如何将图模型整合到已有关系型数据库中？

龙恒

PingCAP

Staff Software Engineer

目录

图数据库 01

关系型数据库 02

图 + 关系型多模数据库 03

DEMO 04

特性 05

Benchmark 06

第一部分

图数据库

图数据库

目前常见的使用图数据库场景有:

- 风控(欺诈检测/反洗钱)
- 图神经网络
- 知识图谱
- 社交网络

目前的问题

- 对于复杂的关系网络，传统关系型数据库无能为力
- 单独部署图数据库集群
- 部署运维两套数据库集群成本太高
- 在两个不同的数据库中数据一致性不能保障

探索方向

TiGraph 项目尝试验证在分布式关系型数据中无缝集成图模式：

- 同时包含关系型模型和图模型
- 同一个事务中操作图数据和关系型数据的能力
- 将图遍历作为 SQL 子查询（反之亦然）
- 在 SQL 中扩展出一个让 DBA 一眼就能学会的图遍历语法
- 对于 N 度人脉的场景性能对比

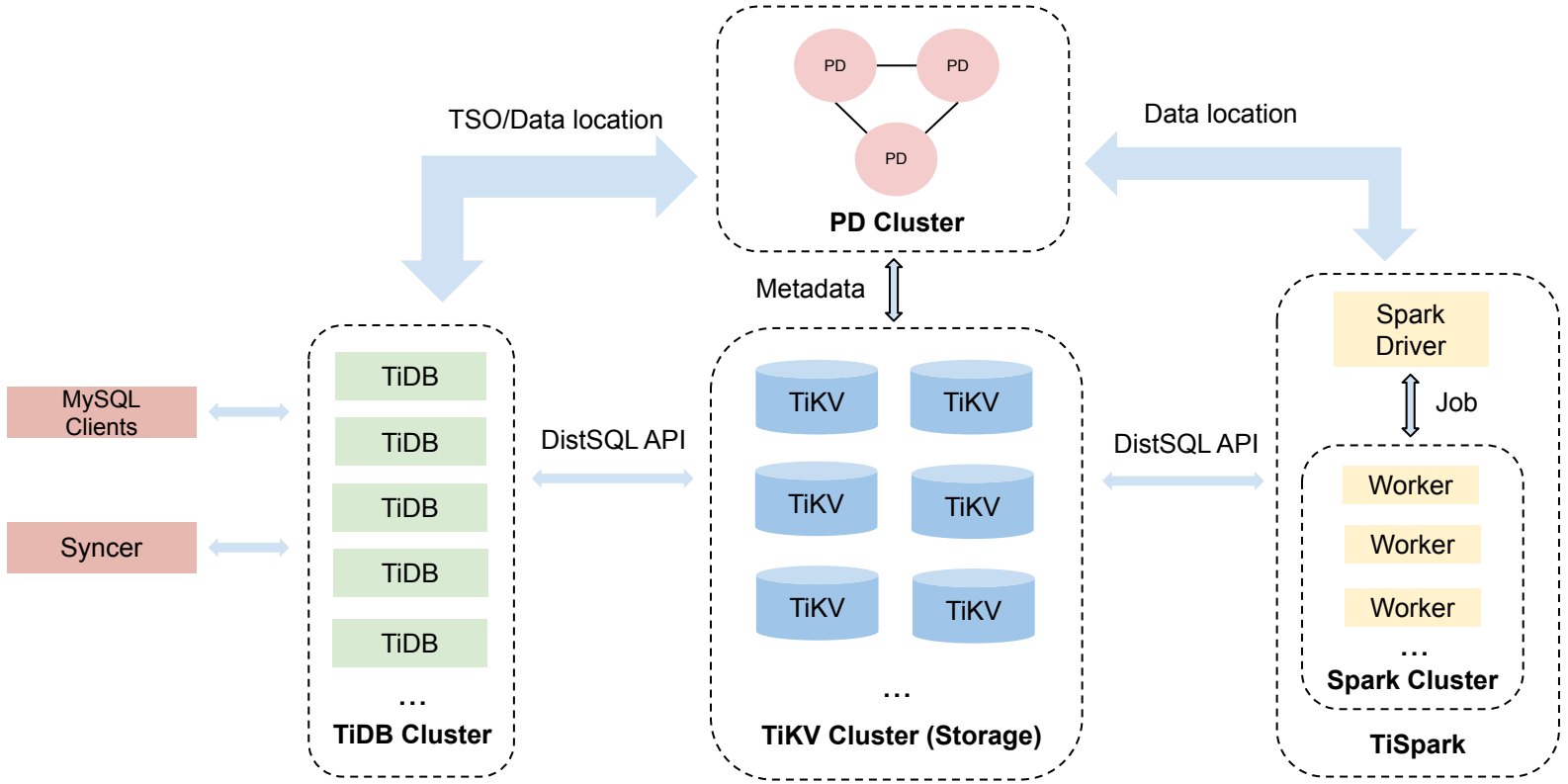
第二部分

关系型数据库

关系型数据库本质

- For a row in a Table, row data is encoded in key-value pairs with the format below:
  - t<<tableID>>_r<<rowID>> => [col1, col2, col3, col4]
- If there is secondary index with a column, the index data of a row is encoded in this way:
  - t<<tableID>>_i<<indexID>>_indexedColumnsValue => rowID
  - t<<tableID>>_i<<indexID>>_indexedColumnsValue_rowID => nil

TiDB 架构


PD
PD
PD
PD Cluster
TSO/Data location
Data location
Metadata
MySQL Clients
Syncer
TiDB
TiDB
TiDB
TiDB
TiDB
...
TiDB Cluster
DistSQL API
TiKV
TiKV
TiKV
TiKV
TiKV
TiKV
...
TiKV Cluster (Storage)
DistSQL API
Spark Driver
Job
Worker
Worker
Worker
...
Spark Cluster
TiSpark

