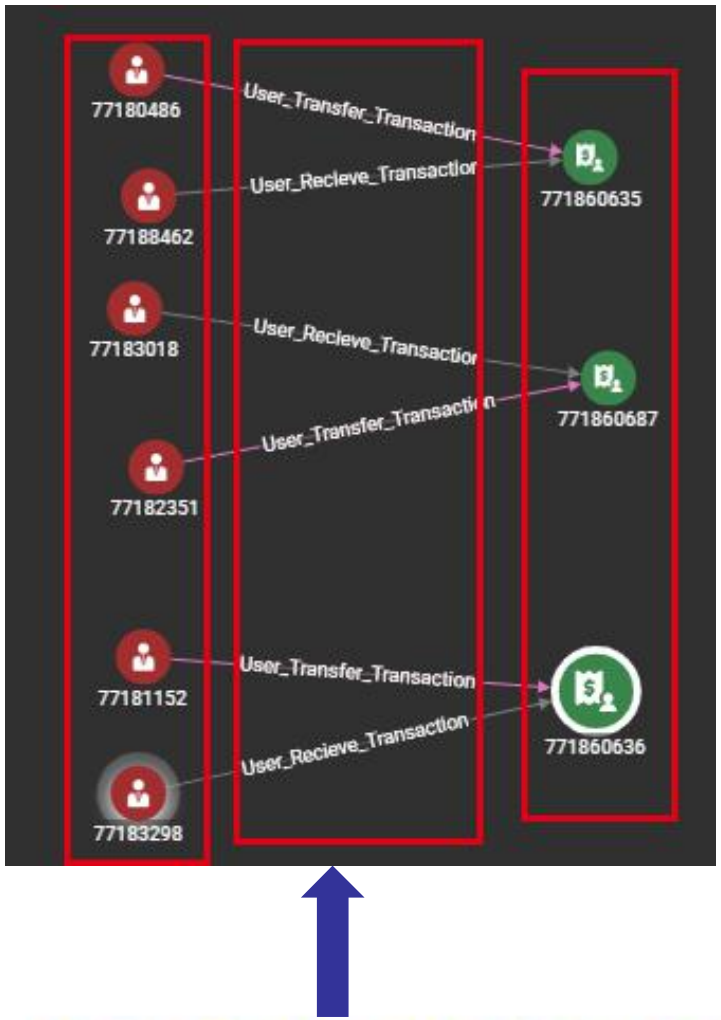
```
MaxAccum  <INT>    @@ globalMax;
SetAccum  <STRING> @  tags;
```

- Scalar Accumulators（标量累加器）
  - SumAccum （保留求和值）
  - MinAccum / MaxAccum / AvgAccum （保留最小，最大，平均值）
  - AndAccum / OrAccum （保留bool值）
  - BitwiseAndAccum / BitwiseOrAccum
- Container Accumulators（容器累加器）
  - ListAccum ------> []
  - SetAccum ------> ()
  - BagAccum ------> 无序容器
  - MapAccum ------> {}
  - ArrayAccum -----> 多维数组
  - HeapAccum -----> 堆排序
  - GroupByAccum -----> 复合累加器

2.6 编写查询（DML）

ACCUM （面向边计算）

```
AvgAccum @@avgIncome;
MinAccum<DOUBLE> @@minIncome;
MaxAccum<DOUBLE> @@maxIncome;

Result = SELECT cust
     FROM Customer:cust-(salary_by:e)->Company:comp
     WHERE cust.age BETWEEN 18 AND 50
     ACCUM @@avgIncome += e.income,
           @@minIncome += e.income,
           @@maxIncome += e.income;
```

