DTCC

# 第十一届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2020

## 架构革新 高效可控

IT168.com ChinaUnix.net ITPUB

北京国际会议中心 | 2020/12/21-12/23

# 分布式图数据库在贝壳找房的应用实践

高攀-贝壳找房搜索平台负责人

2020年12月22日

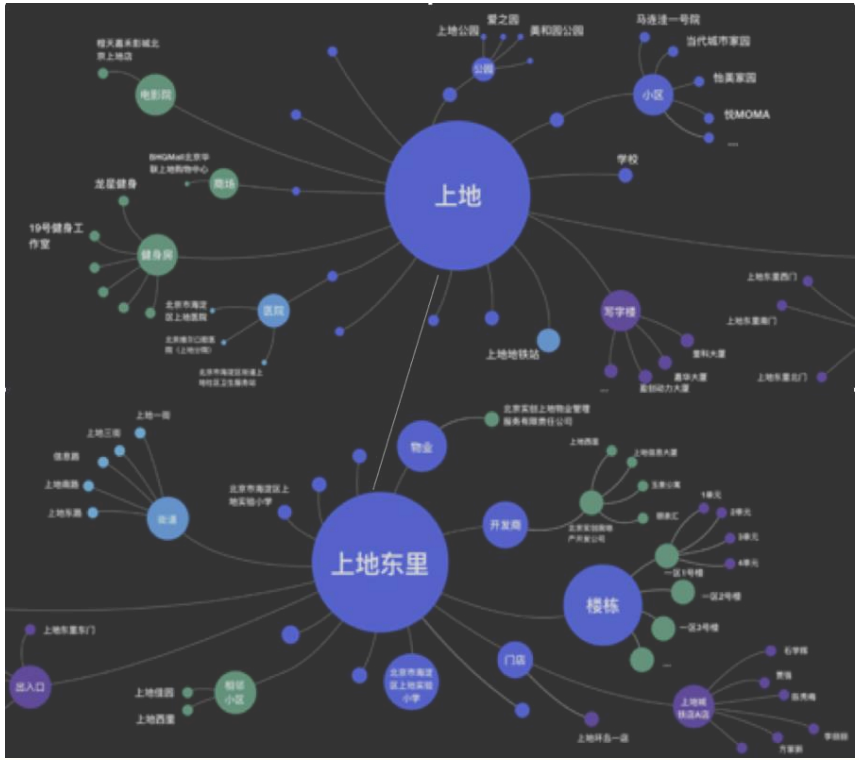
# 分享提纲

- 贝壳图数据库应用场景
- 图数据库技术选型
- 图数据库平台建设
- 原理&优化&不足

# 图数据库场景——行业知识图谱

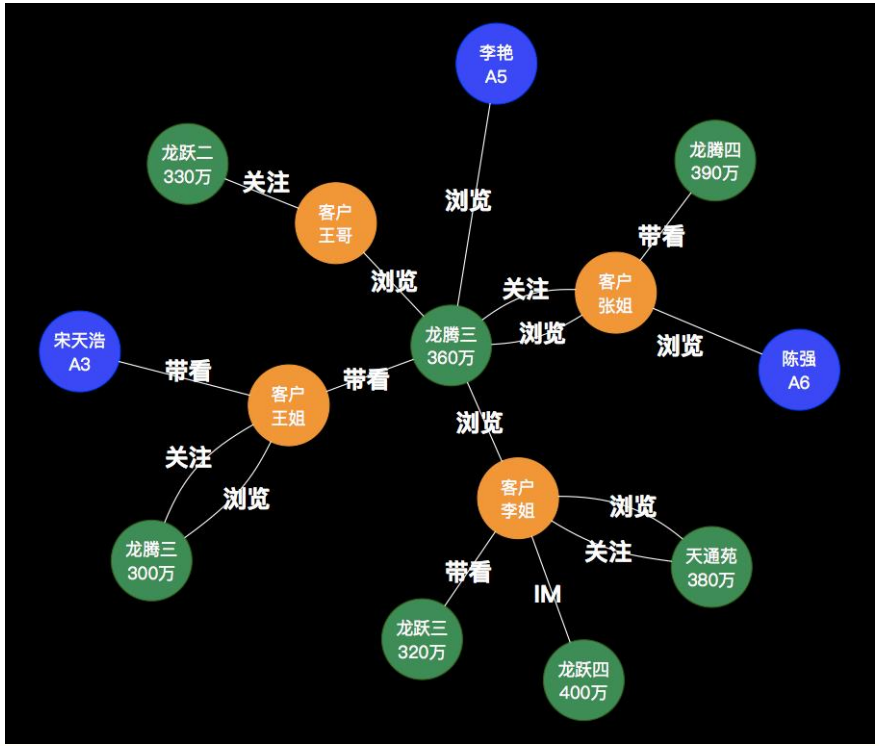
- 覆盖房源、客户、经纪人、开发商、小区、地铁、医院、学校、商场等140多个类别，共计500多亿三元组
- 偏事实关系
- 应用场景：搜索推荐，智能问答

- 例：查询开发商是XXX，小区绿化率大于20%，周边200米有大型商场，500米有地铁，1000米有三甲医院，2000米有升学率超过60%的高中，房价在800W以内，最近被经纪人带看次数最多的房子？？？



