金融风控场景的使用案例

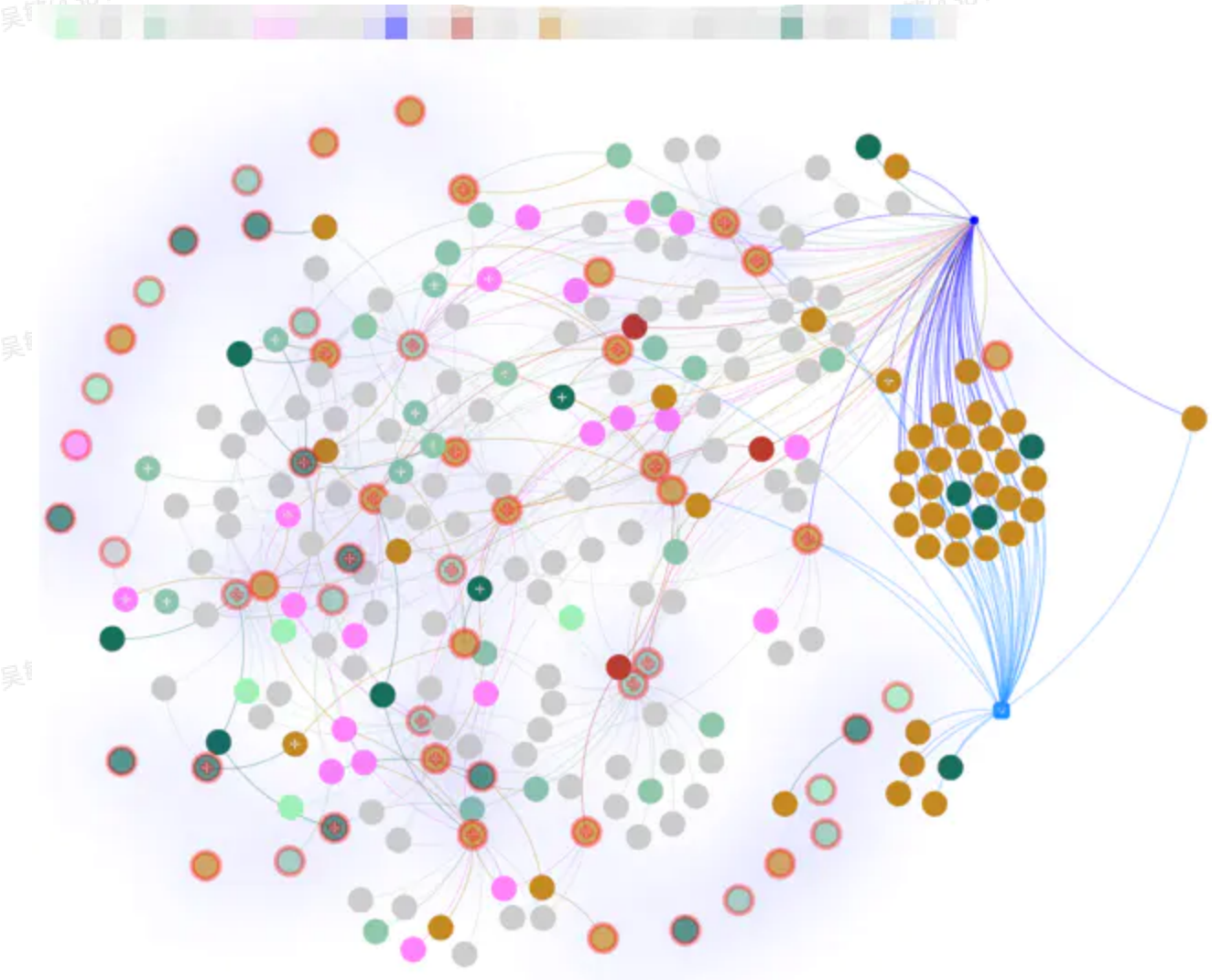
背景介绍

数据原型

查询

附录：

本文示例数据



本篇文章将介绍 Nebula Graph [1]联手某金融公司[2]打造智能风控平台服务的故事。目前的图数据量为顶点 20 亿，边 200 亿的规模。

背景介绍

与 Nebula Graph 共同打造的人工智能驱动的金融科技平台致力于通过科技创新为广大个人和家庭提供安全、快捷、普惠的金融服务，创造更美好的生活。

目标： 通过分析交易行为识别欺诈交易特征，实现在交易中实时探测欺诈特征，阻断可疑交易。

面临挑战： 要在金融交易发生的短时间内实时探测出可疑交易并返回结果，面临如下挑战：

1. 实时并发读写。
2. 由于是核心链路上的核心 DB，对可用性要求极高。
3. 欺诈交易涉及的数据具有高度复杂的关联关系。
4. 保证实时遍历复杂关联数据的性能，在交易时间内返回分析结果。
5. 适应欺诈手段和规模的不断变迁。

解决方案： 借助原生图数据库 Nebula Graph，保证高效、实时遍历高度关联的复杂数据。并且 Nebula Graph 灵活的数据模型，在新增业务逻辑时不影响现有的逻辑。作为高可用的数据库，Nebula Graph 可以保障关键业务不断线。