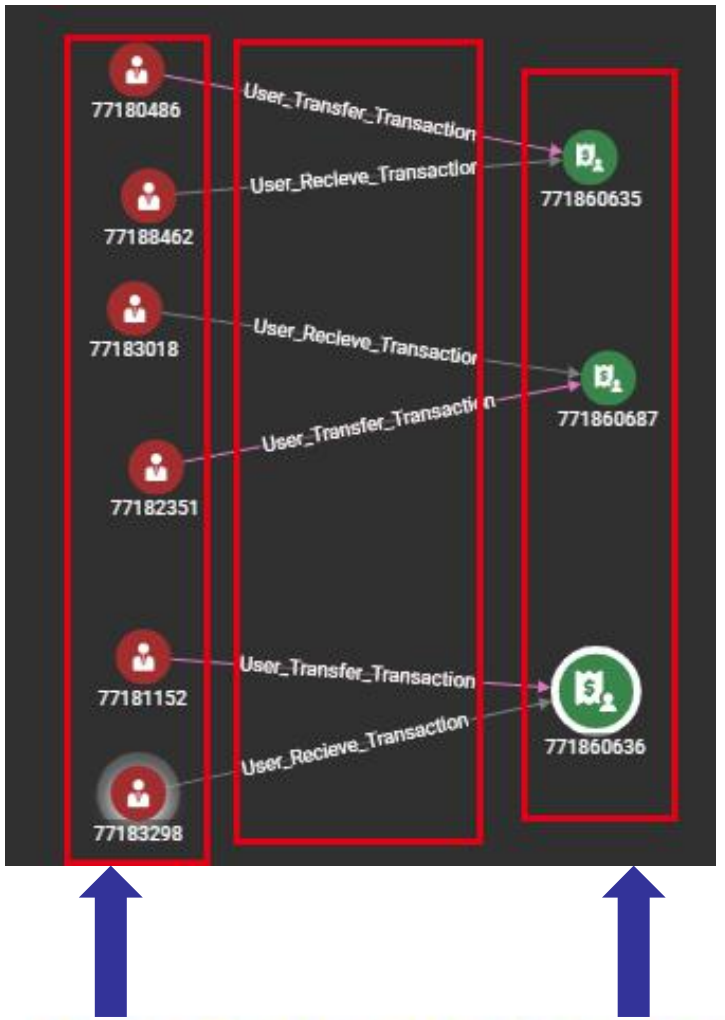


2.6 编写查询（DML）

postAccum（面向点集计算）

postAccum
1、面向点集
2、在accum之后计算

```
users= {inputUser};
invited_users = SELECT t
        FROM users:s-(User_Refer_User:e)-:t
        WHERE t != inputUser
        ACCUM @@visRes += e
        POST-ACCUM @@invitedPersonNum += 1;
```



2.6 编写查询（DML）

流程控制语句--判断

```
IF condition THEN statements
  [ELSE IF condition THEN statements]*
  [ELSE statements]
END
```

```
people = SELECT v FROM startingVertex -(friend:e)->:v
                ACCUM CASE v.gender
                  WHEN "Male" THEN @@males += 1
                  WHEN "Female" THEN @@females +=1
                  ELSE @@unknown += 1
                END;
```

2.6 编写查询（DML）

流程控制语句--循环

```
WHILE condition [LIMIT (name|integer)] DO
       statements
END
-------------------------------------------------
FOREACH item IN @@companyList DO
  IF item in selectedCountry THEN
       count = count + 1;
  END;
END;
-------------------------------------------------
FOREACH item IN RANGE[a,b].STEP(c) DO
  @@total += @@masterList[item];
END;
```

2.6 编写查询（DML）

集合操作

VSet1 = SELECT …

[1,2,3]

VSet2 = SELECT …

[3,4,5]

Vset3 = VSet1 UNION VSet2;

[1,2,3,4,5]

VSet4 = Vset1 INTERSECT VSet2;

[3]

VSet5 = VSet1 MINUS VSet2;

[1,2]

2.5 编写查询

查询的嵌套调用

```
CREATE QUERY sub (VERTEX v) FOR GRAPH MyGraph RETURNS (SumAccum<INT>) {
    SumAccum<INT> @@result, @cnt = 1;
    Start = {v};
    Start = select t from Start-(:e)-:t;
    Start = select t from Start-(:e)-:t where t!=v accum @@result += t.@cnt;
    return @@result;
}
CREATE QUERY main () FOR GRAPH MyGraph{
    SumAccum<INT> @cnt;
    Start = {*.*};
    Start = select s from Start:s post-accum s.@cnt = sub(s);
    print Start;
}
```

2.6 案例（PageRank）

```
CREATE QUERY pageRank (FLOAT maxChange, INT maxIter, FLOAT damping)
FOR GRAPH gsql_demo
{
  MaxAccum<FLOAT> @@maxDiff = 9999;
  SumAccum<FLOAT> @recvScore = 0;
  SumAccum<FLOAT> @score = 1;
  Vset = {Page.*};
  WHILE @@maxDiff > maxChange LIMIT maxIter DO
    @@maxDiff = 0;
    Vset = SELECT s
         FROM Vset:s-(Linkto)->:t
         ACCUM t.@recvScore  += s.@score/s.outdegree()
         POST-ACCUM t.@score = damping + (1-damping) * t.@recvScore,
                    t.@recvScore  = 0,
                    @@maxDiff +=   abs(t.@score - t.@score');
    PRINT @@maxDiff;
  END;
  PRINT Vset.page_id, Vset.@score;
}
```