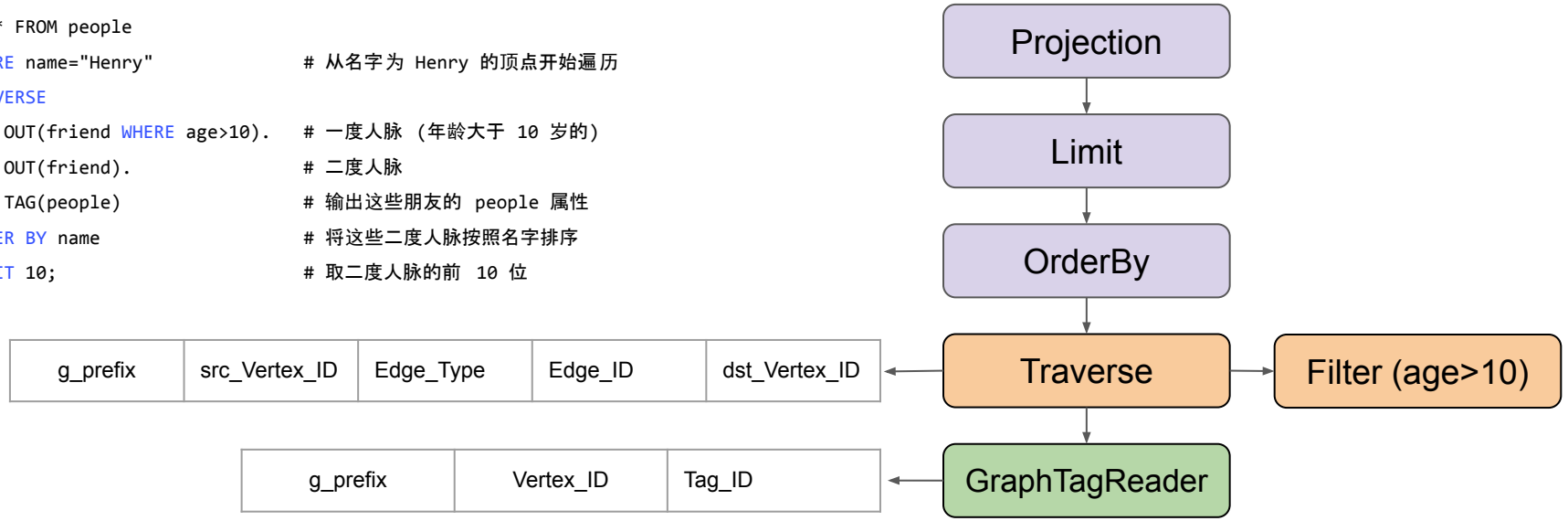
“

《三国演义》

天下大势，合久必分，分久必合

图 + 关系型多模实现

```
SELECT * FROM people
    WHERE name="Henry"              # 从名字为 Henry 的顶点开始遍历
    TRAVERSE
        OUT(friend WHERE age>10).   # 一度人脉 (年龄大于 10 岁的)
        OUT(friend).                # 二度人脉
        TAG(people)                 # 输出这些朋友的 people 属性
    ORDER BY name                   # 将这些二度人脉按照名字排序
    LIMIT 10;                       # 取二度人脉的前 10 位
```



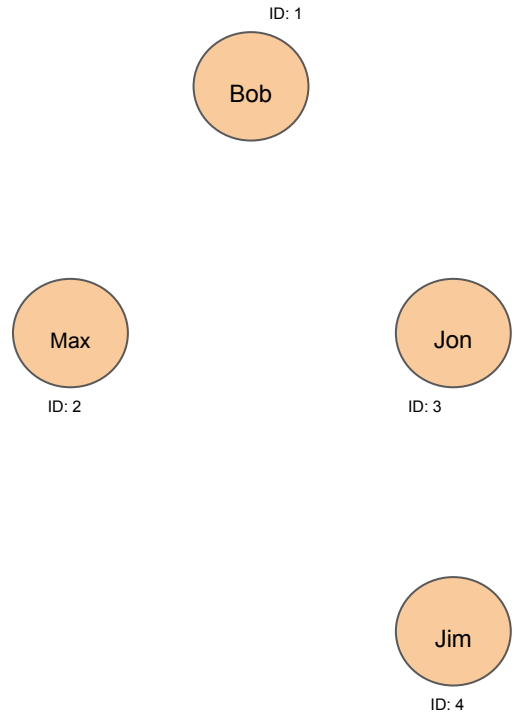
https://github.com/tigraph/tidb

第三部分

DEMO

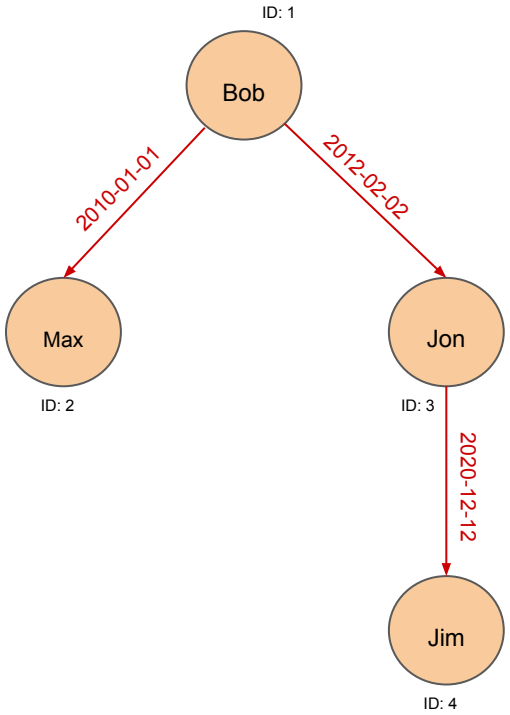
DEMO

```
-- 创建 people 点
CREATE TAG people (id BIGINT, name VARCHAR(32));

-- 写入 4 个点
INSERT INTO people VALUES
(1, 'Bob'),(2,'Max'),(3,'Jon'),(4,'Jim');
```