数据原型

基于 该使用场景，我们可以使用 Nebula Graph 构建以下 schema：

- tag。3 种 tag，即 3 种类型的点，包括 applicant（申请人）、device（设备） 和 wifi。
- edge type。3 种边类型，申请人与申请人之间的联系 (applicant)-[know]->(applicant)，
  申请人使用设备 (applicant)-[use]->(device)，
  设备接入wifi (device)-[addTo]->(wifi)。

Tag	属性	属性数据类型	说明
applicant	id	int	用户 ID
–	name	string	用户名
–	info	bool	是否完成身份认证
–	overdue	bool	是否有逾期历史
device	id	int	设备 ID

–	type	string	设备类型
wifi	ip	string	wifi IP 地址

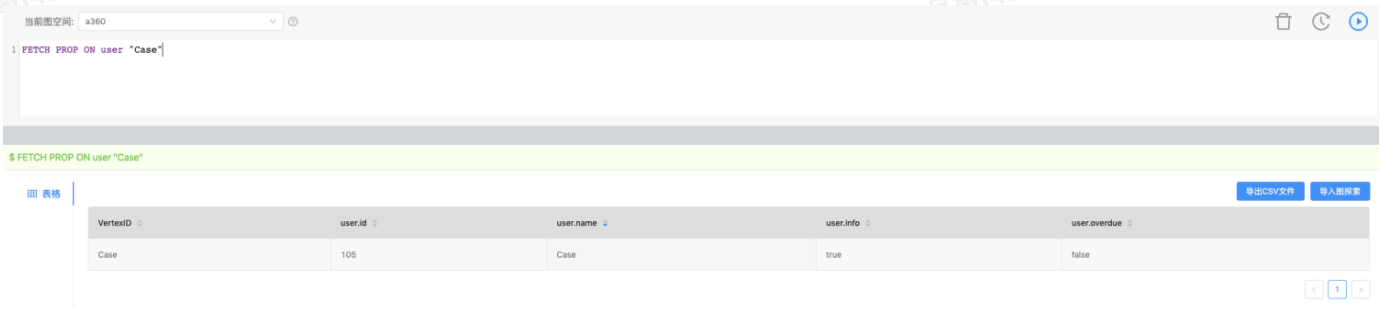
边类型	起点	终点
know	applicant	applicant
use	applicant	device
addTo	device	wifi