目录

03

PART 03

图应用场景分享

3.1 案例-移动电影推荐

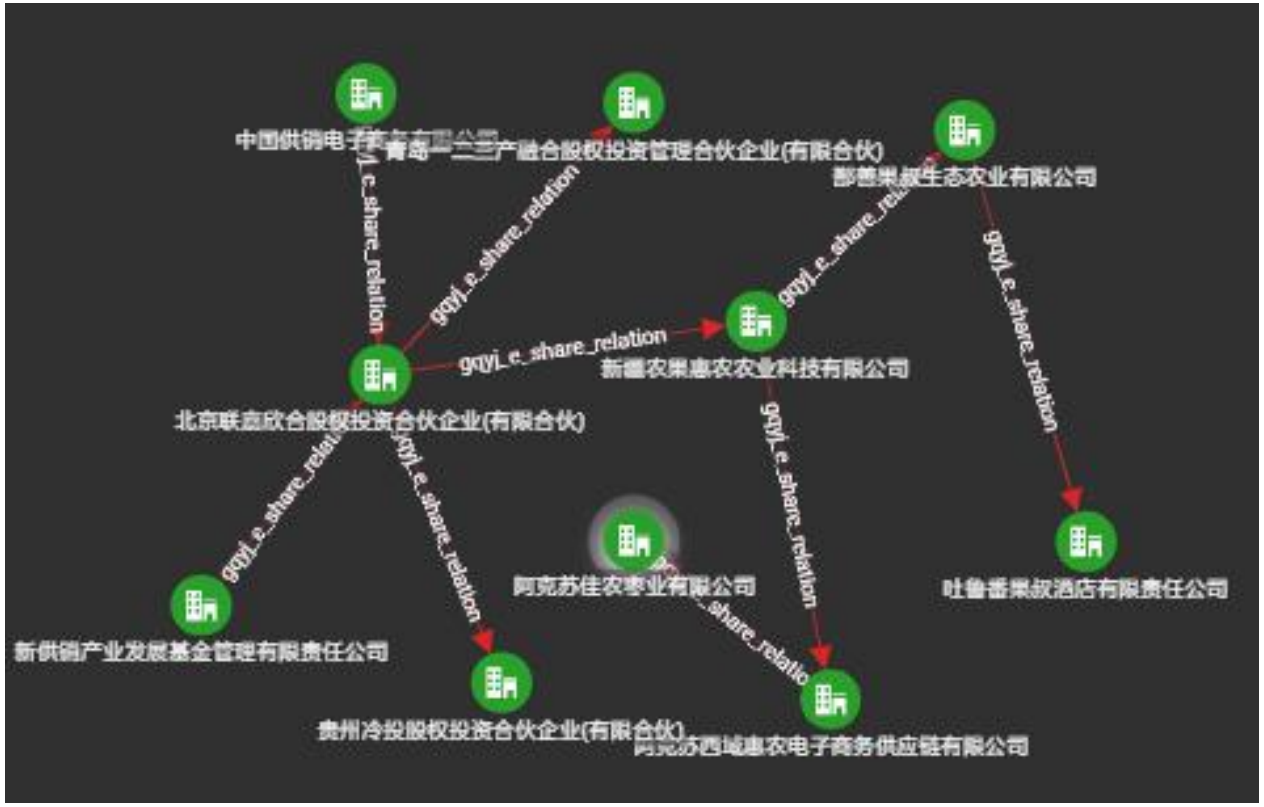
$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$


```
CREATE QUERY recommendMovie (VERTEX<person> p1, INT k1, INT k2) FOR GRAPH recommend_movie
{
    TYPEDEF tuple<STRING title, DOUBLE score> moviescore;
    OrAccum @rated;
    SumAccum<double> @ratingByp1;
    SumAccum<double> @lengthA, @lengthB, @dotAB, @cosineSimi;
    AvgAccum @recommendScore;
    HeapAccum <moviescore>(k2, score DESC) @@onlyMovieName;

    Start = {p1};
    # S1: 找出p1这个人看过的电影
    p1RatedMovie = SELECT m
                FROM Start:s - (ratings:r) -> movies:m
                ACCUM m.@rated = true, m.@ratingByp1 = r.rating;

    #S2：找出看过p1_movie的其他人
    peopleRatedSameMovie =
        SELECT p
        FROM p1RatedMovie:m - (ratings:r) -> person:p
        WHERE p != p1
        ACCUM p.@lengthA += m.@ratingByp1 * m.@ratingByp1,
                p.@lengthB += r.rating * r.rating,
                p.@dotAB += m.@ratingByp1 * r.rating
        POST-ACCUM p.@cosineSimi = p.@dotAB / sqrt(p.@lengthB) / sqrt(p.@lengthB)
        ORDER BY p.@cosineSimi DESC
        LIMIT k1;

```

3.1 案例-移动电影推荐

```
27
28      #S3: 找出p2这些人看过的电影的平均分前k2
29      result =
30          SELECT m
31          FROM peopleRatedSameMovie:p - (ratings:r) -> movies:m
32          WHERE m.@rated == false
33          ACCUM m.@recommendScore += r.rating,
34                @@onlyMovieName += moviescore(m.title, r.rating)
35          ORDER BY m.@recommendScore DESC
36          LIMIT k2;
37
38      PRINT result.title, result.@recommendScore;
39      PRINT @@onlyMovieName;
40
41  }
```

3.2 图应用场景分享--股权预警业务需求



通过公司集团控股数据，分析被目标集团母公司为其单一大股东的公司

3.2 图应用场景分享--设计图结构

```
CREATE VERTEX gqyj_v_company_node(PRIMARY_ID id STRING,
                cust_name STRING,
                unn_soc_cr_cd STRING,
                 entp_inf_idr STRING,
                 status STRING,
                 data_version DATETIME,
                 level INT,
                 mu_c STRING)
```