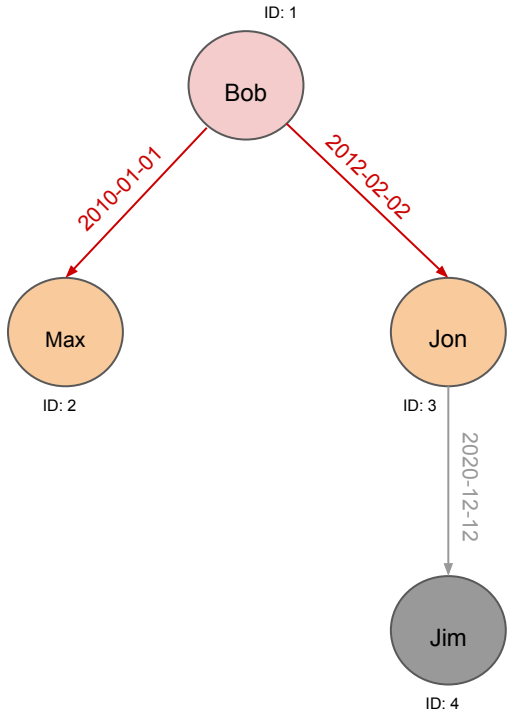
DEMO

```
-- 创建 follow 边
CREATE EDGE follow (src BIGINT, dst BIGINT, time DATE);

-- 写入 3 条边
INSERT INTO follow VALUES (1, 2, '2010-01-01'),
                          (1, 3, '2012-02-02'),
                          (3, 4, '2020-12-12');
```



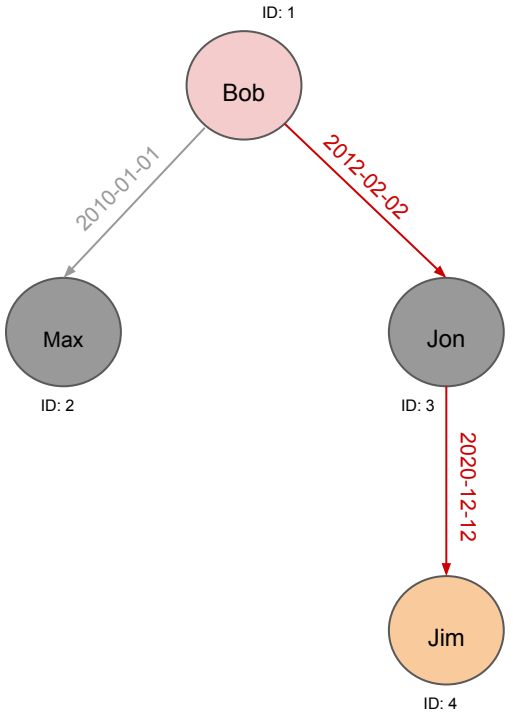
DEMO

```
-- 查询 Bob 关注的人(Bob 的 1 度人脉)
SELECT * FROM people
     WHERE name = 'Bob'
     TRAVERSE
          OUT(follow).
          TAG(people);
```



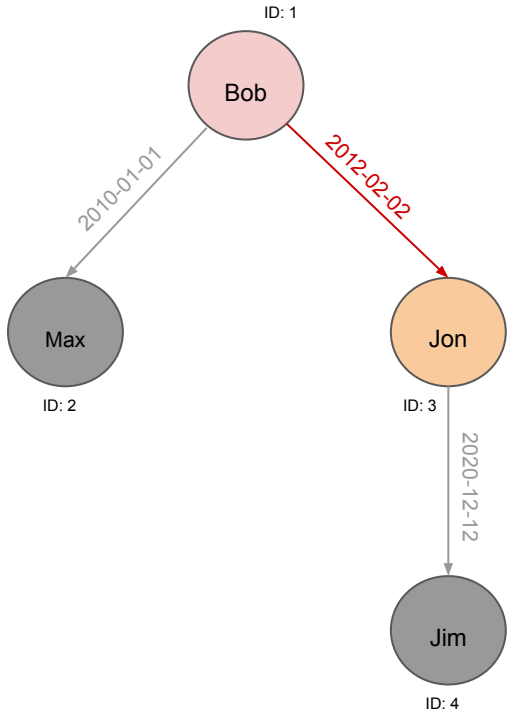
DEMO

```
-- 查询 Bob 的 2 度人脉
SELECT * FROM people
     WHERE name = 'Bob'
     TRAVERSE
          OUT(follow).
          OUT(follow).
          TAG(people);
```