# 图数据库场景——贝壳关系图谱

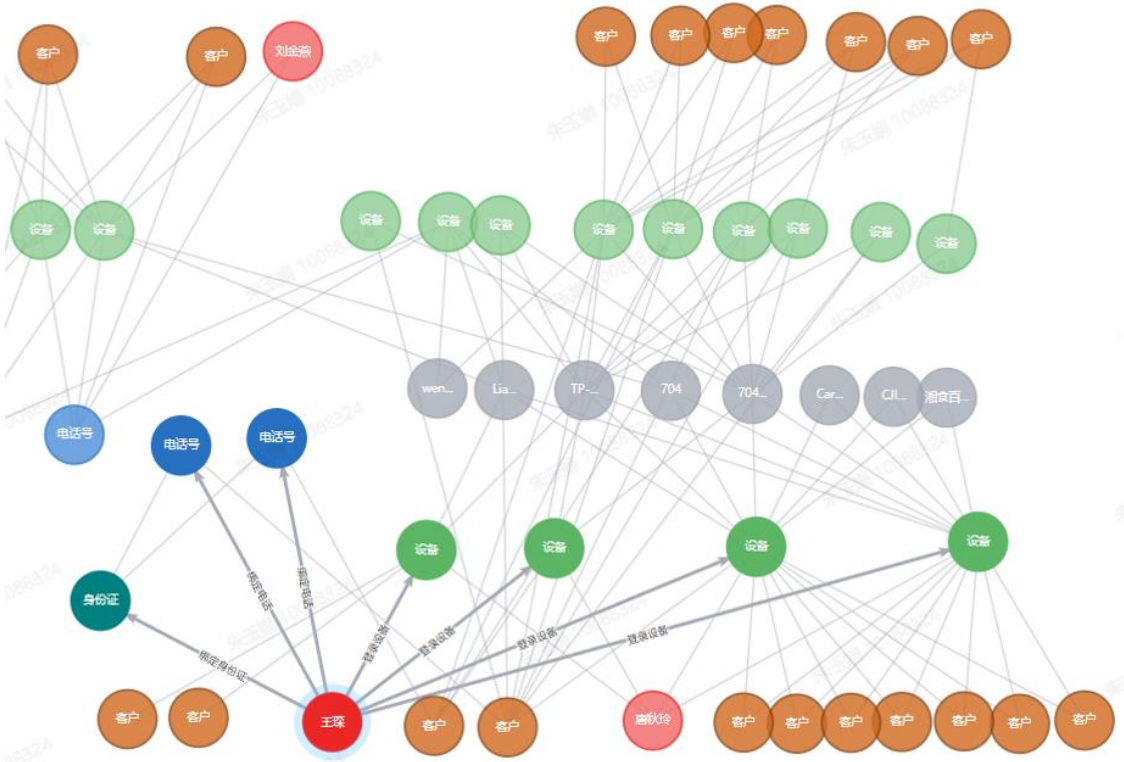
- 节点：经纪人、房源、客户
- 关系：浏览、关注、带看等
- 偏行为关系
- 应用场景：房源推荐、客源维护、影响力分级