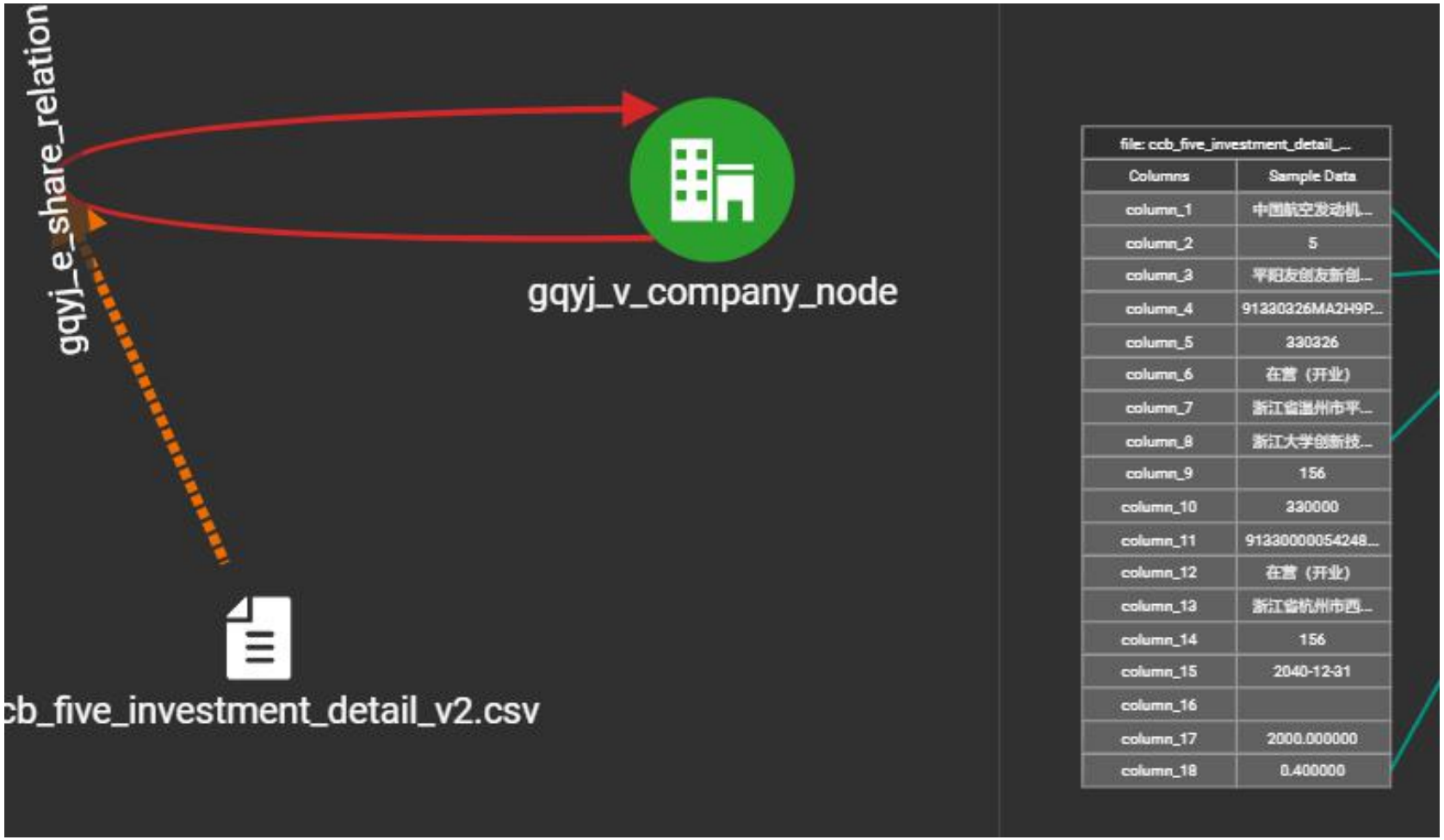
Vertex	gqyj_v_company_node
(PRIMARY ID) id	STRING
cust_name	STRING
unn_soc_cr_cd	STRING
entp_inf_idr	STRING
status	STRING
data_version	DATETIME
level	INT
mu_c	STRING

```
CREATE  DIRECTED EDGE gqyj_e_share_relation(
                FROM gqyj_v_company_node, TO gqyj_v_company_node,
                fndd_pctg_desc DOUBLE,
                data_version DATETIME,
                mu_set SET<STRING>) WITH REVERSE_EDGE="gqyj_e_share_relation_by"
```

Edge	gqyj_e_share_relation
reverse edge	gqyj_e_share_relation_by
fndd_pctg_desc	DOUBLE
data_version	DATETIME
level	INT
mu_c	STRING
mu_set	SET<STRING>

```
CREATE  GRAPH  gqyj(gqyj_v_company_node , gqyj_e_share_relation)
```