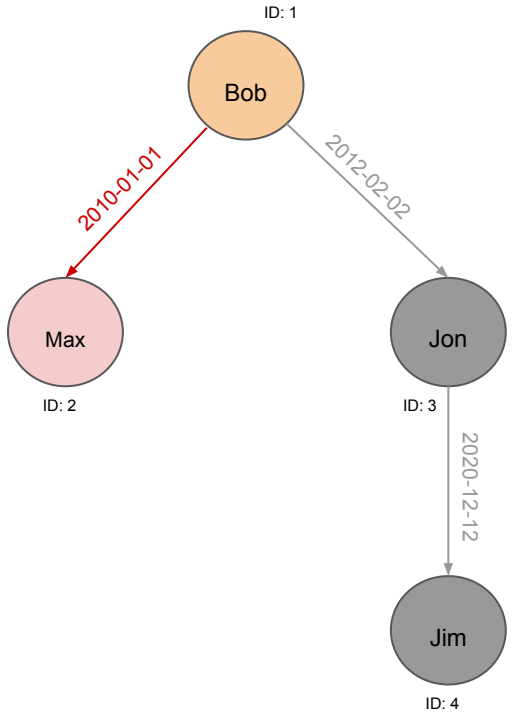
DEMO

```
-- 查询 Bob 在 2012 年关注的人
SELECT * FROM people
     WHERE name = 'Bob'
     TRAVERSE
          OUT(follow WHERE YEAR(time) = 2012).
          TAG(people);
```



DEMO

```
-- 查询关注了 Max 了的人
SELECT * FROM people
     WHERE name = 'Max'
     TRAVERSE
          IN(follow).
          TAG(people);
```



第四部分

特性

一些主要特性

- ✓ DDL
  - ✓ Create tag
  - ✓ Create edge
  - ✓ Secondary index
- ✓ DML
  - ✓ Insert
  - ✓ Update
  - ✓ Delete
  - ✓ Insert on duplicate
  - ✓ Insert ignore
  - ✓ Secondary index
- ❑ Query
  - ✓ Secondary index
  - ✓ Selection
  - ✓ Order by
  - ❑ Group by

- ✓ Query
  - ✓ Limit
  - ✓ Expression Compatibility
  - ✓ Join
  - ✓ Subquery
  - ✓ Graph traverse
    - ✓ Out direction
    - ✓ In direction
    - ✓ Bidirection
  - ✓ Edge traverse
  - ✓ Distinct

