悦数图数据库手册

v3.5.1

悦数图数据库

Copyright © Yueshu Graph Technology - All rights reserved.

Table of contents

1. 欢迎阅读悦数图数据库 3.5.1 文档 6
1.1 快速开始 6
1.2 最新发布 6
1.3 其他资料 6
1.4 图例说明 6
1.5 修改文档中的错误 7
2. 简介 8
2.1 图 8
2.2 图数据库的市场概况 27
2.3 相关技术 38
2.4 什么是悦数图数据库 51
2.5 数据模型 54
2.6 路径 56
2.7 点 VID 58
2.8 服务架构 60
3. 悦数 License 76
3.1 悦数 License 76
3.2 使用 License Manager 78
4. 快速入门 83
4.1 从云开始 83
4.2 本地部署 98
4.3 nGQL 命令汇总 116
5. nGQL 指南 136
5.1 nGQL 概述 136
5.2 数据类型 151
5.3 变量和复合查询 170
5.4 运算符 175
5.5 函数和表达式 189
5.6 通用查询语句 232
5.7 子句和选项 270
5.8 图空间语句 299
5.9 Tag 语句 309
5.10 Edge type 语句 317
5.11 点语句 323
5.12 边语句 330

Copyright © Yueshu Graph Technology - All rights reserved.

5.13 原生索引 337
5.14 全文索引 348
5.15 子图和路径 357
5.16 查询调优与终止 364
5.17 作业管理 370
6. 安装部署 374
6.1 准备安装和运行悦数图数据库的环境 374
6.2 单机安装 378
6.3 使用 RPM/DEB 包部署悦数图数据库多机集群 381
6.4 使用生态工具安装悦数图数据库 388
6.5 管理悦数图数据库服务 389
6.6 连接悦数图数据库服务 392
6.7 管理 Storage 主机 394
6.8 升级悦数图数据库 3.x 至 3.5.1 395
6.9 卸载悦数图数据库 398
7. 配置与日志 400
7.1 配置 400
7.2 日志 423
8. 监控 428
8.1 查询悦数图数据库监控指标 428
8.2 RocksDB 统计数据 437
8.3 悦数黑匣子 438
9. 数据安全 443
9.1 验证和授权 443
9.2 SSL 加密 455
10. 备份与恢复 457
10.1 悦数备份&恢复 457
10.2 管理快照 468
11. 同步与迁移 470
11.1 负载均衡 470
11.2 集群间数据同步 471
12. 导入与导出 482
12.1 数据导入导出概述 482
12.2 NebulaGraph Importer 483
12.3 NebulaGraph Exchange 491
12.4 NebulaGraph Spark Connector 597
12.5 NebulaGraph Flink Connector 604

Copyright © Yueshu Graph Technology - All rights reserved.

13. 最佳实践 605
13.1 Compaction 605
13.2 Storage 负载均衡 607
13.3 图建模设计 610
13.4 系统设计建议 614
13.5 执行计划 615
13.6 超级顶点（稠密点）处理 616
13.7 实践案例 618
14. 客户端 620
14.1 客户端介绍 620
14.2 NebulaGraph Console 621
14.3 NebulaGraph CPP 625
14.4 NebulaGraph Java 627
14.5 NebulaGraph Python 629
14.6 NebulaGraph Go 630
15. 悦数图数据库（公有云） 631
15.1 什么是悦数图数据库（公有云） 631
15.2 悦数图数据库（阿里云版） 632
16. 运维监控 639
16.1 什么是悦数运维监控 639
16.2 部署悦数运维监控 641
16.3 连接悦数运维监控 647
16.4 创建及导入集群 649
16.5 集群管理 658
16.6 权限管理 696
16.7 任务中心 699
16.8 许可证管理 700
16.9 平台设置 701
16.10 监控指标说明 707
16.11 常见问题 FAQ 716
17. 图探索 718
17.1 什么是悦数图探索 718
17.2 安装与登录 720
17.3 页面概览 729
17.4 数据库管理 732
17.5 图探索 745
17.6 可视化查询 757
17.7 画布功能 762

Copyright © Yueshu Graph Technology - All rights reserved.