- 例：当某个用户经常浏览关注或者咨询某个房源时，该房源的维护人A1会邀请该用户的维护人A2带客户来看房。

# 图数据库场景——风控关系图谱

- 事实图谱、行为图谱、社交图谱、作业图谱、工商图谱……
- 风控场景：虚假房源、虚假客源、虚假带看、私单飞单……



# 贝壳图数据库应用场景

分别使用不同图数据库，各自为战：




行业知识图谱　　贝壳关系图谱　　风控关系图谱

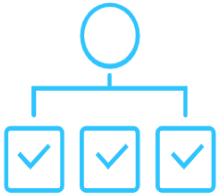
# 分享提纲

- 图数据库在贝壳的应用场景
- 图数据库技术选型
- 图数据库平台建设
- 原理&优化&不足

# 图数据库技术选型

开源

成熟



扩展

文档

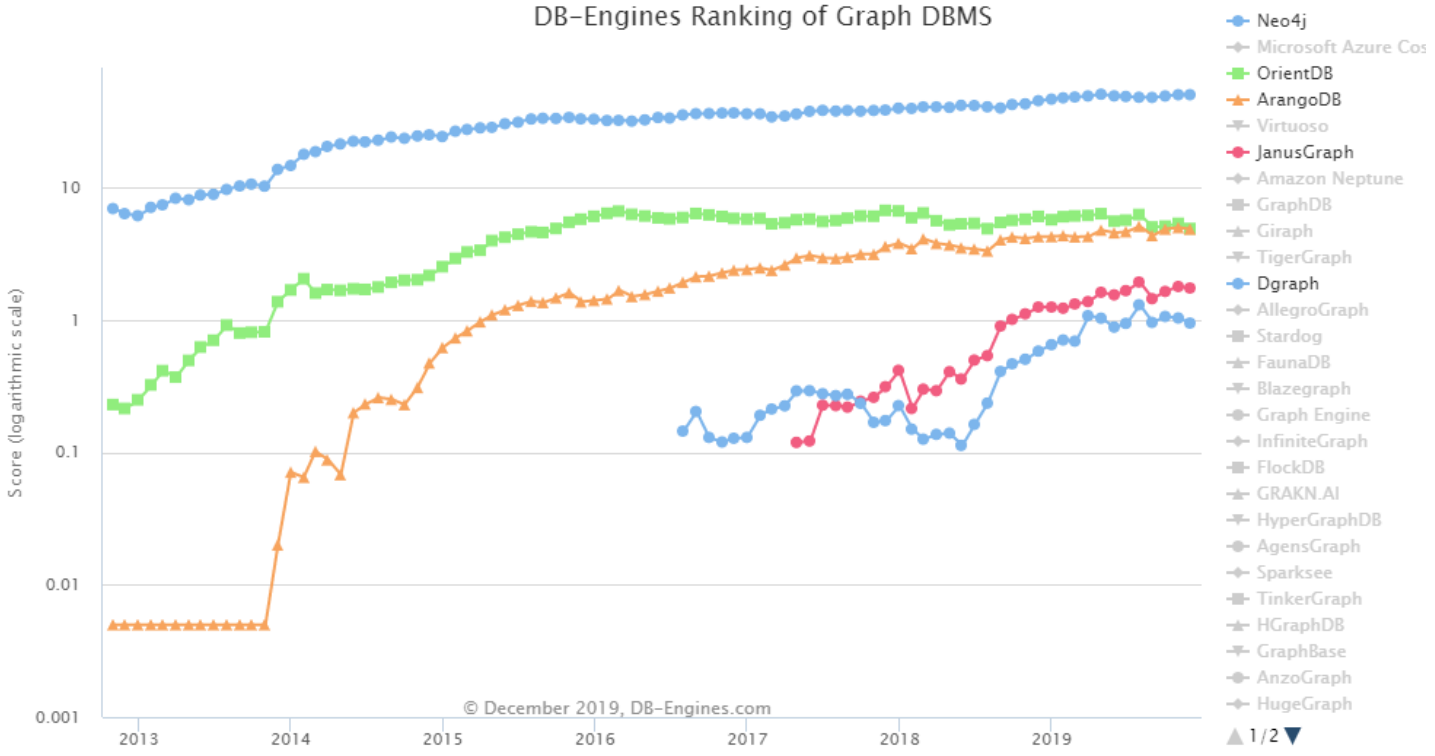
性能

稳定

运维

易用

# 图数据库技术选型

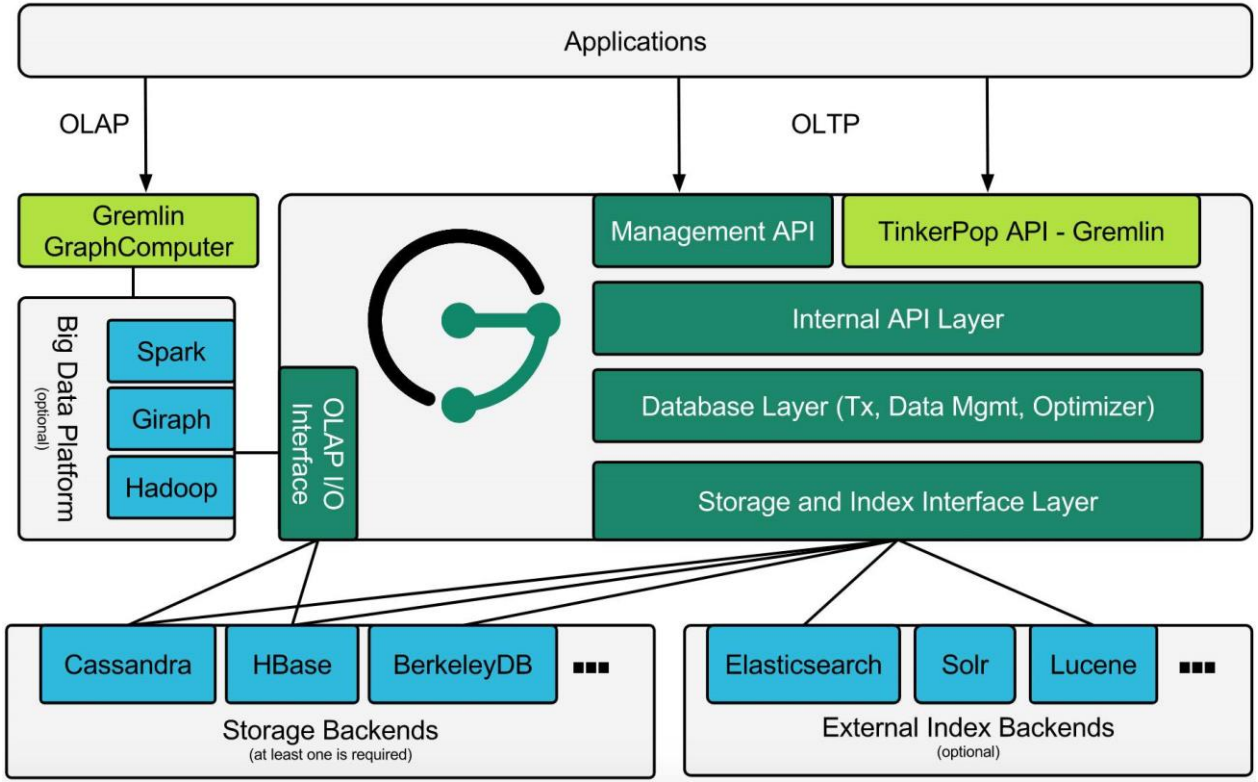

DB-Engines Ranking of Graph DBMS
Score (logarithmic scale)
10
1
0.1
0.01
0.001
2013
2014
2015
2016
2017
2018
2019
© December 2019, DB-Engines.com
Neo4j
Microsoft Azure Cos
OrientDB
ArangoDB
Virtuoso
JanusGraph
Amazon Neptune
GraphDB
Giraph
TigerGraph
Dgraph
AllegroGraph
Stardog
FaunaDB
Blazegraph
Graph Engine
InfiniteGraph
FlockDB
GRAKN.AI
HyperGraphDB
AgensGraph
Sparksee
TinkerGraph
HGraphDB
GraphBase
AnzoGraph
HugeGraph
1/2

# 图数据库技术选型——主流图数据库对比



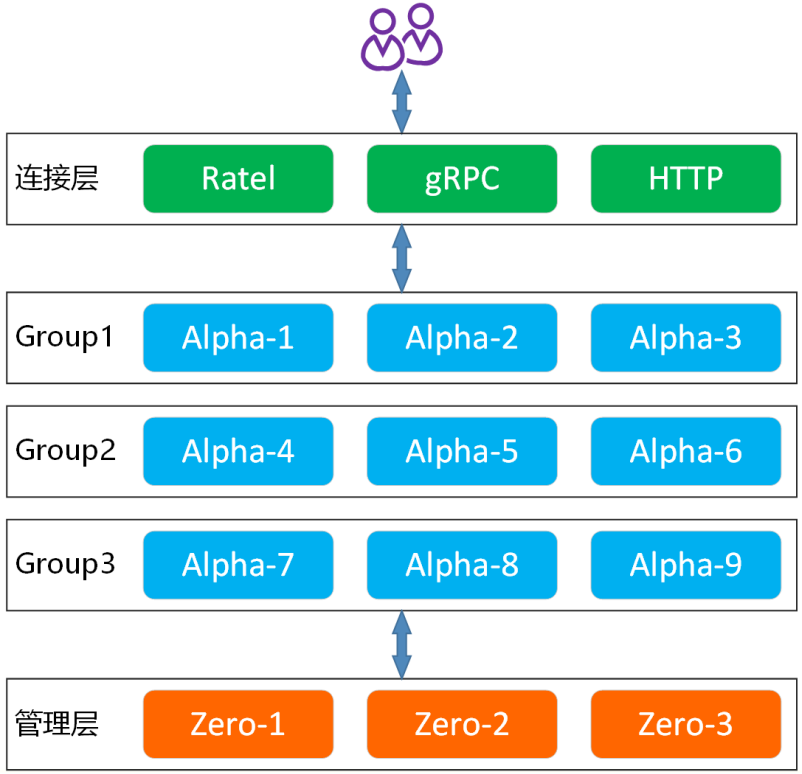
| 图数据库 | Neo4j | OrientDB | ArangoDB | JanusGraph | DGraph |
|---|---|---|---|---|---|
| 初次release | 2007 | 2010 | 2012 | 2017 | 2016 |
| 是否开源 | 社区版开源 | 开源 | 开源 | 开源 | 开源 |
| 是否收费 | 企业版收费 | webUI管理模块收费 | 企业版收费 | 免费 | 免费 |
| 数据模型 | graph | doc、graph、KV | doc、graph、KV | graph | graph |
| SQL | 不支持 | 类SQL | 不支持 | 不支持 | 不支持 |
| 存储系统 | 原生 | 原生 | RockDB | 依赖其他存储 | 原生 |
| 分布式 | 企业版支持 | 后期支持 | 后期支持 | 原生支持 | 原生支持 |
| 相关文档 | 非常多 | 多 | 多 | 少 | 少 |

