# TuGraph-DB 5.0：面向AI时代的图存储能力增强

王志勇
蚂蚁集团-图计算技术专家

蚂蚁集团 ANT GROUP | TuGraph™

# 目 录



蚂蚁集团
ANT GROUP

# TuGraph-DB介绍

**TuGraph**

**简介**

- 蚂蚁图计算研发并开源的一款图数据库
- https://github.com/TuGraph-family/tugraph-db
- Apache 2.0 License
- 当前版本 4.5.0

**特点**

- C++编写，高性能
- 单机形态，支持HA
- 属性图模型
- OpenCypher
- 强Schema
- 兼容neo4j客户端
- 内置20+种图算法
- 存储过程

# TuGraph 5.0： GraphRAG

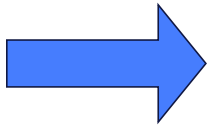
## Schema-Free + 图搜索 + 向量检索 + 全文检索

## 单系统一体化混合检索

# Schema-Free 特性简介

**4.x**

- 强 schema
- 预先定义点边结构
- 增加删除字段走DDL语句，重操作
- 点只能有一个标签
- 类似于SQL数据库
- 存储引擎LMDB



**5.0**

- 弱 schema
- 不需要预先定义点边结构
- 随时增加删除字段，轻操作
- 点可以有多个标签
- 类似于No-SQL数据库
- 存储引擎RocksDB

# Schema-Free 存储

**4.x**

LMDB

- mmap
- 纯读性能卓越
- 串行写事务
- b-tree
- 设计简单

**5.0**

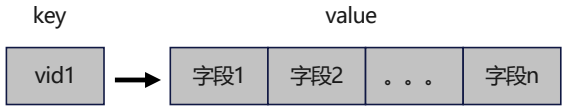
RocksDB

- 写性能优秀
- 读性能不差
- 并发写事务
- lsm-tree
- 设计复杂

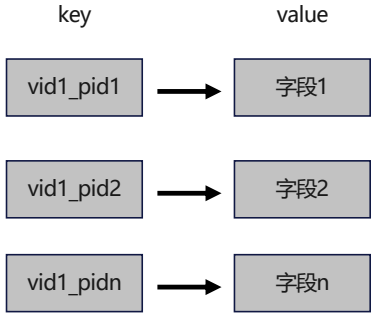
**future**

原生图存储

行存，字段紧密排列




字段散列存储



# Schema-Free 用户侧效果

4.x 版本 强 schema 使用

```
#创建点类型
CALL db.createVertexLabel('Person', 'id', 'id' ,'INT32', false, 'name', 'STRING', false)
#创建边类型
CALL db.createEdgeLabel('is_friend','[['Person', 'Person']])

#插入点边
CREATE (n1:Person {name:'jack',id:1}), (n2:Person {name:'lucy',id:2})
MATCH (n1:Person {id:1}), (n2:Person {id:2}) CREATE (n1)-[r:is_friend]->(n2)

#添加字段
CALL db.alterLabelAddFields('vertex', 'Person', ['age', int64, 0, false])
#删除字段
CALL db.alterLabelDelFields('vertex', 'Person', ['age'])
```

# Schema-Free 用户侧效果

5.x 版本 弱 schema 使用

```
#插入点边
CREATE (n1:Person {name:'jack',id:1}), (n2:Person {name:'lucy',id:2})
MATCH  (n1:Person {id:1}), (n2:Person {id:2}) CREATE (n1)-[r:is_friend]->(n2)

#添加字段
MATCH(n:Person {id:1}) SET n.age = 10
#删除字段
MATCH(n:Person {id:1}) REMOVE n.age

#多标签
create (n:Person:Student {name:'bob', id:2})
```

弱 schema 更契合cypher的语义

# 向量检索

技术选型

## VSAG

### 简介

- 用于相似性检索的开源向量索引库
- 蚂蚁向量数据库团队研发
- https://github.com/Alipay/vsag
- OceanBase最新发布向量检索基于VSAG