17.8 工作流 771
17.9 内联框架 800
17.10 系统设置 802
17.11 基本操作和快捷键 803
17.12 常见问题 FAQ 804
18. 图计算 806
18.1 算法简介 806
18.2 NebulaGraph Algorithm 820
18.3 悦数图计算 825
18.4 悦数图探索 Workflow 832
19. 自动化部署运维 833
19.1 什么是 NebulaGraph Operator 833
19.2 使用流程 835
19.3 部署 NebulaGraph Operator 836
19.4 部署悦数图数据库 841
19.5 通过 Nebular Operator 连接悦数图数据库 853
19.6 配置悦数图数据库 858
19.7 使用 NebulaGraph Operator 备份和恢复数据 870
19.8 故障自愈 874
19.9 常见问题 875
20. NebulaGraph Bench 877
20.1 适用场景 877
20.2 更新说明 877
21. 常见问题 FAQ 878
21.1 关于历史兼容性 878
21.2 关于执行报错 878
21.3 关于设计与功能 880
21.4 关于运维 882
21.5 关于连接 885
21.6 关于扩容、缩容 886
22. 附录 887
22.1 Release Note 887
22.2 悦数图数据库学习路径 893
22.3 产品端口全集 899
22.4 生态工具概览 902
22.5 错误码 906

Copyright © Yueshu Graph Technology - All rights reserved.

1. 欢迎阅读悦数图数据库 3.5.1 文档

Note

本文档更新时间2023-10-7，GitHub commit b33e7a16。该版本主色系为"金盏花"，色号为 #FFCC83。

Caution

该版本文档仅包括悦数图数据库 3.5.1 企业版内容和对应版本的周边工具内容。详情查看版本发布说明。

1.1 快速开始

- 快速开始
- 部署要求
- nGQL 命令汇总
- FAQ
- 生态工具
- Academy 课程

1.2 最新发布

- 悦数图数据库 3.5.1
- 悦数运维监控
- 悦数图探索

1.3 其他资料

- 学习路径
- 主页

1.4 图例说明

Note

额外的信息或者操作相关的提醒等。

Caution

需要严格遵守的注意事项。不遵守 caution 可能导致系统故障、数据丢失、安全问题等。

Danger

会引发危险的事项。不遵守 danger 必定会导致系统故障、数据丢失、安全问题等。

Copyright © Yueshu Graph Technology - All rights reserved.

Performance

性能调优时需要注意的事项。

Faq

常见问题。

Compatibility

nGQL 与 openCypher 的兼容性或 nGQL 当前版本与历史版本的兼容性。

Enterpriseonly

描述社区版和企业版的差异。

1.5 修改文档中的错误

悦数图数据库文档以 Markdown 语言编写。单击文档标题右上侧的铅笔图标即可提交修改建议。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2. 简介

- 第一课：图的概念（03 分 45 秒）

- 第二课：图的结构（02 分 24 秒）

2.1 图

当前，从计算机行业巨头（例如 Amazon 和 Facebook）到小型研究团队，都投入了大量的资源探索图数据库在解决各种数据关系问题上的潜力。当然你也可以选择像他们这样进行尝试，现在可供选择的数据库有很多。那么图数据库究竟是什么？它可以做些什么？作为一类数据库，它在数据库领域里处于什么位置呢？要回答这些问题，我们首先得了解图。

图是计算机科学研究的主要领域之一。图能够高效地解决目前存在的诸多问题。本章将从图说起，继而说明图数据库的优点及其在现代应用程序开发中的巨大潜力，然后介绍分布式图数据库的区别和几种其他类型的数据库。

2.1.1 图、图片与图论

图无处不在。当听到图这个词时，很多人都会想到条形图或折线图，因为有时候我们确实会把它们称作图。从传统意义上来说，图是用来展示两个或多个数据系统之间的联系的。最简单的例子如下图，下图展示了悦数图数据库 GitHub 仓库星星数量随时间推移的变化。

这是相对比较简单的一种图，横轴为时间，纵轴为星星数量。可以看到，星星数量是随着时间推移而上升的。这种类型的图通常称为折线图。折线图可以显示随时间（根据常用比例设置）而变化的连续数据。此处我们只给出了折线图的例子。当然图的形式有多种，比如饼图、条形图等。

还有一种“图”在日常口语中会更多的被提及，例如，“图像识别”，“美图秀”，“修图”等。这些“图”都是指图片。

但是——总会有但是——我们在本书中讨论的图是另外一个概念——“图论”中的图。

图论始于 18 世纪初期的柯尼斯堡七桥问题。柯尼斯堡当时是普鲁士的城市，普雷格尔河穿过柯尼斯堡，不仅把柯尼斯堡分成了两部分，而且还在河中间形成了两个小岛。这就将整个城市分割成了四个区域，各区域由七座桥连接。在所有的桥都只能走一遍的前提下，如何能把这个地方所有的桥都走一遍呢？

Copyright © Yueshu Graph Technology - All rights reserved.

有兴趣的话可以试试寻找这个小游戏的答案[1]。下图为柯尼斯堡七座桥的简化图。

Copyright © Yueshu Graph Technology - All rights reserved.