# 图数据库技术选型——JanusGraph架构



# 图数据库技术选型——Dgraph架构

- Zero：集群大脑，用于控制集群，将服务器分配到一个组，并均衡数据。通过raft选主
- Alpha：存储数据并处理查询，托管谓词和索引
- Group：多个alpha组成一个group，数据分片存储到不同group，每个group内数据通过raft保证强一致性
- Ratel：可视化界面，用户可通过界面来执行查询，更新或修改schema



# 图数据库技术选型——性能对比



| | 类型 | | JanusGraph | Dgraph |
|---|---|---|---|---|
| 写入性能 | 实时写入 | 点 | 15000/s | 35000/s |
| | | 边 | 9000/s | 10000/s |
| | 初始化写入三元组 | | —— | 24W/s |
| 查询性能（随机1W次平均） | 查询结点的所有属性 | | 1.63 ms | 2.24 ms |
| | 查询结点的一度关系 | | 1.25 ms | 2.30 ms |
| | 查询和当前结点关联的所有一度结点 | | 11.84 ms | 3.18 ms |
| | 查询两节点间小于6度的所有最短路径 | | 4.37 ms | 1.03 ms |
| | 查询一度以内所有顶点及属性 | | 36.36 ms | 3.26 ms |
| | 查询二度以内所有顶点及属性 | | 307.07 ms | 3.58 ms |
| | 查询三度以内所有顶点及属性 | | 763.21 ms | 3.76 ms |

测试机器：3台物理机，48核，128G内存，SATA硬盘
测试数据集：4800w点，6300w边，4.5亿三元组，大小30G

# 图数据库技术选型——Dgraph VS JansGraph



| 特性 | Dgraph | JanusGraph |
| --- | --- | --- |
| 架构 | 分布式 | 构建于其他分布式数据库之上 |
| 副本 | 强一致性 | 依赖底层DB |
| 数据均衡 | 自动 | 依赖底层DB |
| 语言 | GraphQL+- | Gremlin |
| 全文检索 | 原生支持 | 依赖外部检索系统 |
| 正则表达式 | 原生支持 | 依赖外部检索系统 |
| 地理位置检索 | 原生支持 | 依赖外部检索系统 |
| 可视化 | 原生支持 | 依赖外部系统 |
| 维护成本 | 低 | 很高 |
| 写入性能 | 高 | 较高 |
| 查询性能 | 简单和复杂查询都很快 | 复杂查询较慢 |