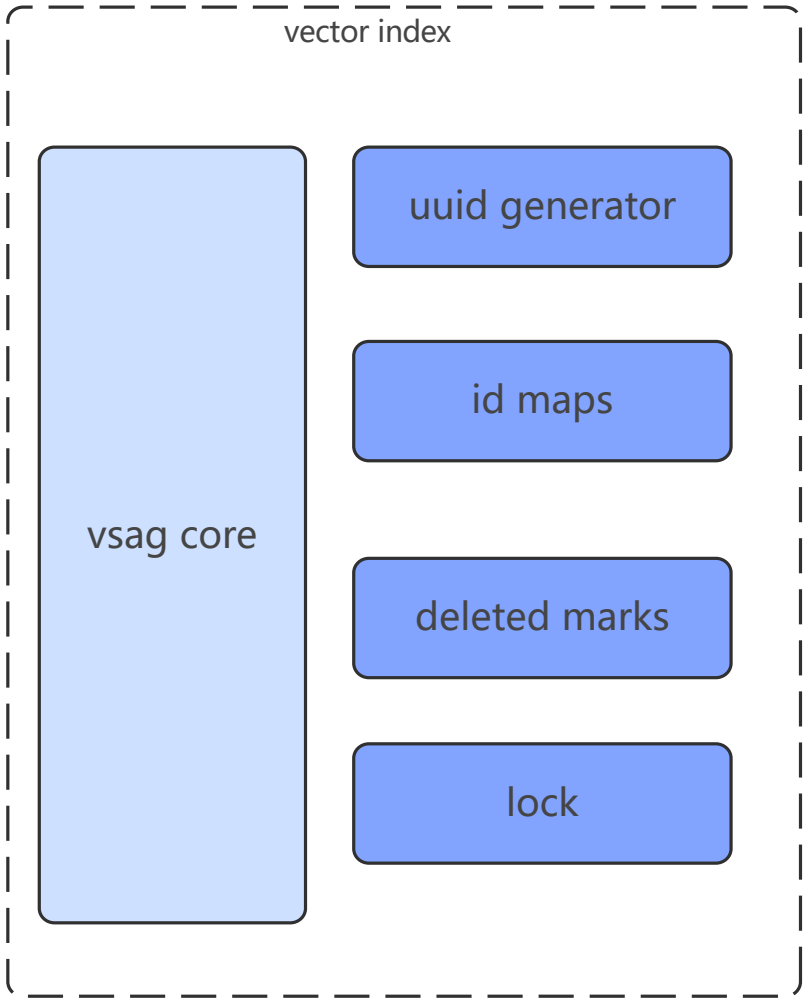
### 特点

- C++编写
- 高召回率，低内存使用
- 高性能：ANN-Benchmarks测试排名第一
- 支持混合搜索：topk搜索，range搜索

# 向量检索集成

VSAG自身局限性

- 只能Add不能Delete




**增加**

1.write lock
2.分配唯一内部id
3.业务id和内部id映射，加入id maps
4.vasg add

**删除**

1.write lock
2.业务id 通过 id maps 找到内部id
3.将内部id加入deleted marks，软标记删除
4.删除 id maps中的映射