大数学家欧拉于 1735年提出，并没有方法能圆满解决这个问题，并在第二年发表论文，证明符合条件的走法并不存在。这篇论文在圣彼得堡科学院发表，成为图论史上第一篇重要文献。

在论文中，欧拉把实际的抽象问题简化为平面上的点与边组合，将城市的四个区域抽象成点，将连接城市的七座桥抽象成连接点的边。

图中四个点代表柯尼斯堡的四个区域，点之间的线代表连接四个区域的七座桥。从图中不难看出，偶数座桥连接的区域可以轻松通过，因为来去可以选择不同的路线。奇数座桥连接的区域只能作为起点或者终点，因为同样的路线只能走一次。和节点相关联的边的条数称为节点度。现在可以证明，只有两个节点有奇数度，另外节点有偶数度时，也即两个区域必须有偶数座桥，剩下的区域有奇数座桥时，柯尼斯堡问题才能解决。欧拉论述了，由于柯尼斯堡七桥问题中所有区域都为奇数度，它无法实现符合题意的遍历。

欧拉发表的相关论文被认为是图论领域的第一篇文章，因此普遍认为欧拉是图论的创始人。

Copyright © Yueshu Graph Technology - All rights reserved.

图论中的相关概念，我们将在后面的学习中提到。简单来说，图论就是研究图的学问。图是基本研究对象，用于表示实体与实体之间的关系。

在数学的分支图论中，图（Graph）是基本研究对象。在中文中，强调为“拓扑图”、“网络图”等。这一名词最早由西尔维斯特在 1878 年提出。他是著名的英国数学家、牛津大学几何教授，用图来表示数学和化学分子结构之间的关系。

一张图由一些小圆点（称为顶点或节点，即 Vertex）和连接这些圆点的直线或曲线（称为边，即 Edge）组成。

2.1.2 属性图

从数学角度来说，图论是研究建模对象之间关系结构的学科。但是从工业界使用的角度，通常会对基础的图模型进行扩展，称为属性图模型。属性图通常由以下几部分组成：

- 节点，即对象或实体。在本书中，通常简称为点（Vertex）。
- 节点之间的关系，在本书中，通常简称为边（Edge）。通常边是有方向或者无方向的，以表示两个实体之间有持续的关系。

- 此外，在节点和边上，还可以有属性（properties）。

在现实生活中，有很多属性图的例子。

在网络理论中，图可以用来做可视化的社会网络分析，研究社会实体之间的关系结构。例如企查查或者 BOSS 直聘这类的公司，用图来建模商业股权关系网络。这个网络中，点通常是一个自然人或者是一家企业，边通常是某自然人与某企业之间的股权关系。点上的属性可以是自然人姓名、年龄、身份证号等。边上的属性可以是投资金额、投资时间、董监高等职位关系。

Copyright © Yueshu Graph Technology - All rights reserved.

在一个股票市场里面，点可以是一家上市公司，边可以是上市公司之间的相关性。点的属性可以为股票代码、简称、市值、板块等；边的属性可以为股价的时间序列相关性系数[2]。

Copyright © Yueshu Graph Technology - All rights reserved.

图关系还可以是类似《权力的游戏》这样电视剧中的人物关系网[3]：点为人物，边为人物之间的互动关系；点的属性为人物姓名、年龄、阵营等，边的属性（距离）为两个人物之间的互动次数，互动越频繁距离越近。

Copyright © Yueshu Graph Technology - All rights reserved.

图也可以用于 IT 系统内部的治理。例如，对于像微众银行这样的公司，通常有着非常庞大的数据仓库，以及相应的数仓管理工具。这些管理工具记录了数仓内 Hive 表之间通过 Job 实现的 ETL 关系[4]，这样的 ETL 关系，可以非常方便的用图的形式呈现和管理，当出现问题时也可以非常方便地追溯根源。

Copyright © Yueshu Graph Technology - All rights reserved.

图也可以用于记录一个大型 IT 系统内部错综复杂的微服务之间的调用关系[5]，运维团队用其进行服务治理。这里每个点表示一个微服务，边表示两个微服务之间的调用关系；这样，运维人员可以方便地寻找可用性低于阈值 (99.99%) 的调用链路，或者发现那些出故障会影响面特别大的微服务节点。

Copyright © Yueshu Graph Technology - All rights reserved.

服务治理
- 图谱数据
  - 将RPC服务调用关系写入图谱
  - 包含service、api、team等4类实体及5类关系
  - 点边数量在百万级别，实时写入
- 用于服务链路治理和告警优化

```
//查找API com.sankuai.ia.search.api:SearchControllerV2.search过去七天可用率低于99.99%的链路的thrift调用，最大图遍历深度为10
GO 1 TO 10 STEPS FROM
hash("com.sankuai.ia.search.api:SearchControllerV2.search") OVER call WHERE call.availability>0 AND call.availability<1000000 AND
$$.api.type=="mtthrift" YIELD call._src,call._dst
```

```
//查找所有java类型服务提供的API，并统计其会影响的上游API的数量，从高到低排序看影响次数大小（调用的可用率小于4个9）
LOOKUP ON service WHERE service.type=="java"
| GO FROM $-.VertexID OVER provider YIELD
provider._dst AS java_api_id
| GO FROM $-.java_api_id OVER call REVERSELY WHERE
call.availability>0 AND call.availability<1000000
YIELD call._src AS api_src, call._dst AS api_dst
| GROUP BY $-.api_src YIELD $-.api_src AS api_id,
count(1) AS call_cnt
| ORDER BY call_cnt DESC
| FETCH PROP ON api $-.api_id YIELD
api.appkey,api.method,$-.call_cnt
```