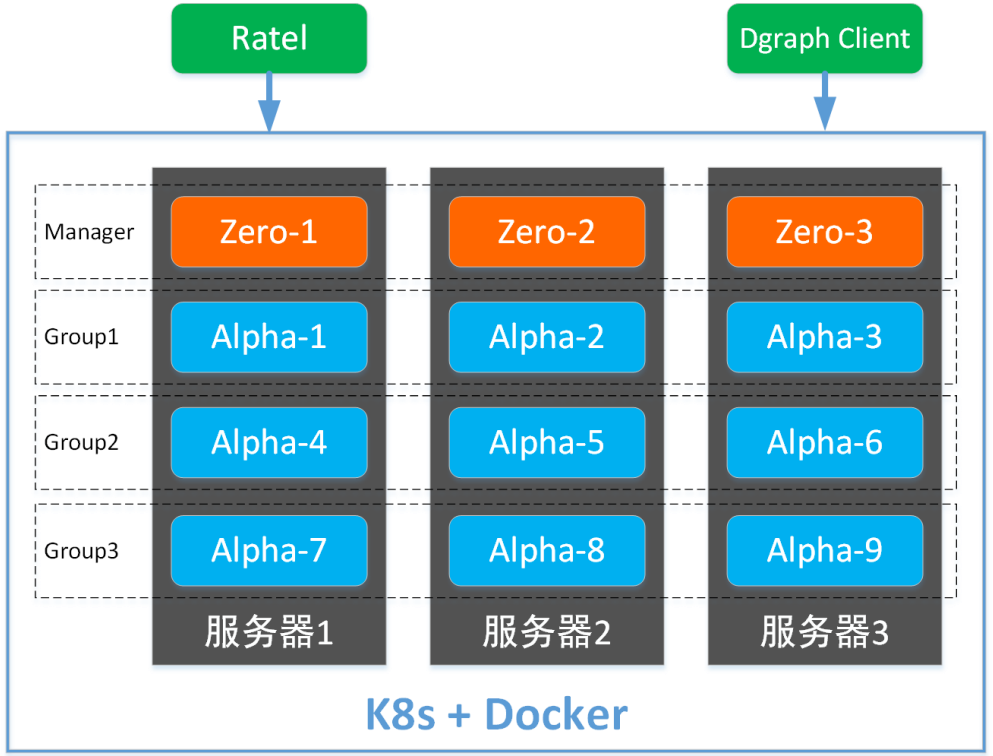
# 分享提纲

- 贝壳图数据库应用场景
- 图数据库技术选型
- 图数据库平台建设
- 原理&优化&不足

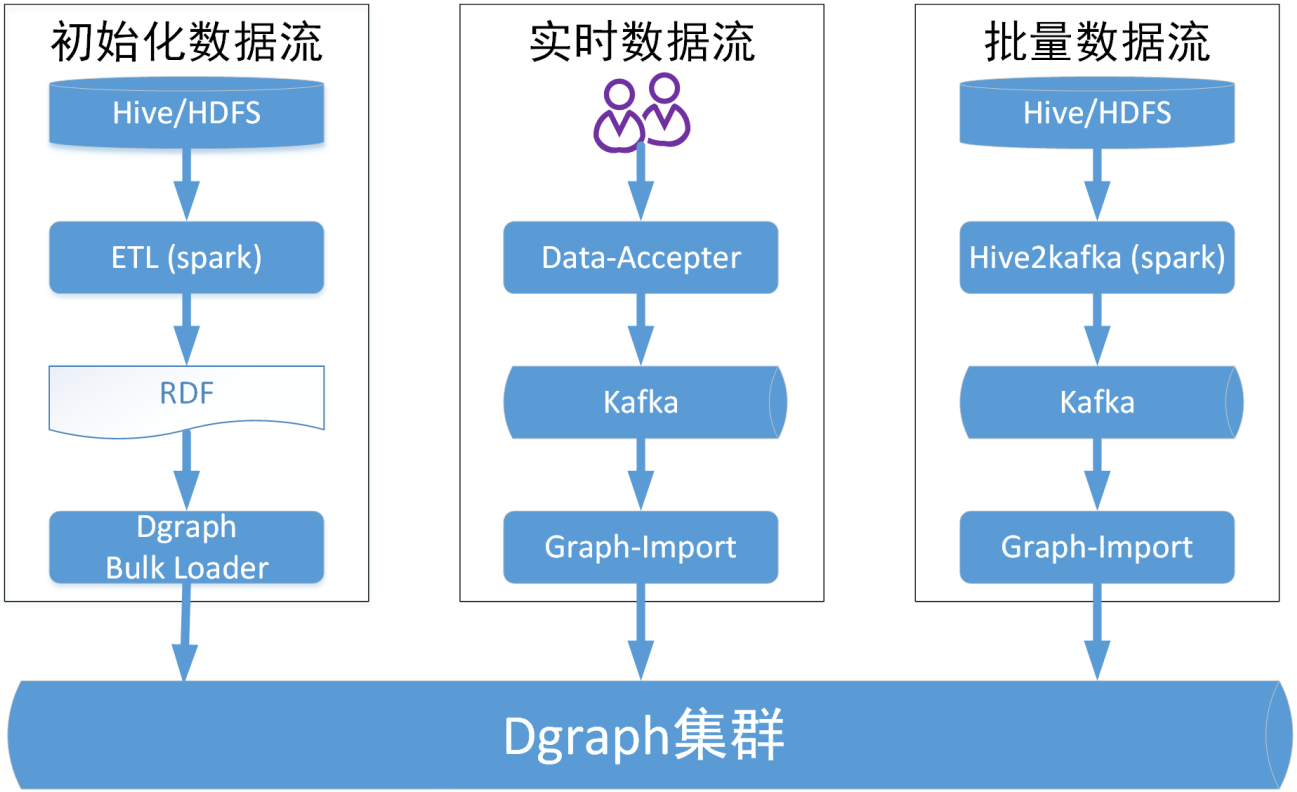
# 图数据库平台建设——集群搭建

dgraph zero --replicas 3

dgraph alpha --zero localhost:5080

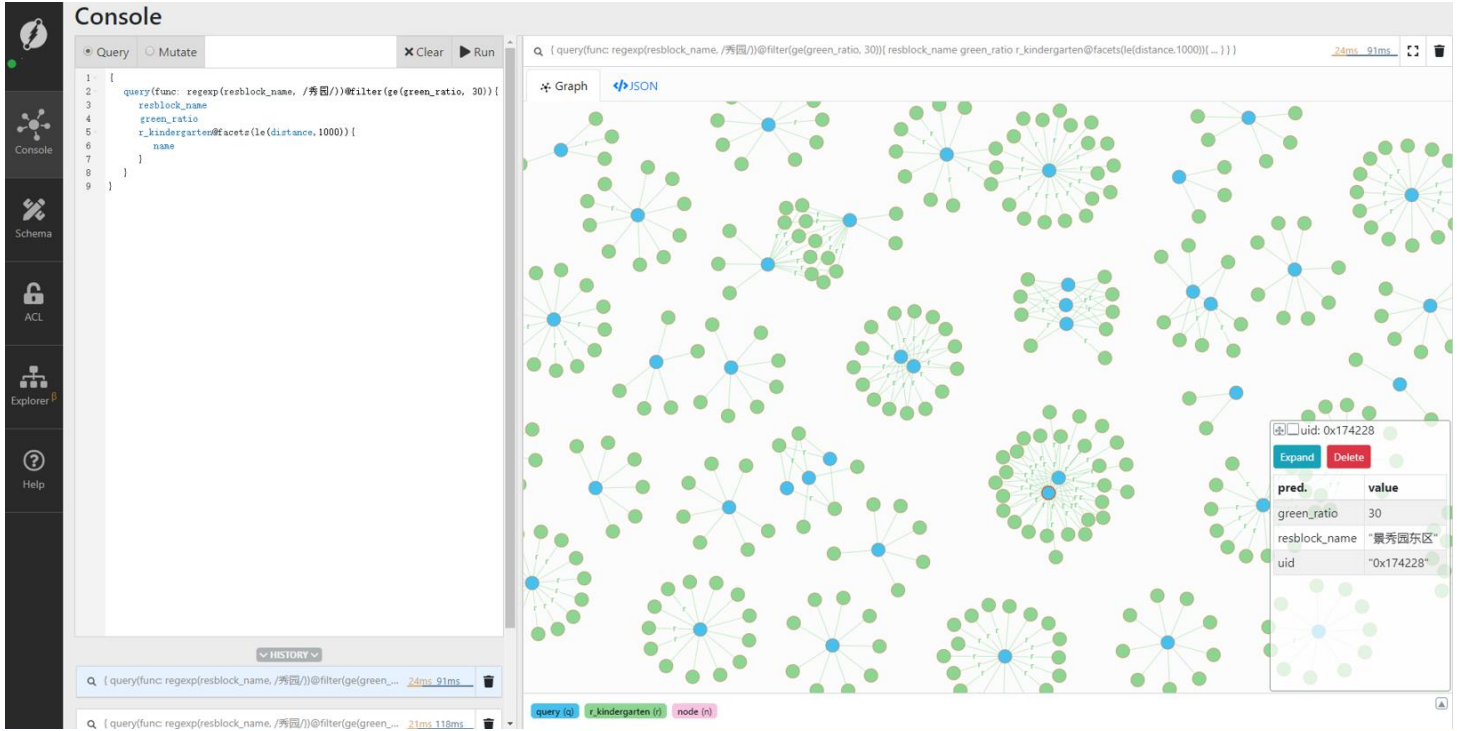


# 图数据库平台建设——数据写入



# 图数据库平台建设——数据查询

查询名字包含“秀园”，绿化率大于30%的小区附近1km的幼儿园



# 图数据库平台建设——GraphSQL

查询名字包含“秀园”，绿化率大于30%的小区附近1km的幼儿园

**GraphQL**

```
{
  query(func: regexp(resblock_name, /秀园/))@filter(ge(green_ratio, 30)){
    resblock_name
    green_ratio
    r_kindergarten@facets(le(distance,1000)){
      name
    }
  }
}
```

**Gremlin**

```
g.V().has('name', textContains('秀园')).has('green_ratio', gt(30)).as('resblock').outE('kindergarten')
     .has('distance', lt(1000)).as('kindergarten').select('resblock','kindergarten')
```

**Graph SQL**

```
SELECT resblock.resblock_name, resblock.green_ratio, kindergarten.name
FROM resblock-r_kindergarten->kindergarten
WHERE regexp(resblock.resblock_name,'秀园') AND resblock.green_ratio >= 30 AND r_kindergarten.distance <= 1000;
```

# 图数据库平台建设——GraphSQL



```
SELECT
    nodeLabel.property
    nodeLabel.*
    edgeType.property
    nodeLabel.property as alias
    count(nodeLabel)
    avg(nodeLabel.property)
    shortest(numpaths: 1, depth: 6)
    ndegree(depth: 2)
FROM
    (nodeName:nodeLabel)-[edgeAlias:edgeType]->(nodeName:nodeLabel)
    (nodeName)-[:edgeType]->(nodeName)
WHERE
    nodeLabel.property = value AND