查询

有申请者申请贷款的时候，首先判断申请人是否有逾期历史，如果有，则直接拒绝放货。如果没有，则继续接下来的流程。

1. 比如~~申请人 Case~~，首先查询其是否有逾期历史。

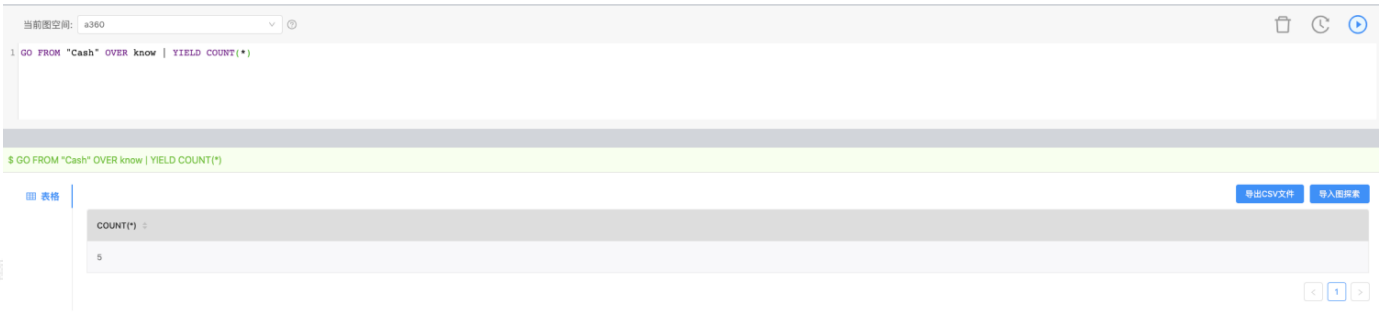
```
1 FETCH PROP ON applicant "Case";
```



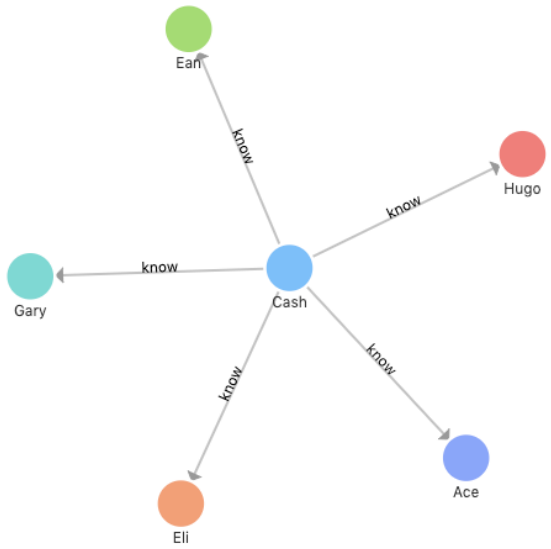
可以看到 overdue 选项为 false，说明未逾期。

2. 接下来查询申请人 Case 有多少哪些一度好友，以及一度好友中有多少哪些人有逾期历史。

```
1 GO FROM "Cash" OVER know | YIELD COUNT(*);
```