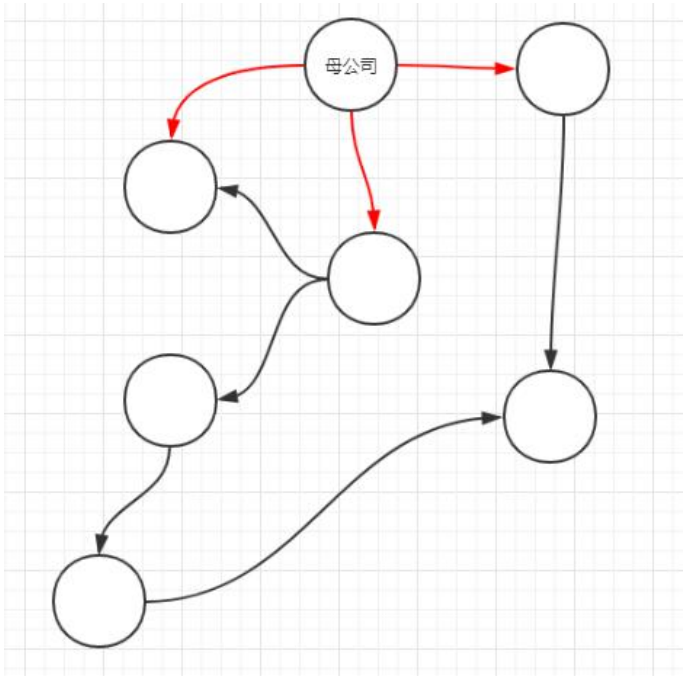
3.3 图应用场景分享--数据清洗&导入

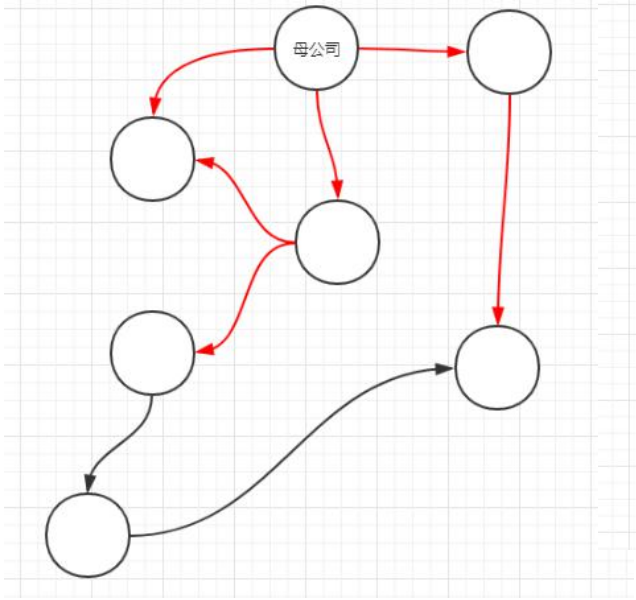


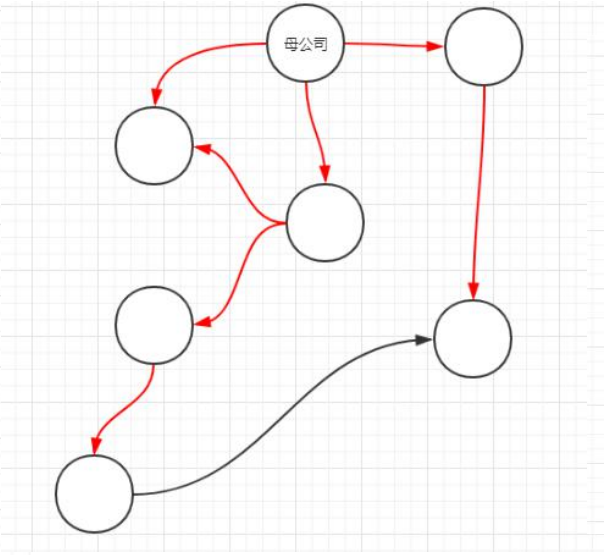
file: ccb_five_investment_detail_...	
Columns	Sample Data
column_1	中国航空发动机...
column_2	5
column_3	平阳友创友新创...
column_4	91330326MA2H9P...
column_5	330326
column_6	在营（开业）
column_7	浙江省温州市平...
column_8	浙江大学创新技...
column_9	156
column_10	330000
column_11	91330000054248...
column_12	在营（开业）
column_13	浙江省杭州市西...
column_14	156
column_15	2040-12-31
column_16	
column_17	2000.000000
column_18	0.400000