图也可以用于提升代码开发效率。用图存放代码之间的函数调用关系[5]，可以提升研发团队审查和测试代码的效率。在这样的图中，每个点是代码中的一个函数或者变量，每个边是函数或者变量之间的调用关系。当有新提交的代码之时，人们可以更方便的看到可能会受到影响到的其他接口，这样可以帮助测试人员更好的评估潜在的上线风险。

代码依赖分析
- 图谱数据
  - 将公司代码库中代码的依赖关系写入图谱
  - 包含method、field、class、interface等4类顶点、7类关系
  - 点边数量在千万级别，实时写入
- 用于QA精准测试
  - RD向代码仓库提交PR后，能查询出所修改代码能影响到的外部接口，并展示调用路径

```
//查找最外层method到某个method的所有无环路径
(GO 1 to 30 STEPS FROM 2946345526231222882 OVER
method_call_method REVERSELY YIELD DISTINCT
method_call_method._dst AS id
MINUS
GO 1 to 30 STEPS FROM 2946345526231222882 OVER
method_call_method REVERSELY YIELD DISTINCT
method_call_method._src as id )
| FIND NOLOOP path FROM $-.id TO
2946345526231222882 OVER method_call_method UPTO
30 STEPS
```

```
//确认两个method间是否有路径
FIND SINGLE SHORTEST PATH FROM hash("method1")
TO hash("method2") OVER method_call_method UPTO
30 STEPS
```

此外，相对于静态不发生变化的属性图，我们还可以通过增加一些时间信息，发掘出更多的使用场景。

例如，在一个银行间账户资金流向网络里面[6]，点是账户，边是账户之间的转账记录。边属性记录了转账的时间、金额等。同盾、邦盛、半云科技等公司采用图技术，可以方便地通过图的方式探索发现明显的资金挪用、“以贷还贷”、“团伙贷款”等现象。

Copyright © Yueshu Graph Technology - All rights reserved.

同样的方法也可以用于探索发现加密货币的流向。

在一个黑产账户和设备网络中[7]，其中的点可以是账户、手机设备和 WIFI 网络，边是这些账户与手机设备之间的登录关系，以及手机设备和 WIFI 网络之间的接入关系。

Copyright © Yueshu Graph Technology - All rights reserved.

这些登录记录的网络构成了黑产群体网络的团伙作案特征。360 数科[7]、快手[8]、微信[9]、知乎[10]、携程金融这些公司都通过图技术实时（毫秒级的）识别超过百万个的黑产社群。

更进一步，除了时间这个维度外，我们通过添加一些地理空间信息，还能发现属性图更多的应用场景。

例如新冠病毒的流行病学溯源[11]，点是人物，边是人与人之间的接触；点属性为人物的身份证、发病时间等信息，边属性为人物之间发生密切接触的时间和地理空间等。为卫生防疫部门快速识别高风险人群和其行为轨迹提供帮助。

Copyright © Yueshu Graph Technology - All rights reserved.

地理空间与图的结合也可以用于一些 O2O 的场景，例如基于 POI（Point-of-Interest）的实时美食推荐[12]，使得美团这类本地生活服务平台公司能在消费者在打开 APP 的时候，实时推荐出更为合适的商家。

Copyright © Yueshu Graph Technology - All rights reserved.

图还可以用于更深度的知识推理，华为、vivo、OPPO、微信、美团等公司，将图用于表征底层知识关系的数据模型。

由此可见，图可用于在物理、生物、社会和信息系统中建模许多类型的关系和过程，许多实际问题可以用图来表示。因此，图论成为运筹学、控制论、信息论、网络理论、社会科学、语言学、计算机科学等众多学科强有力的数学工具。

Copyright © Yueshu Graph Technology - All rights reserved.

2.1.3 为什么要使用图数据库

虽然关系型数据库与 XML/JSON 等半结构类型的数据库，都可以用来描述图结构的数据模型，但是，图（数据库）不仅可以描述图结构与存储数据本身，更着眼于处理数据之间的关联（拓扑）关系。具体来说，图（数据库）有这么几个优点：

- 图是一种更直观、更符合人脑思考直觉的知识表示方式。这使得我们在抽象业务问题时，可以着眼于“业务问题本身”，而不是“如何将问题描述为数据库的某种特定结构（例如表格结构）”。

Copyright © Yueshu Graph Technology - All rights reserved.