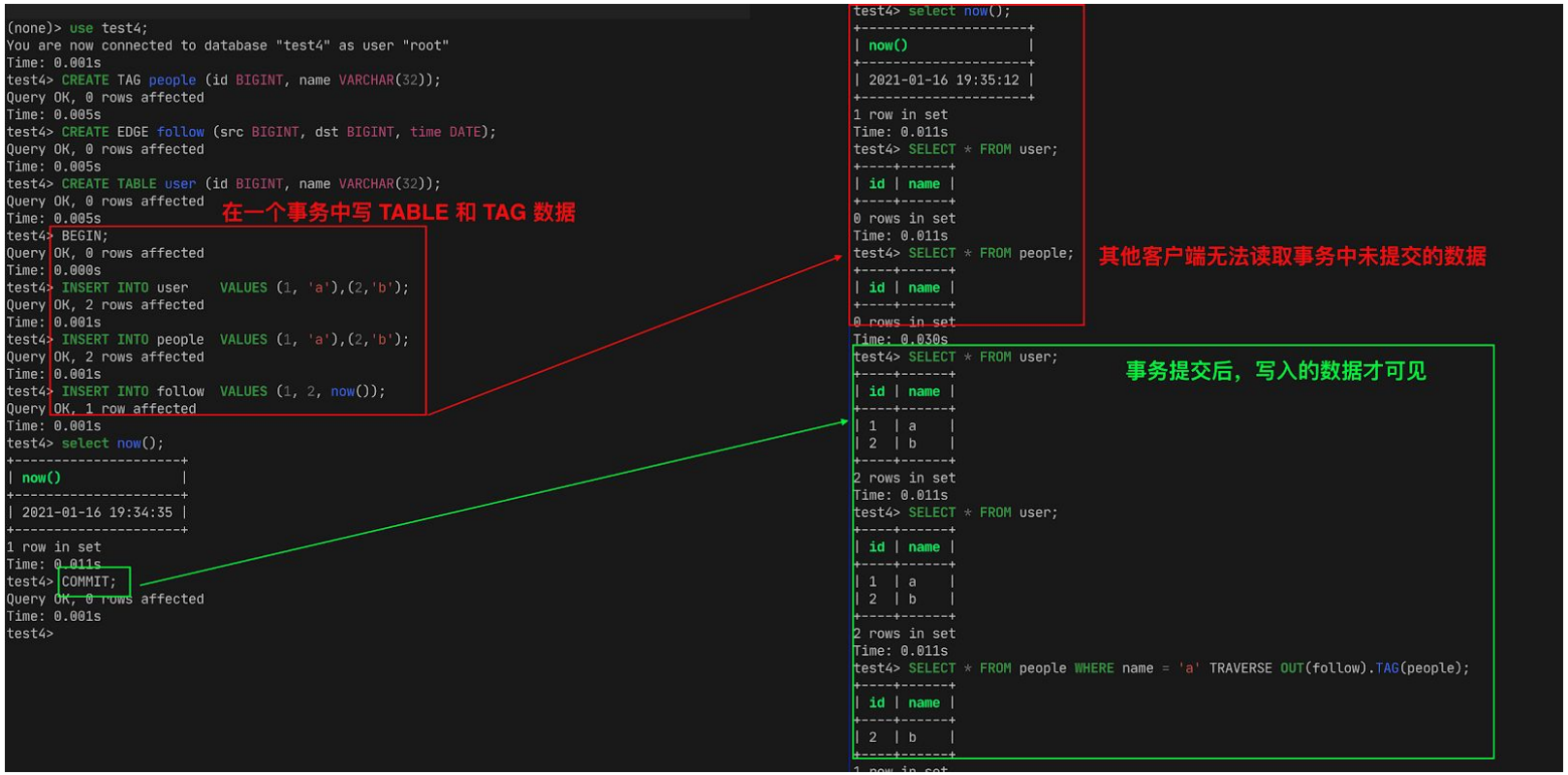
Index

```
test3> desc SELECT * FROM people WHERE name = 'Bob' TRAVERSE OUT(follow).TAG(people);
+---------------+---------+------+-----------------------------+---------------+
| id            | estRows | task | access object               | operator info |
+---------------+---------+------+-----------------------------+---------------+
| Traverse_7    | 1.00    | root |                             |               |
| └─Point_Get_6 | 1.00    | root | tag:people, index:name(name) |               |
+---------------+---------+------+-----------------------------+---------------+
2 rows in set
Time: 0.011s
test3> select * from people;
+----+------+---------------------+
| id | name | register            |
+----+------+---------------------+
| 1  | Bob  | 2010-01-16 18:00:00 |
| 2  | Max  | 2011-01-15 18:00:00 |
| 3  | Jon  | 2012-01-14 18:00:00 |
| 4  | Jim  | 2020-01-13 18:00:00 |
+----+------+---------------------+
4 rows in set
Time: 0.118s
test3> desc SELECT * FROM people use index (register) WHERE register='2010-01-16 18:00:00' TRAVERSE OUT(follow).TAG(people);
+-----------------------------+---------+-----------+-----------------------------------+-----------------------------------------------------------------------------------------------------+
| id                          | estRows | task      | access object                     | operator info                                                                                       |
+-----------------------------+---------+-----------+-----------------------------------+-----------------------------------------------------------------------------------------------------+
| Traverse_9                  | 0.00    | root      |                                   |                                                                                                     |
| └─IndexLookUp_8             | 0.00    | root      |                                   |                                                                                                     |
|   ├─IndexRangeScan_6(Build) | 0.00    | cop[tikv] | tag:people, index:register(register) | range:[2010-01-16 18:00:00,2010-01-16 18:00:00], keep order:false, stats:pseudo                     |
|   └─GraphTagIDScan_7(Probe) | 0.00    | cop[tikv] | tag:people                        | keep order:false, stats:pseudo                                                                      |
+-----------------------------+---------+-----------+-----------------------------------+-----------------------------------------------------------------------------------------------------+
```

用 Unique Index 走 point-get 优化

用 IndexLookUp 避免 “全表扫”

事务



- 支持分布式事务。
- 可以在同一个事务里面同时操作 TABLE 和 TAG.