    edgeType.property = value AND
    allofterms(nodeLabel.property, 'string')
    nodeLabel.property/edgeLabel IS NOT NULL AND
```

```
GROUP BY
    nodeLabel.property
    edgeType
Having
    avg/min/max/sum(nodeLabel.property) > value
ORDER BY
    nodeLabel.property DESC
    edgeType.property ASC
    avg/min/max/sum(nodeLabel.property)
LIMIT
    nodeLabel/edgeLabel(first: N, offset: M)
```

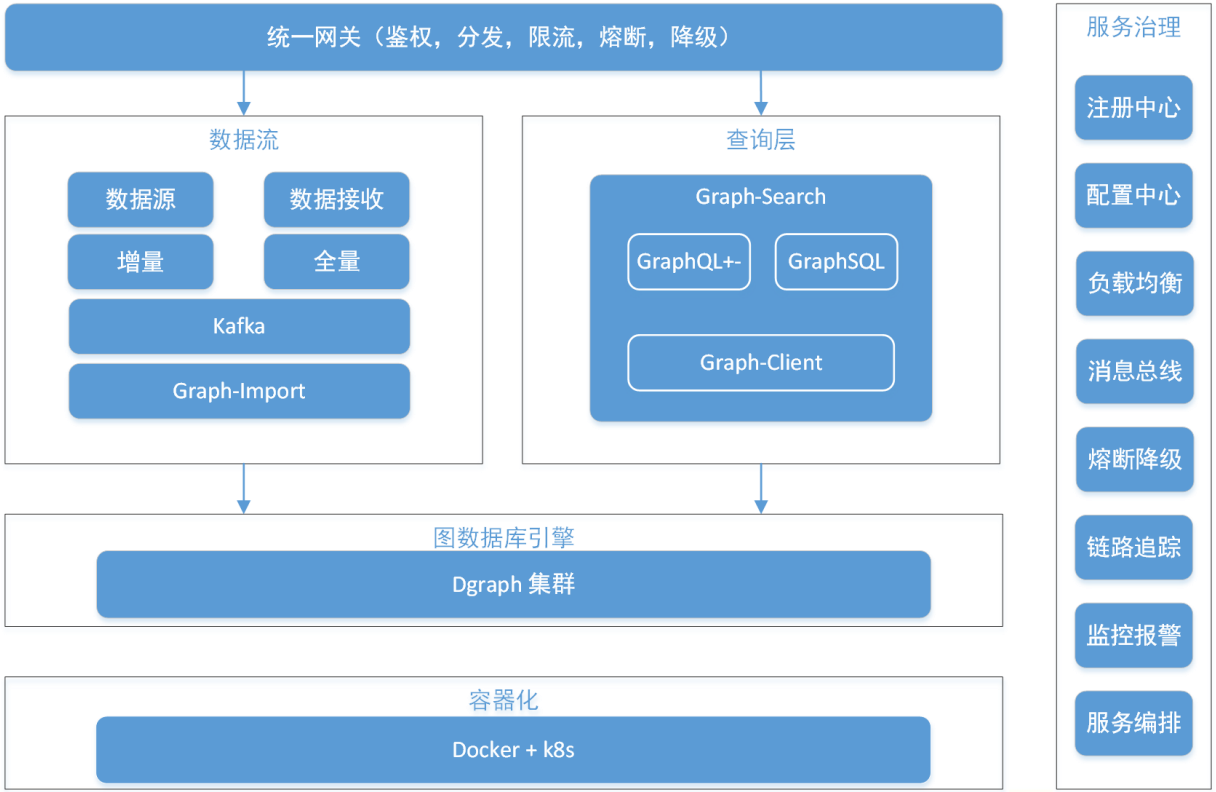
# 图数据库平台建设——GraphSQL



```
→  ~ curl -s -X POST "http://t.gw.graph.search.ke.com/api/graph/search/6014400001/" -H 'Content-Type: text/plain' -H 'token: aPy2RQ8nvb' -d "
SELECT resblock.resblock_name, resblock.green_ratio, kindergarten.name
FROM resblock-r_kindergarten->kindergarten
WHERE regexp(resblock.resblock_name,'秀园') AND resblock.green_ratio >= 30 AND r_kindergarten.distance <= 1000;" | jq
{
  "code": 0,
  "msg": "success",
  "data": {
    "query": [
      {
        "green_ratio": 45,
        "r_kindergarten": [
          {
            "name": "精英传奇华幼儿园"
          },
          {
            "name": "天津市东丽区军宏幼儿园"
          },
          {
            "name": "军丽幼儿园"
          },
          {
            "name": "乐童幼儿园"
          },
          {
            "name": "世纪之星双语托幼点"
          },
          {
            "name": "广福幼儿园"
          },
          {
            "name": "希望之星艺术托幼点"
          }
        ],
        "resblock_name": "军秀园"
      },
```