- 图更容易展现数据的特征，例如转账的路径、近邻的社区。例如，如果要分析《权力的游戏》中的人物派别关系和人物重要性，表的组织方式如下：

Id	Label	Interval	Modularity Class	betweenness	community
-4364237027732478972	Arya-Stark		4	0.017495	0
-4973959390558533367	Gendry		0	0.0	0
-7025436182975816947	Hallis-Mollen		5	0.0	0
8249703968876499989	Raymun-Darry		0	0.0	0
-9085767779232784107	Hobb		3	0.0	1
7505218728385384214	Randyll-Tarly		3	0.0	1
5114251854412197144	Viserys-Targaryen		7	0.002858	2
-2958304360028784612	Jonos-Bracken		2	0.0	3
-4359919586974142177	Lancel-Lannister		0	0.0	0
7195357273749773604	Robb-Stark		5	0.072984	0
1075531505424338989	Marq-Piper		4	0.006237	0
-3670573862518875597	Addam-Marbrand		2	0.0	0
3931316780000927284	Robert-Arryn		2	0.0	0
4826035041806538 79	Irri		7	0.000096	2
7181995877190281794	Jommo		7	0.0	2
1787258805162148944	Eddard-Stark		0	0.269604	0
-558158798715221422	Danwell-Frey		1	0.021389	4
-2434779847791595692	Mordane		4	0.001056	0
8409318129262644314	Clydas		3	0.0	1
-1738391870688186527	Karyl-Vance		2	0.010753	5
6902668443292429674	Haggo		7	0.000068	2
958616218496853 59	Mya-Stone		2	0.0	0
2025059960255100299	Porther		0	0.0	0
-1967336888960310122	Colemon		2	0.0	0
5894253429758865049	Ilyn-Payne		4	0.00032	0
-1356662602606843742	Pycelle		0	0.000321	0
382702656484230079	Gared		6	0.004328	1
1306559795724859089	Jorah-Mormont		7	0.012611	2
-96410199981447977	Cayn		0	0.000022	0

这显然不如下方图的组织方式直观：

Copyright © Yueshu Graph Technology - All rights reserved.

特别是当某些中心节点被删除:

Copyright © Yueshu Graph Technology - All rights reserved.

或者，增加一条边，可以彻底地改变整个图拓扑：

虽然只是个别数据的细微改变，图可以比表更直观地表现其中的重要而系统的信息。

Copyright © Yueshu Graph Technology - All rights reserved.

- 图查询语言是针对图结构访问设计的，可以更加直观。例如，下面是一个 LDBC 中的查询示例，要求：查找某人（Person）在社交网络上发布的帖子（Posts）；查找相应的回复（Message，回复本身还会被多次回复）;发帖时间、回帖时间都满足一定条件；根据回帖数量对结果排序。

如果使用 PostgreSQL 编写查询语句：

```
--PostgreSQL
WITH RECURSIVE post_all(psa_threadid
                      , psa_thread_creatorid, psa_messageid
                      , psa_creationdate, psa_messagetype
                       ) AS (
    SELECT m_messageid AS psa_threadid
         , m_creatorid AS psa_thread_creatorid
         , m_messageid AS psa_messageid
         , m_creationdate, 'Post'
      FROM message
     WHERE 1=1 AND m_c_replyof IS NULL -- post, not comment
       AND m_creationdate BETWEEN :startDate AND :endDate
  UNION ALL
    SELECT psa.psa_threadid AS psa_threadid
         , psa.psa_thread_creatorid AS psa_thread_creatorid
         , m_messageid, m_creationdate, 'Comment'
      FROM message p, post_all psa
     WHERE 1=1 AND p.m_c_replyof = psa.psa_messageid
     AND m_creationdate BETWEEN :startDate AND :endDate
)
SELECT p.p_personid AS "person.id"
     , p.p_firstname AS "person.firstName"
     , p.p_lastname AS "person.lastName"
     , count(DISTINCT psa.psa_threadid) AS threadCount
END) AS messageCount
     , count(DISTINCT psa.psa_messageid) AS messageCount
  FROM person p left join post_all psa on (
       1=1   AND p.p_personid = psa.psa_thread_creatorid
   AND psa_creationdate BETWEEN :startDate AND :endDate
   )
 GROUP BY p.p_personid, p.p_firstname, p.p_lastname
 ORDER BY messageCount DESC, p.p_personid
 LIMIT 100;
```