子查询

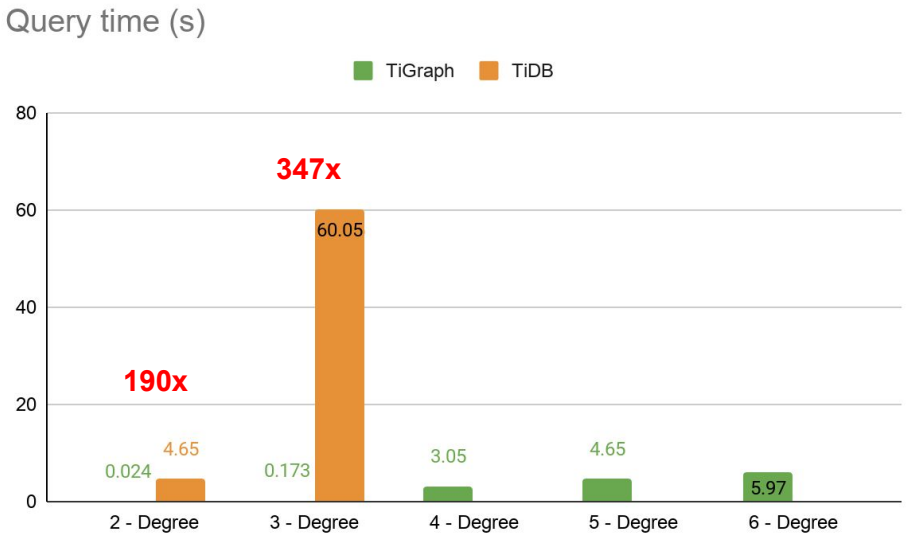
```
mysql> CREATE TABLE orders(id bigint PRIMARY KEY AUTO_INCREMENT, user_id bigint, item_id bigint);
mysql> CREATE TAG user(id bigint, name varchar(64));
mysql> CREATE EDGE contact(src bigint, dst bigint);

mysql> INSERT INTO USER VALUES (1, "UserA"), (2, "UserA"), (3, "UserA"), (4, "UserA"),  (5, "UserA"), (6, "UserA");
mysql> INSERT INTO contact VALUES (1, 2),  (2, 3), (3, 4), (2, 5),  (5, 6), (4, 7);
mysql> INSERT INTO orders(user_id, item_id) VALUES (4, 19230), (6, 4323);

mysql> -- 查找某一个用户的三度人脉的下单记录
mysql> SELECT * FROM orders WHERE user_id IN (SELECT id FROM USER WHERE id=1 traverse OUT(contact).OUT(contact).OUT(contact).TAG(USER) LIMIT 10);
+------+-----------+-----------+
|   id |   user_id |   item_id |
|------+-----------+-----------|
|    1 |         4 |     19230 |
|    2 |         6 |      4323 |
+------+-----------+-----------+
2 rows in set (0.003s)
```

图遍历结果作为子查询

第五部分

Benchmark

Benchmark



- Workload
  - 24vc + 64G
  - Unistore
  - N 度人脉
  - 数据规模
    - 100k vertices or 100k rows
    - 30 - 100 friends (6.5m edges)
- Notes
  - TiDB 4-degree > 7h (cancelled)

```
+----+------+-----------+------+---------+-------+------------+-----------+
| Id | User | Host      | db   | Command | Time  | State      | Info      |
+----+------+-----------+------+---------+-------+------------+-----------+
| 3  | root | 127.0.0.1 | test | Query   | 26637 | autocommit | SELECT cou|
| 11 | root | 127.0.0.1 | test | Query   | 0     | autocommit | show p...|
+----+------+-----------+------+---------+-------+------------+-----------+
```

关系型数据库跑 4 度人脉
至少提升 26637/3.05 = 8733 倍

Thank You

email: heng@lonng.org
wechat: oss_lonng