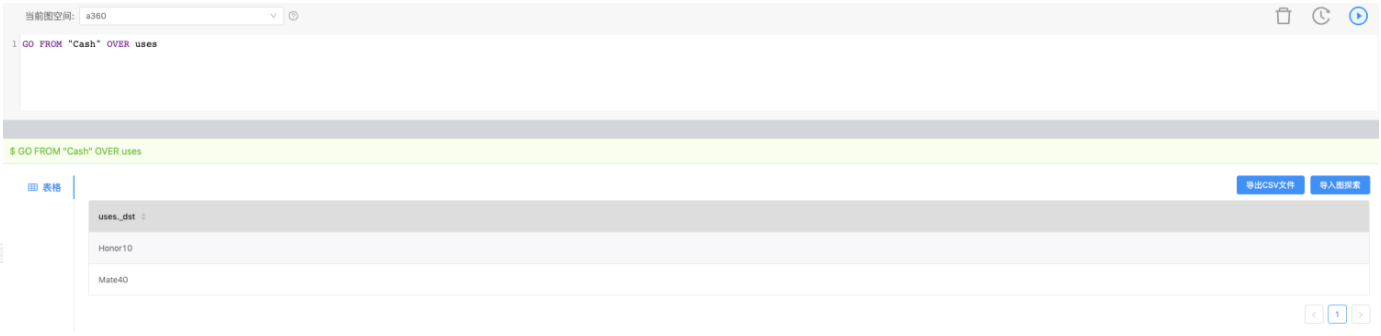
可视化效果见下图



3. 统计申请人的一度好友有几个人有逾期历史

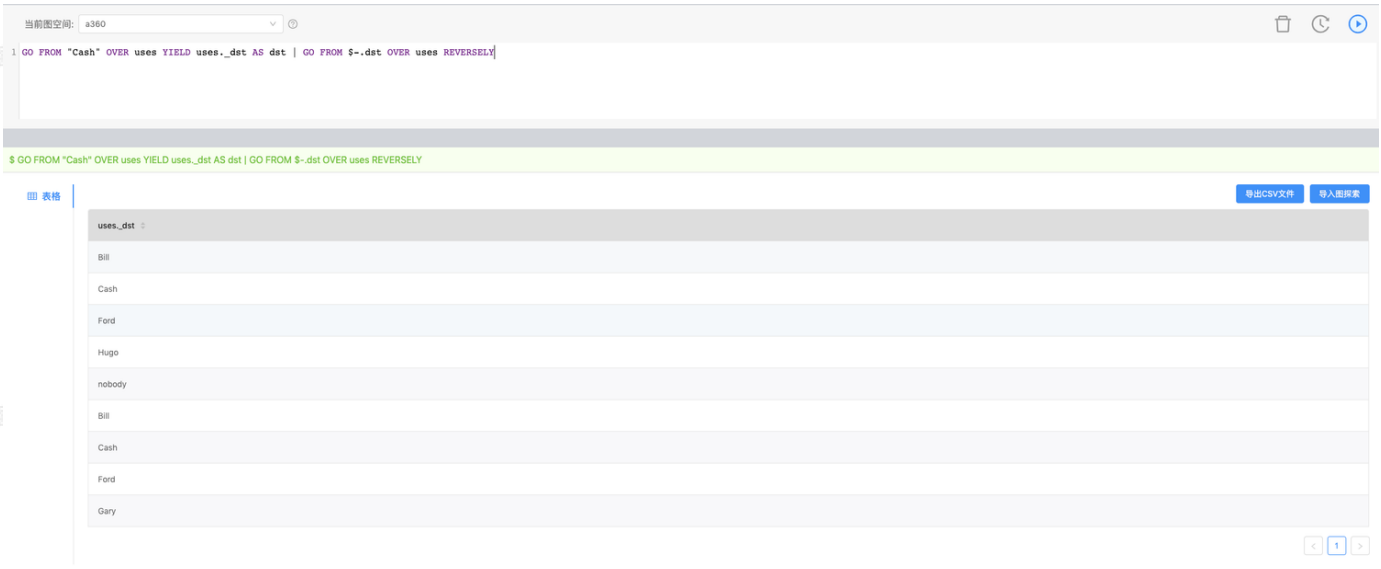
```
GO FROM "Cash" OVER know WHERE $$.applicant.overdue == TRUE | YIELD COUNT(*);
```