如果使用为图专门设计的图语言 Cypher 编写查询语句：

```
--Cypher
MATCH (person:Person)<-[:HAS_CREATOR]-(post:Post)<-[:REPLY_OF*0..]-(reply:Message)
WHERE post.creationDate >= $startDate AND post.creationDate <= $endDate
  AND reply.creationDate >= $startDate AND reply.creationDate <= $endDate
RETURN
  person.id, person.firstName, person.lastName, count(DISTINCT post) AS threadCount,
  count(DISTINCT reply) AS messageCount
ORDER BY
  messageCount DESC, person.id ASC
LIMIT 100
```

- 由于存储引擎和查询引擎可以针对图的结构专门设计，图的遍历（对应 SQL 中的 join）要高效得多。下图是知名产品 Neo4j 所做的一个对比[12]。

深度	关系型数据库的执行时间(s)	Neo4j的执行时间(s)	返回的记录条数
2	0.016	0.01	~2500
3	30.267	0.168	~110000
4	1543.505	1.359	~600000
5	未完成	2.132	~800000

关系数据库 vs 图数据库(多跳查询)

Copyright © Yueshu Graph Technology - All rights reserved.

- 图数据库具有广泛的适用场景。例如数据集成（知识图谱）、个性化推荐、欺诈与威胁检测、风险分析与合规、身份（与控制权）验证、IT 基础设施管理、供应链与物流、社交网络研究等。
- 根据文献[13] 的统计，使用图技术最多的领域，依次是：信息技术(IT)、学术界研究、金融、工业界实验室、政府、医疗健康、国防、制药业、零售与电子商务、交通运输、电信、保险。
- 2019 年，根据 Gartner 的问卷调研，27% 的客户（500 组）在使用图数据库，20% 有计划使用。

2.1.4 RDF

受篇幅所限，本章不讨论 RDF 数据模型。

---

1. 图片来源 https://medium.freecodecamp.org/i-dont-understand-graph-theory-1c96572a1401. ↩
2. https://nebula-graph.com.cn/posts/stock-interrelation-analysis-jgrapht-nebula-graph/ ↩
3. https://nebula-graph.com.cn/posts/game-of-thrones-relationship-networkx-gephi-nebula-graph/ ↩
4. https://nebula-graph.com.cn/posts/practicing-nebula-graph-webank/ ↩
5. https://nebula-graph.com.cn/posts/meituan-graph-database-platform-practice/ ↩↩
6. https://zhuanlan.zhihu.com/p/90635957 ↩
7. https://nebula-graph.com.cn/posts/graph-database-data-connections-insight/ ↩↩
8. https://nebula-graph.com.cn/posts/kuaishou-security-intelligence-platform-with-nebula-graph/ ↩
9. https://nebula-graph.com.cn/posts/nebula-graph-for-social-networking/ ↩
10. https://mp.weixin.qq.com/s/K2QinpR5Rplw1teHpHtf4w ↩
11. https://nebula-graph.com.cn/posts/detect-corona-virus-spreading-with-graph-database/ ↩
12. https://nebula-graph.com.cn/posts/meituan-graph-database-platform-practice/ ↩↩
13. https://arxiv.org/abs/1709.03188 ↩

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.2 图数据库的市场概况

既然已经讨论了什么是图，接下来让我们进一步认识基于图论和属性图模型发展起来的图数据库。

不同的图数据库在术语方面可能会略有不同，但是归根结底都是在讲点、边和属性。至于更多的功能，例如标签、索引、约束、TTL、长任务、存储过程和UDF等这些高级功能，在不同图数据库中，会存在明显的差异。

图数据库用图来存储数据，而图是最接近高度灵活、高性能的数据结构之一。图数据库是一种专门用于存储和检索庞大信息网的存储引擎，它能够高效地将数据存储为点和边，并允许对这些点边结构进行高性能的检索和查询。我们也可以为这些点和边添加属性。

图数据库几乎适用于存储所有领域的数据。因为在几乎所有领域中，事物之间都是由某种相关联的。图数据库支持存储实体之间的丰富关系，并且能够将这些关系完美地呈现出来，而无需像其他建模方式那样，将关系也当成实体存储。因此图数据库能够以最接近对数据直观认知的形式存储数据。

2.2.1 三方机构的统计和预测

DB-Engines 的统计

根据世界知名的数据库排名网站DB-Engines.com的统计，图数据库至2013年以来，一直是“增速最快”的数据库类别[1]。

该网站根据一些指标来统计每种类别的数据库的流行度变化趋势，这些指标包括基于Google等搜索引擎的收录和趋势情况、主要IT技术论坛和社交网站上讨论的技术话题、招聘网站的职位变化等。该网站共收录了371种数据库产品，并分为12个类别。这12个类别中，图数据库这种类别的增速远远快于其他任何的类别。

Gartner 的预测

世界顶级智库Gartner早在2013年之前[2]，就将图数据库作为主要的"商业智能与分析技术趋势"。在那个时候，Big Data正火热的如日中天，数据科学家更是炙手可热的职位。

Copyright © Yueshu Graph Technology - All rights reserved.