https://github.com/LianjiaTech/dgraph-sql

# 图数据库平台建设——GraphSQL



# 分享提纲

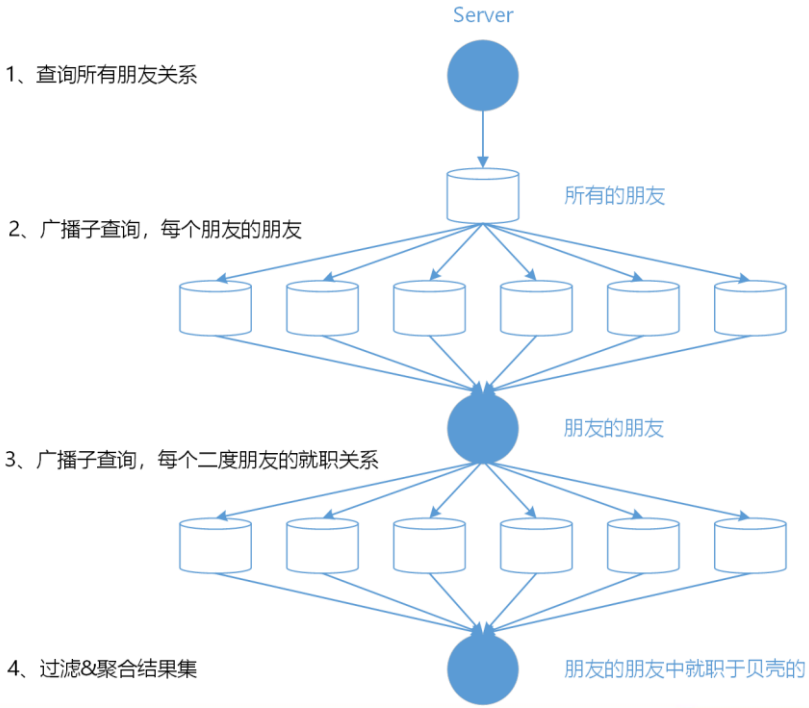
- 图数据库在贝壳的应用场景
- 图数据库技术选型
- 图数据库平台建设
- 原理&优化&不足

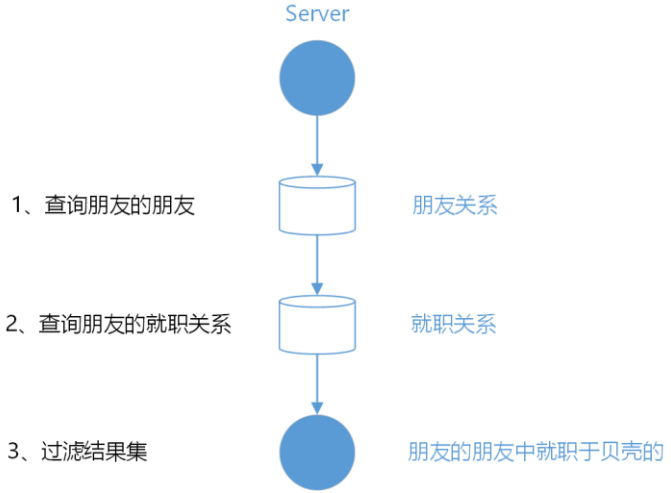
# 原理&优化——Dgraph原理

- 存储引擎
  - Badger：一个高效和持久化的，基于 LSM的键值数据库，纯Go语言编写
  - 随机读比RocksDB快3.5倍
- 存储结构
  - (Predicate, Subject) --> [sorted list of ValueId]
  - (friend, me) --> [person1,person2,person3,person4,person5]
- 数据分片
  - 根据谓词分片，相同谓词的数据按序存储在同一个节点，减少RPC
  - 定期数据均衡（rebalance_interval）
  - group根据replicas和alpha启动顺序确定
- 高可用
  - 每个group至少3个alpha，互为副本，raft协议保证强一致性
  - write-ahead logs，预写日志