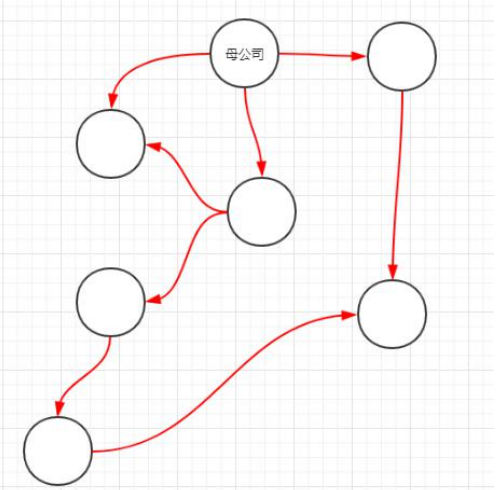
3.4 图应用场景分享--编写查询一

单一大股东规则一：
从母公司出发，沿着51%以上控股路径 n度关联到的公司









3.4 图应用场景分享--编写查询一

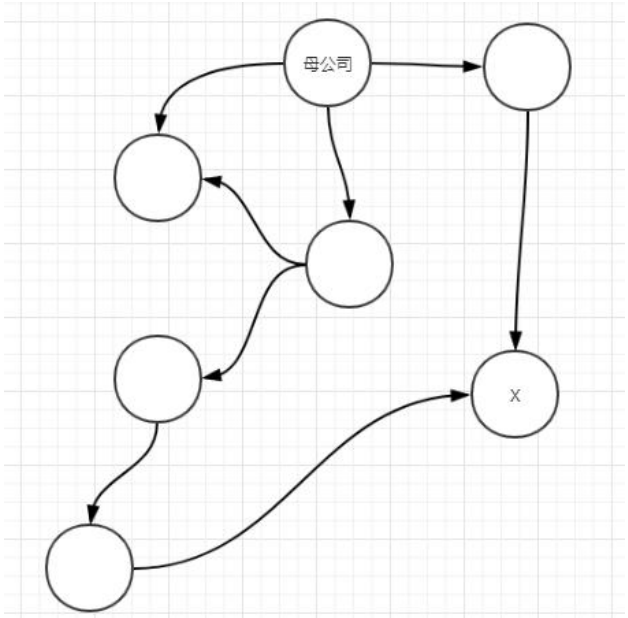
单一大股东规则一：
从母公司出发，沿着51%以上控股路径 n度关联到的公司

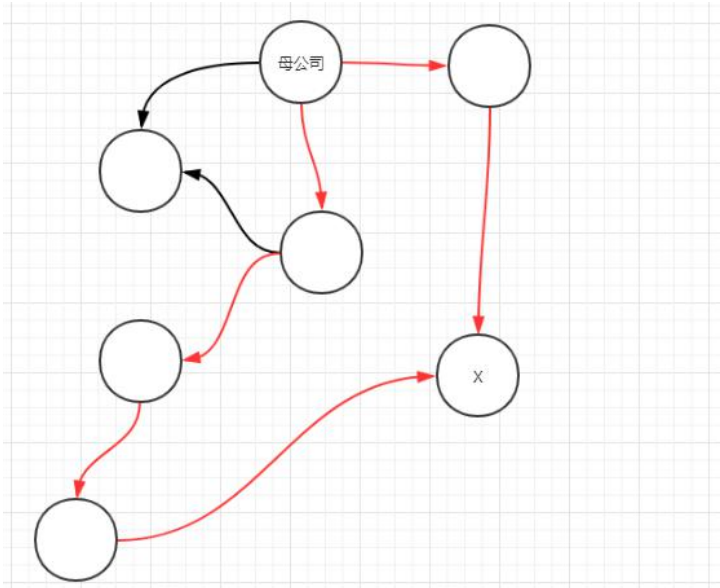
```
# 进行规则判断

start = {target_v};
# role1
# OrAccum @is_control_rule1
FOREACH sub_step in RANGE[1,step] do
   L1 = select t from start:s-(gqyj_e_share_relation:e)->:t
         where e.fndd_pctg_desc> (0.51-epsilon)
          and e.fndd_pctg_desc< (1 + epsilon)
          and e.data_version == datatime
          and t.data_version == datatime
          and t.@is_control_rule1 != TRUE
          and trim(t.status) == "在营（开业）"
        post-accum  t.@is_control_rule1 = TRUE;
  if L1.size()==0 then
       break;
   end;
   start = L1;
end;
log(true,"完成规则1判断");
```