4. 查看申请人使用过哪些设备

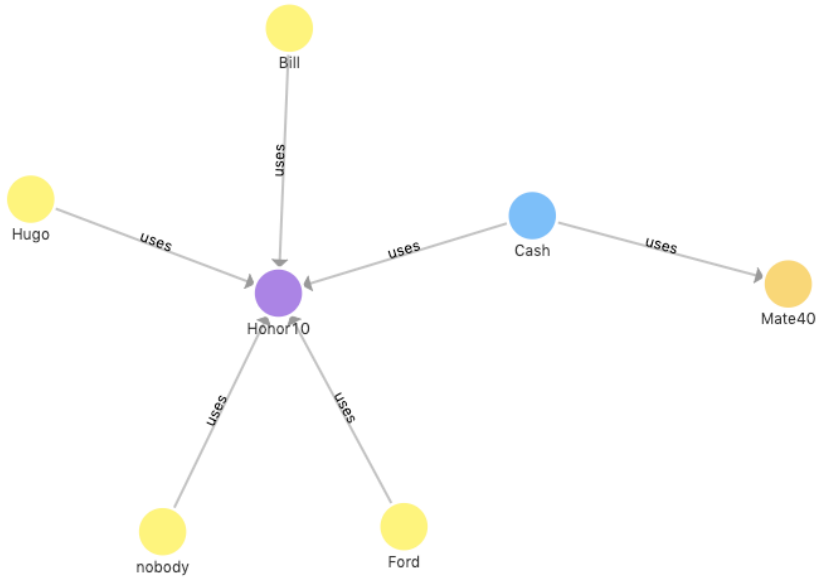


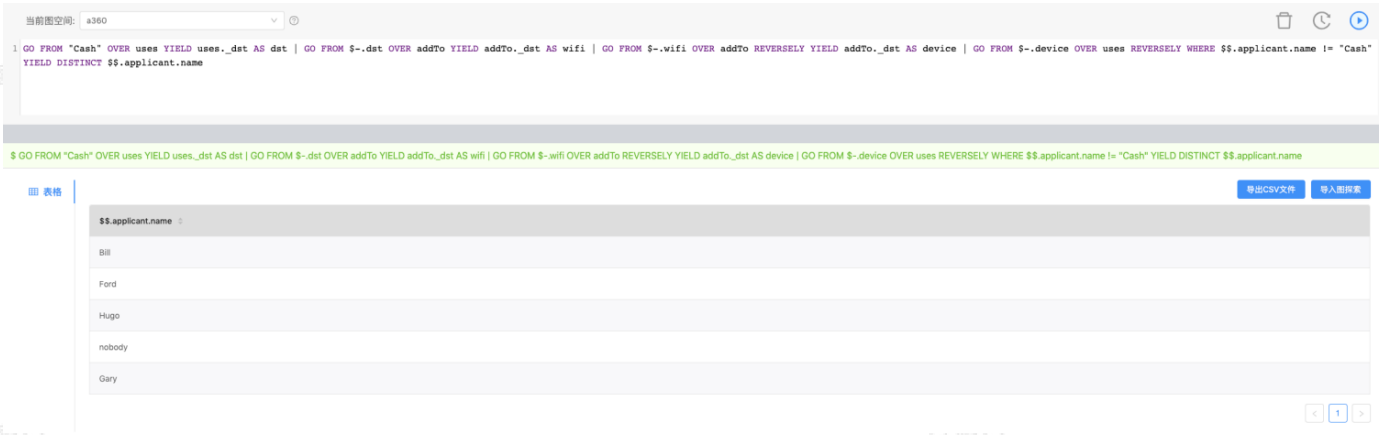
可视化效果见下图

5. 查看有多少人和该申请人使用同一个 wifi。

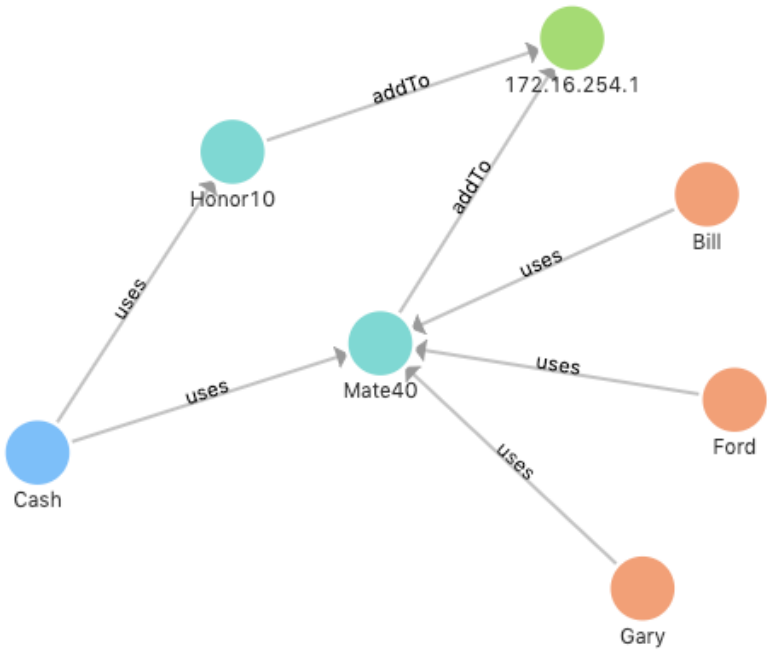


可视化效果见下图


Bill
uses
Hugo
uses
uses
Cash
uses
Mate40
Honor10
uses
uses
nobody
Ford


当前图空间: a360
1 GO FROM "Cash" OVER uses YIELD uses._dst AS dst | GO FROM $-.dst OVER addTo YIELD addTo._dst AS wifi | GO FROM $-.wifi OVER addTo REVERSELY YIELD addTo._dst AS device | GO FROM $-.device OVER uses REVERSELY WHERE $$.applicant.name != "Cash" YIELD DISTINCT $$.applicant.name
$ GO FROM "Cash" OVER uses YIELD uses._dst AS dst | GO FROM $-.dst OVER addTo YIELD addTo._dst AS wifi | GO FROM $-.wifi OVER addTo REVERSELY YIELD addTo._dst AS device | GO FROM $-.device OVER uses REVERSELY WHERE $$.applicant.name != "Cash" YIELD DISTINCT $$.applicant.name
表格
导出CSV文件
导入图探索
$$.applicant.name
Bill
Ford
Hugo
nobody
Gary
1

可视化效果见下图



6. 还可以查询申请人 Cash 的二度好友，以及二度好友中有多少好友逾期。

```
GO 2 STEPS FROM "Cash" OVER know | YIELD COUNT(*);
+----------+
| COUNT(*) |
+----------+
| 3        |
+----------+

GO 2 STEPS FROM "Cash" OVER know WHERE $$.applicant.overdue == TRUE | YIELD COUNT(*);
Empty set (time spent 4513/5295 us)
```

附录：

[1] Nebula Graph https://nebula-graph.com.cn/
[2] 案例背景 https://mp.weixin.qq.com/s/1yhW2GYhBkatNnwvr_mNZQ
[3] 批量数据导入工具 https://docs.nebula-graph.com.cn/nebula-exchange/about-exchange/ex-ug-what-is-exchange/
[4] 可视化工具 https://docs.nebula-graph.com.cn/nebula-studio/about-studio/st-ug-what-is-graph-studio/