**修改**

1.write lock
2.删除流程
3.新增流程

**查询**

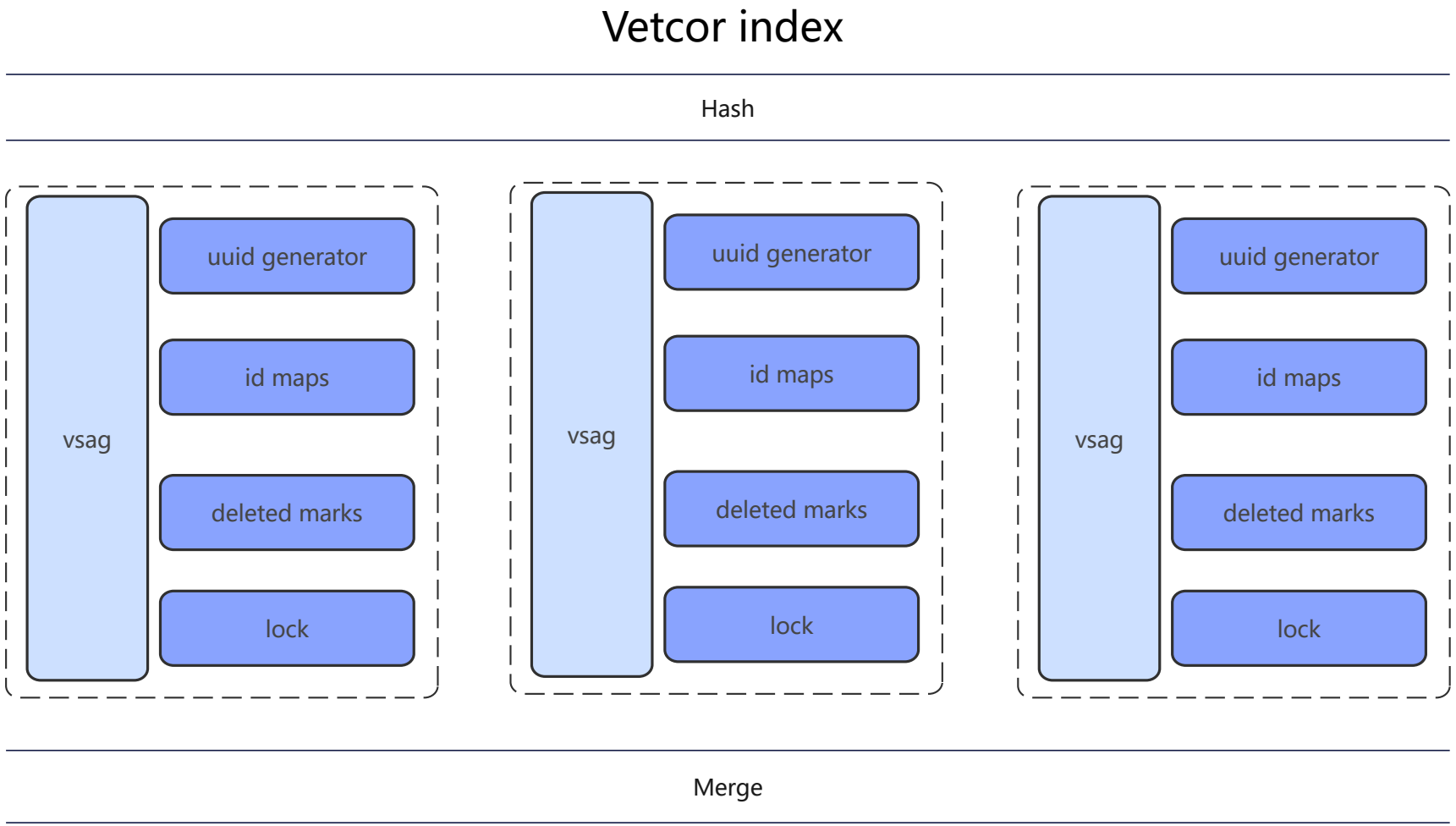
1.read lock
2.vasg search
3.deleted marks 过滤
4.从 id maps反查出业务id

# 向量检索集成

## VSAG自身局限性

- 全内存
- 只能全量数据持久化

- 写入：hash分片写入
- 查询：全部查询，merge 结果
- 启动：并行内存构建



# 向量检索-用户侧使用效果

向量索引 + Cypher 深度融合

```
#为Person类型点上的embedding字段创建向量索引，取名vector_index，维度是4，其他向量参数默认。
CALL db.index.vector.createNodeIndex('vector_index','Person', 'embedding',{dimension:4});

#插入几条点边数据
CREATE (n1:Person {id:1, age:10, embedding: toFloat32List([1.0,1.0,1.0,1.0])})
CREATE (n2:Person {id:2, age:20, embedding: toFloat32List([2.0,2.0,2.0,2.0])})
CREATE (n3:Person {id:3, age:30, embedding: toFloat32List([3.0,3.0,3.0,3.0])})
CREATE (n1)-[r:like]->(n2),
       (n2)-[r:like]->(n3),
       (n3)-[r:like]->(n1);

# 搜索与[1.0,2.0,3.0,4.0]相似的节点，取top2，然后过滤掉age<20的，最后再查找这些点的一跳邻居是谁。
CALL db.index.vector.knnSearchNodes("vector_index", [1.0,2.0,3.0,4.0], {top_k:2})
YIELD node where node.age > 20 with node as p
match(p)-[r]-(m) return m;
```