3.5 图应用场景分享--编写查询二

单一大股东规则二：
从母公司出发，沿着控股路径 n度关联到的公司中，不被规则一命中的公司，间接控股比例超过51%

1、所有间接控股路径之和

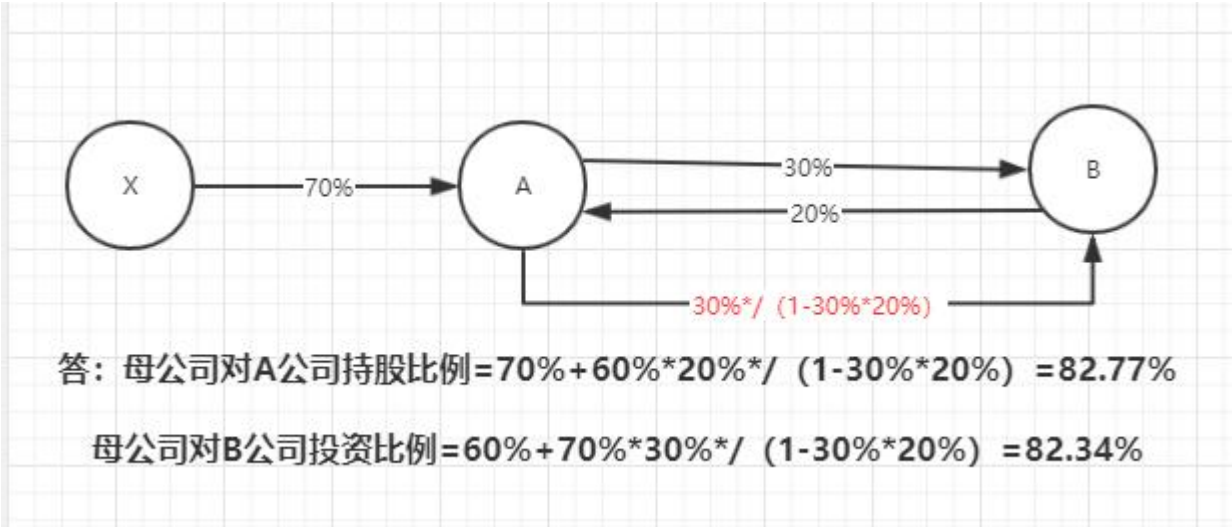




3.5 图应用场景分享--编写查询二

单一大股东规则二：
从母公司出发，沿着控股路径 n度关联到的公司中，不被规则一命中的公司，间接控股比例超过51%

2、交叉控股：
节点序列无重复节点
交叉控股关系比例替换



3.5 图应用场景分享--编写查询二（找出子图与交叉控股）

```
# 找出子图
FOREACH sub_step in RANGE[1,step] do
 start = select t from start:s-(gqyj_e_share_relation:e)->:t
      where trim(t.status) == "在营（开业）"
        and t.data_version == datatime
      and e.data_version == datatime
      and e.mu_set.contains (mu_co)
      post-accum t.@visited = true;
  all_v = all_v union start;
  print sub_step,all_v.size();
end;

# 找出交叉持股节点与股份

start = select s from all_v:s-((gqyj_e_share_relation):e)->:t
         where trim(t.status) == "在营（开业）"
        and t.data_version == datatime
        and e.data_version == datatime
        and e.mu_set.contains (mu_co)
             and t.@visited == true
             AND  t!=s
             and t in s.neighbors("gqyj_e_share_relation_by")
         accum s.@circle_share += (t.cust_name->e.fndd_pctg_desc);
```