# 原理&优化——Dgraph原理

例：查询我所有朋友的朋友中就职于贝壳的人。

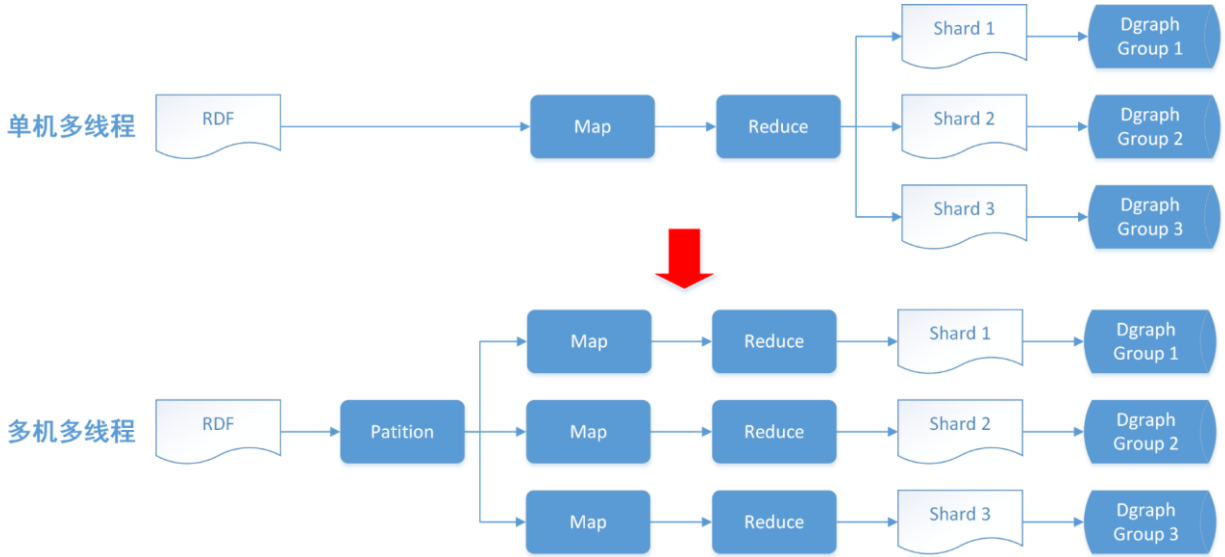



Dgraph：避免广播，一次网络调用
执行一次连接，低延迟，可预测

# 原理&优化——Dgraph优化

- Bulkloader导入优化，解决内存溢出问题，分布式导入改造
  - 行业图谱500亿三元组导入时间：48h -> 15h 提升 **3** 倍（9物理机）
- 增加数据均衡开关，业务高峰期禁止均衡，避免影响实时写入



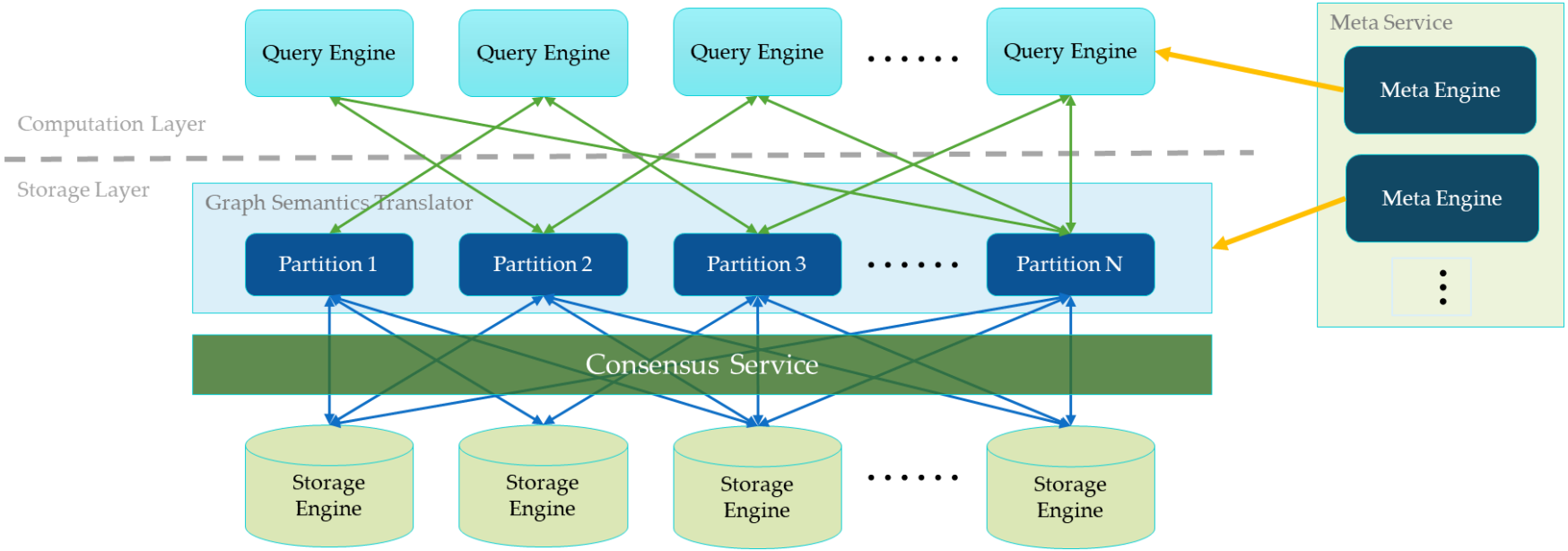
# 原理&优化&不足——Dgraph不足

- 不支持多重边
  - 任意一对顶点，相同标签类型的边只允许存在一条
- 一个集群只支持一个图
  - 企业版支持多图
- 大数据生态兼容不够
  - Spark写入容易overload
- 容易出现超级“边”
- 不是很成熟

# 分享提纲

- 贝壳图数据库应用场景
- 图数据库技术选型
- 图数据库平台建设
- 原理&优化&不足
- 新的选型

# Nebula Graph



# Nebula Graph VS Dgraph

- 测试场景：带过滤条件多度查询，结果集在1000以内，返回部分属性，压测最大QPS
- 测试数据：小区子图，200w点，8000W边，1亿RDF
- 测试机器：3台物理机，48核，128G内存，SSD硬盘

| 查询 | Dgraph<br>最大QPS/avg/p99 | Nebula Graph<br>最大QPS/avg/p99 |
|---|---|---|
| 查询指定节点7个属性 | 10475/18ms/71ms | 99053/6ms/11ms |
| 查询一度节点3个属性 | 5995/32ms/145ms | 5403/36ms/78ms |
| 查询二度节点3个属性 | 1899/48ms/344ms | 2526/39ms/156ms |
| 查询三度节点3个属性 | 1315/51ms/368ms | 698/56ms/640ms |
| 查询四度节点3个属性 | 1134/60ms/529ms | 655/60ms/764ms |

# Nebula Graph VS Dgraph

| 特性 | Dgraph | Nebula Graph |
| --- | --- | --- |
| 架构 | 存储计算一体 | 存储计算分离 |
| 副本 | 强一致性 | 强一致性 |
| 语言 | GraphQL+- | nGQL |
| 多重边 | 不支持 | 支持 |
| 多图空间 | 不支持 | 支持 |
| 分片方式 | 按边分片 | 按点分片 |
| 分布式事务 | 支持 | 不支持 |
| 写入性能 | 较高 | 很高 |
| 查询性能 | 3度以上更快 | 3度以内占优 |

# THANKS