本文示例数据

```
--DDL

CREATE SPACE IF NOT EXISTS a360(partition_num = 1, vid_type = FIXED_STRING(30));
USE a360;

CREATE TAG IF NOT EXISTS applicant(id int, name string, info bool, overdue bool);
CREATE TAG IF NOT EXISTS device(id int, name string);
CREATE TAG IF NOT EXISTS wifi(ip string);
CREATE EDGE IF NOT EXISTS know();
CREATE EDGE IF NOT EXISTS uses();
CREATE EDGE IF NOT EXISTS addTo();

--DML

INSERT VERTEX applicant (id, name, info, overdue) VALUES "nobody":(100,"nobody", FALSE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Aaron":(101,"Aaron", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Ace":(102,"Ace", TRUE, FALSE);
```

```
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Ben":(103,"Ben", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Bill":(104,"Bill", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Case":(105,"Case", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Cash":(106,"Cash", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Dan":(107,"Dan", FALSE, TRUE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Don":(108,"Don", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Ean":(109,"Ean", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Eli":(110,"Eli", FALSE, TRUE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Ford":(111,"Ford", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Gary":(112,"Gary", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Hugo":(113,"Hugo", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Ian":(114,"Ian", TRUE, FALSE);
INSERT VERTEX applicant (id, name, info, overdue) VALUES "Jan":(115,"Jan", TRUE, FALSE);

INSERT VERTEX device(id, name) VALUES "Honor10":(200, "Honor10");
INSERT VERTEX device(id, name) VALUES "Mate40":(201, "Mate40");
INSERT VERTEX device(id, name) VALUES "iPad10":(202, "iPad10");
INSERT VERTEX device(id, name) VALUES "MacPro":(203, "MacPro");

INSERT VERTEX wifi(ip) VALUES "wifi1":("172.16.254.1");
INSERT VERTEX wifi(ip) VALUES "wifi2":("129.12.203.1");
INSERT VERTEX wifi(ip) VALUES "wifi3":("172.16.254.2");
INSERT VERTEX wifi(ip) VALUES "wifi4":("192.26.204.1");

INSERT EDGE know() VALUES "nobody" -> "Aaron":();
INSERT EDGE know() VALUES "nobody" -> "Bill":();
INSERT EDGE know() VALUES "nobody" -> "Ford":();
```

```
INSERT EDGE know() VALUES "nobody" -> "Ean":();
INSERT EDGE know() VALUES "Cash" -> "Ean":();
INSERT EDGE know() VALUES "Cash" -> "Gary":();
INSERT EDGE know() VALUES "Cash" -> "Hugo":();
INSERT EDGE know() VALUES "Ian" -> "Ean":();
INSERT EDGE know() VALUES "Gary" -> "Hugo":();
INSERT EDGE know() VALUES "Hugo" -> "Ford":();
INSERT EDGE know() VALUES "Cash" -> "Eli":();
INSERT EDGE know() VALUES "Cash" -> "Ace":();
INSERT EDGE know() VALUES "Hugo" -> "Cash":();


INSERT EDGE uses() VALUES "nobody" -> "Honor10":();
INSERT EDGE uses() VALUES "Bill" -> "Honor10":();
INSERT EDGE uses() VALUES "Ford" -> "Honor10":();
INSERT EDGE uses() VALUES "Cash" -> "Honor10":();
INSERT EDGE uses() VALUES "Hugo" -> "Honor10":();
INSERT EDGE uses() VALUES "Gary" -> "Mate40":();
INSERT EDGE uses() VALUES "Ford" -> "Mate40":();
INSERT EDGE uses() VALUES "Cash" -> "Mate40":();
INSERT EDGE uses() VALUES "Bill" -> "Mate40":();
INSERT EDGE uses() VALUES "Ian" -> "MacPro":();
INSERT EDGE uses() VALUES "Ean" -> "MacPro":();
INSERT EDGE uses() VALUES "Eli" -> "iPad10":();
INSERT EDGE uses() VALUES "Dan" -> "MacPro":();

INSERT EDGE addTo() VALUES "Honor10" -> "wifi1":();
INSERT EDGE addTo() VALUES "Mate40" -> "wifi1":();
INSERT EDGE addTo() VALUES "iPad10" -> "wifi2":();
INSERT EDGE addTo() VALUES "MacPro" -> "wifi3":();
INSERT EDGE addTo() VALUES "MacPro" -> "wifi4":();

CREATE TAG INDEX IF NOT EXISTS applicant_name ON applicant(name(20));
CREATE TAG INDEX IF NOT EXISTS applicant_info ON applicant(info);
CREATE TAG INDEX IF NOT EXISTS device_name ON device(name(20));
CREATE TAG INDEX IF NOT EXISTS ip ON wifi(ip(20));

REBUILD TAG INDEX applicant_name;
REBUILD TAG INDEX device_name;
```

```
84 REBUILD TAG INDEX ip;
85 REBUILD TAG INDEX applicant_info;
```

实际生产系统中，数据是通过应用端实时写入，并在数仓中存留一份。