Figure 1. Hype Cycle for Business Intelligence and Analytics, 2013

BI = business intelligence; DBMS = database management system; SaaS = software as a service; PaaS = platform as a service

直到最近，图数据库及相关的图技术依旧是"2021年十大数据与分析趋势"[3]:

Copyright © Yueshu Graph Technology - All rights reserved.

趋势八：图技术使一切产生关联（**Graph Relates Everything**）

图技术已成为许多现代数据和分析能力的基础，能够在不同的数据资产中发现人、地点、事物、事件和位置之间的关系。数据和分析领导者依靠图技术快速回答需要在了解情况并理解多个实体之间的联系和优势的性质后才能回答的复杂业务问题。

Gartner预测，到2025年图技术在数据和分析创新中的占比将从2021年的10%上升到80%。该技术将促进整个企业机构的快速决策。

可以注意到，Gartner 的预测比较好的吻合了 DB-Engines 的统计结论。技术的进步并不是完全线性的，通常会有一段快速发展的泡沫期，然后进入一段平台期，之后由于新的技术的出现产生新一轮的泡沫期，再经历一段平台期。以此往复螺旋形的循环发展。

Copyright © Yueshu Graph Technology - All rights reserved.

对于市场规模的预测

根据 verifiedmarketresearc[4], fnfresearch[5], marketsandmarkets[6], 以及 gartner[7] 等智库的统计和预测，图数据库市场（包括云服务）规模在2019年大约是8亿美元，将在未来6年保持25%左右的年复合增长(CAGR)至 30-40 亿美元，这大约对应于全球数据库市场 5-10% 的市场份额。

2.2.2 市场参与者

（第一代）图数据库的先行者 **Neo4j**

虽然在 1970 年代，人们已经提出了一些类似于"图”的数据模型和产品原型（例如 CODASYL[8])和相应的图语言 G/G+ 语言[9]。但真正能够让“图数据库”这个概念流行起来，不得不说到这个市场最主要的先行者 Neo4j，甚至(标签)属性图和图数据库这两个主要术语就是 Neo4j 最早提出并实践的。

> 本小节关于**Neo4j**和其创造的图查询语言**Cypher**的历史内容主要摘录自 **ISO WG3** 的工作论文**"An overview of the recent history of Graph Query Languages"**[10] 和[9]，本书作者根据最新两年的进展有删减和更新。

> 关于图查询语言**(Graph Query Language**，**GQL)** 和国际标准的制定

熟悉数据库的读者可能都知道结构化查询语言SQL。通过使用SQL，人们以接近自然语言的方式访问数据库。在 SQL 被广泛采用和标准化之前，关系型数据库的市场是非常碎片和割裂的——各家厂商的产品都有完全不同的接入访问方式，数据库产品自身的开发人员、数据库产品周边工具的开发人员、数据库最终的使用人员，都不得不学习各个厂商的完全不同的产品，在不同产品之间迁移极其困难。当1989年SQL-89标准被制定后，整个关系型数据库的市场快速收敛到SQL-89上。这大大降低了上述各种人员的学习曲线。

类似的，在图数据库领域，图语言(GQL)承担了类似于SQL的作用，是一种用户与图数据库主要的交互方式。但不同于SQL-89这种国际标准，GQL还没有任何国际标准。目前有两种主流的图语言：

Neo4j的Cypher (及其后续——ISO正在制定过程中的 GQL-standard 草案)和Apache TinkerPop的Gremlin。前者通常被称为声明式语言(Declarative query language)——也即用户只需要告诉系统“要什么”，而不管“怎么做”；后者通常被称为命令式语言(Imperative query language)，用户会显式地指定系统的操作。

GQL国际标准正在制定过程中。

Copyright © Yueshu Graph Technology - All rights reserved.

年表简述

- 2000 年，Neo4j 的创始人产生将数据建模成网络（network）的想法。
- 2001 年，Neo4j 开发了最早的核心部分代码。
- 2007 年，Neo4j 开始以一个公司的方式运作。
- 2009 年，Neo4j 团队借鉴 XPath 作为图查询语言，Gremlin[11]最初也是基于这个想法。
- 2010 年，Neo4j 的员工 Marko Rodriguez 采用术语属性图（Property Graph）来描述 Neo4j 和 Tinkerpop / Gremlin 的数据模型。
- 2011 年，第一个公开发行版本 Neo4j 1.4; 并发布了Cypher的第一个版本。
- 2012 年，Neo4j 1.8 为 Cypher 增加写入图的能力。Neo4j 2.0 增加了标签和索引，Cypher 成为一种声明式的语言。
- 2015 年，Neo4j 将 Cypher 开源为 openCypher。
- 2017 年，ISO WG3 工作组开始讨论如何将属性图查询能力引入 SQL。
- 2018 年 12 月，从 Neo4j 3.5 开始其核心部分转为闭源。
- 2019 年, ISO 正式立项两个项目(ISO/IEC JTC 1 N 14279和ISO/IEC JTC 1/SC 32 N 3228），启动关于图数据库语言国际标准的制定工作。
- 2021 年，Neo4j 完成 F 轮 3.25 亿美元的融资，是整个数据库（包括关系型）历史上最大一轮融资。