# 全文检索

技术选型

- Apache lucene
- Clucene
- LucenePlusPlus
- Tantivy

## Tantivy

### 简介

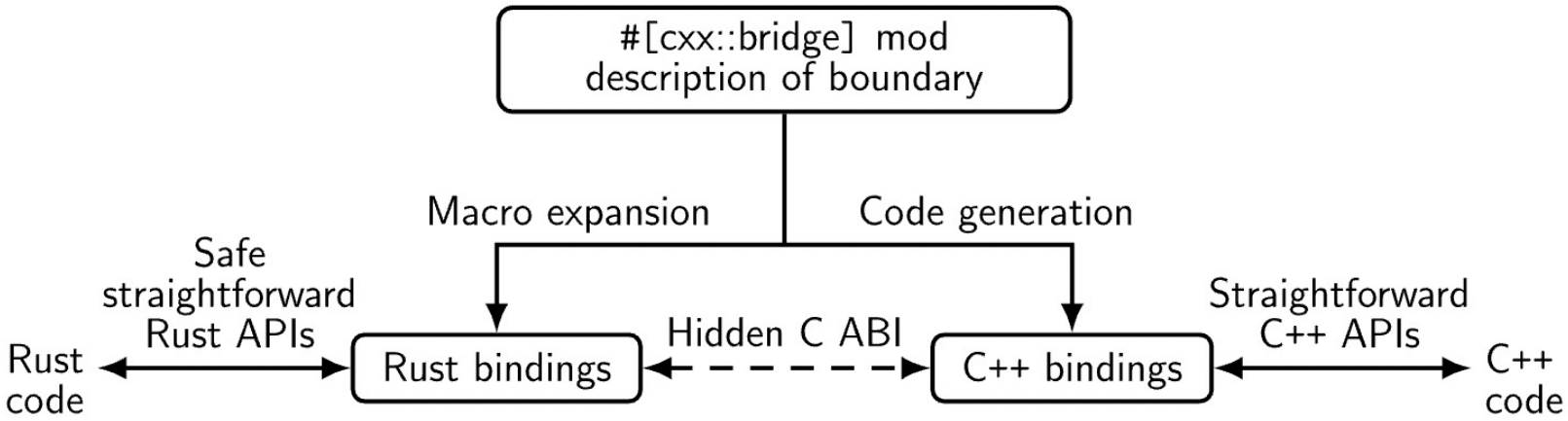
- 开源全文检索引擎库
- 社区活跃 12K star
- https://github.com/quickwit-oss/tantivy

### 特点

- Rust 编写
- 对标 Apache Lucene，后起新秀
- 性能优秀
- 完备度高，功能丰富

# 全文检索集成

- C++调用Rust
  - Rust FFI：Foreign Function Interface
    - FFI允许Rust调用其他语言也允许其他语言调用Rust
  - https://github.com/dtolnay/cxx Rust 和 CPP之间安全互操作库
- 支持点的字符串属性字段建全文索引
- 多类型组合一起加索引



# 全文检索-用户侧效果

全文检索 + Cypher 深度融合

```
#创建全文索引，指定名字是namesAndTeams
CALL db.index.fulltext.createNodeIndex('namesAndTeams', ['Employee', 'Manager'], ['name','team']);

#写入几条点边数据
CREATE (nilsE:Employee {name: 'Nils-Erik Karlsson', team: 'Kernel'})
CREATE (lisa:Manager {name: 'Lisa Danielsson'})
CREATE(nils:Employee {name: 'Nils Johansson',  team:'Operations'})
CREATE (maya:Employee {name: 'Maya Tanaka', team:'Operations'})
CREATE (lisa)-[:REVIEWED {message: 'Nils-Erik is reportedly difficult to work with.'}]->(nilsE),
              (maya)-[:EMAILED {message:'I have booked a team meeting tomorrow.'}]->(nils);

#查找name中含有单词nils的节点，然后过滤出team是kernal的，最后查找这些点的一跳邻居是谁
CALL db.index.fulltext.queryNodes('namesAndTeams', 'Nils', 10)
YIELD node where node.team = 'Kernel' with node
MATCH(node)-[r]-(m) return m;
```

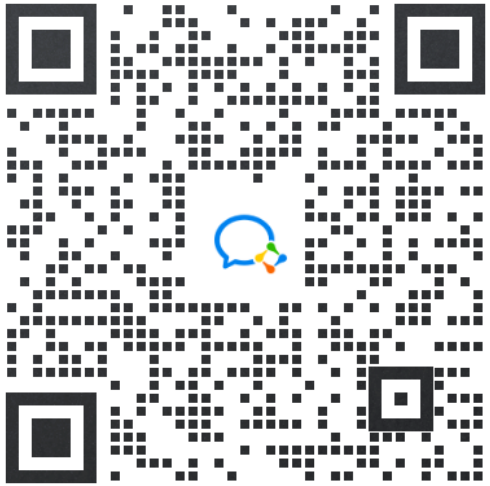
# 当前不足以及未来规划

- https://github.com/TuGraph-family/tugraph-db v5.x分支
- 前端Web适配
- HA
- 向量索引持久化
- 多引擎之间事务一致性
- 边属性索引支持
- 更多 OpenCypher 语法支持

# 联系我们

公众号

社区群



其他

tugraph@service.alipay.com

400-903-0809

# 谢谢大家

蚂蚁集团 ANT GROUP | TuGraph™