3.5 图应用场景分享--编写查询二（找出所有控股路径与比例）

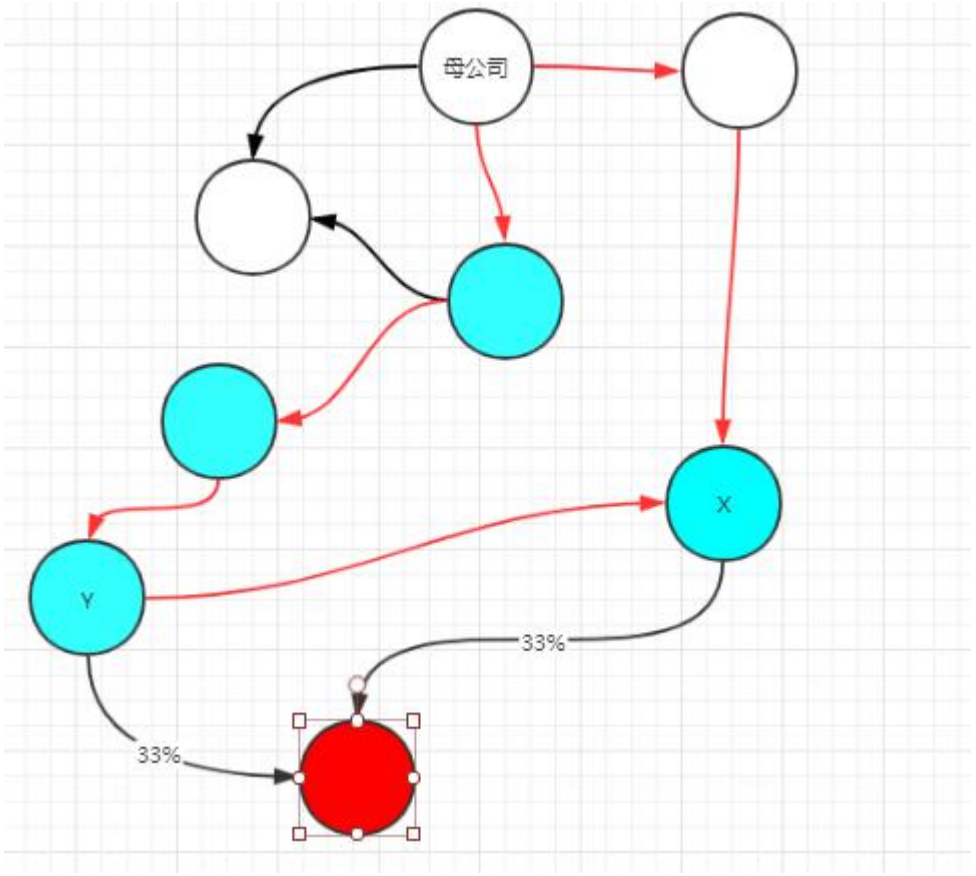
```
TYPEDEF TUPLE<vertex up_company, double share_percent,string unn> item;
HeapAccum<item>(1,share_percent desc) @item_tuple;
MapAccum<string,double> @send_share,@recive_share,@finished_share,@circle_share;
OrAccum @visited,@is_control_rule1,@is_control_rule2,@is_control_rule3;
SetAccum<vertex> @@is_controled_company_rule1,@@is_controled_company_rule2,@@is_controled_company_rule3;
SetAccum<vertex> @rule3_v;
SumAccum<double> @rule2_sum,@rule3_v_sum;
SumAccum<int> @@is_controledd_company_num,@@update_vertex;
MaxAccum<int> @max_level;
start = {target_v};
# 计算对每个点的全部路径
#初始化

start = select t from start:t POST-ACCUM t.@send_share +=(t.cust_name->1);
all_v0 = all_v;
@@update_vertex = 999;
while @@update_vertex>0 do
  @@update_vertex=0;
   all_v0 = select t from all_v0:s-(gqyj_e_share_relation:e)->:t
           where trim(t.status) == "在营（开业）" and t.data_version == datatime and e.data_version == datatime
            and t.@visited == true
            and e.fndd_pctg_desc> (0 + epsilon)
            and e.fndd_pctg_desc< (1 + epsilon)
          and e.mu_set.contains (mu_co)
           accum
              foreach (k,v) in s.@send_share do
                  string tg = "->"+t.cust_name+"->",
                  if not contains_string(k,tg) then
                          string  tg =k+"->"+to_string(e.fndd_pctg_desc)+"->"+t.cust_name,
                      if not t.@finished_share.containsKey(tg) then
                          if not t.@circle_share.containsKey(s.cust_name) then
                            t.@finished_share +=(tg-> v*e.fndd_pctg_desc),
                            t.@recive_share += (tg-> v*e.fndd_pctg_desc)
                          else
                            t.@finished_share +=(tg-> v*e.fndd_pctg_desc/(1-e.fndd_pctg_desc*t.@circle_share.get(s.cust_name))),
                            t.@recive_share += (tg-> v*e.fndd_pctg_desc/(1-e.fndd_pctg_desc*t.@circle_share.get(s.cust_name)))
                          end
                      END
                  end
              end
           POST-ACCUM
              @@update_vertex += t.@recive_share.size() ,
              t.@send_share = t.@recive_share,
              t.@recive_share.clear();
end;
```

3.5 图应用场景分享--编写查询二（计算间接控股并判断）

```
# role2

L2 = select t from all_v:t
         where t!= target_v
               and  trim(t.status) == "在营（开业）"    and t.data_version == datatime
          accum
                  foreach (k,v) in t.@finished_share do
                                t.@rule2_sum += v
                  end
          POST-ACCUM
                 if t.@rule2_sum >(0.51-epsilon) and t.@is_control_rule1 != TRUE then
                     t.@is_control_rule2 = true,
                     @@is_controled_company_rule2 += t
                 end;

print L2.size();
```

3.6 图应用场景分享--编写查询三

单一大股东规则二：
从母公司出发，沿着控股路径 n度关联到的公司中，不被规则一命中的公司，间接控股比例超过51%



3.6 图应用场景分享--编写查询三

单一大股东规则二：
从母公司出发，沿着控股路径 n度关联到的公司中，不被规则一命中的公司，间接控股比例超过51%

```
# rule3
L3 = select s from all_v:s-(gqyj_e_share_relation_by:e)->:t

      where s.@is_control_rule1 != true
      and s.@is_control_rule2 != true
      and t.@is_control_rule1 == true
      and  trim(t.status) == "在营（开业）" and t.data_version == datatime and e.data_version == datatime
and e.mu_set.contains (mu_co)
        accum s.@rule3_v_sum += e.fndd_pctg_desc
         HAVING s.@rule3_v_sum >(0.51-epsilon) ;

L3 = select t from L3:t post-accum t.@is_control_rule3 =true;
```

感谢您的观看

THANK YOU

北京中亦安图科技股份有限公司

ChinaEtek Service & Technology Co., Ltd.

Perfecting IT service and favoring clients 'success
锻造凝练IT服务 助推用户事业发展