NEO4J 的早期历史

Neo4j 和属性图这种数据模型，最早构想于 2000 年。Neo4j 的创始人们当时在开发一个媒体管理系统，所使用的数据库的 schema 经常会发生重大变化。为了支持这种灵活性，Neo4j 的联合创始人 Peter Neubauer，受到 Informix Cocoon 的启发，希望将系统能够建模为一种概念相互连接的网络。印度理工学院孟买分校的一群研究生们实现了最早的原型。Neo4j 的联合创始人 Emil Eifrém 和这些学生们花了一周的时间，将 Peter 最初的想法扩展成为一个更抽象的模型：节点通过关系连接，key-value 作为节点和关系的属性。这群人开发了一个 Java API 来和这种数据模型交互，并在关系型数据库之上实现了一个抽象层。

虽然这种网络模型极大的提高了生产力，但是性能一直很差。所以 Neo4j 联合创始人 Johan Svensson 花了不少精力，为这种网络模型实现了一个原生的数据管理系统。这个就成为了 Neo4j。在最初的几年，Neo4j 作为一个内部产品很成功。在 2007 年，Neo4j 的知识产权转移给了一家独立的数据库公司。

在 Neo4j 的第一个公开发行版中（Neo4j 1.4，2011 年），数据模型由节点和有类型的边构成，节点和边都有 key-value 组成的属性。Neo4j 的早期版本没有任何的索引，应用程序只能从根节点开始自己构造查询结构（search structure）。因为这样对于应用程序非常笨重，Neo4j 2.0（2013.12）引入了一个新概念——点上的标签（label）。基于点标签，Neo4j 可以为一些预定义的节点属性建立索引。

"节点"、"关系"、"属性"、"关系只能有一个标签"、"节点可以有零个或者多个标签"，以上这些概念构成了 Neo4j 属性图的数据模型定义。随着后来增加的索引功能，让 Cypher 成为了与 Neo4j 交互的主要方式。因为这样应用程序开发者只需要关注于数据本身，而不是上段提到的那个开发者自己构建的查询结构（search structure）。

GREMLIN 的创造

Gremlin是基于Apache TinkerPop开发的图语言，其风格接近于一连串的函数（过程）调用。最初 Neo4j 的查询方式是通过 Java API。应用程序可以将查询引擎作为库(library)嵌入到应用程序中，然后使用 API 来查询图。

就在这段时间，NoSQL 这个概念开始出现。NoSQL 型的数据库引擎一般用 REST 和 HTTP 来交互和查询。Neo4j 的早期员工 Tobias Lindaaker、Ivarsson、Peter Neubauer 、Marko Rodriguez用 XPath 作为图查询，Groovy 提供循环结构，分支和计算（等图灵完毕的功能）。 这个就是 Gremlin 最初的原型。 2009 年 11 月发布了第一个版本。

后来，Marko 发现同时用两种不同的解析器（XPath 和 Groovy）有很多问题，就将 Gremlin 改为基于 Groovy 的一种领域特定语言（DSL）。

CYPHER 的创造

Gremlin 和 Neo4j 的 Java API 一样，最初用于表达如何查询数据库的一种过程（Procedural）。它允许更短的语法来表达查询，也允许通过网络远程访问数据库。Gremlin 这种过程式的特性，需要用户知道如何采用最好的办法查询结果，这样对于应用程序开发人员来说仍旧有负担。与此同时，在过去30年中，声明式语言 SQL 取得了极大的成功：SQL 可以将“获取数据的声明方式”和“引擎如何获取数据”相分开，所以 Neo4j 的工程师们希望开发一种声明式的图查询语言。

2010 年，Andrés Taylor 作为工程师加入 Neo4j。受 SQL 启发，他启动了一个项目来开发图查询语言，而这种新语言于 2011 年 Neo4j 1.4 发布，这种新语言就是如今大多数图查询语言的先祖——Cypher 。

Cypher 的语法基础，是用 "ASCII艺术(ASCII art)" 来描述图模式。这种方式最初来源于 Neo4j 工程师团队在源代码中评注如何描述图模式。可以看下图的例子：

Copyright © Yueshu Graph Technology - All rights reserved.

ASCII art 简单说，就是如何用可打印文本来描述点和边。Cypher 文本用 () 表示点，`-[]->` 表示边。`(query)-[modeled as]->(drawing)` 来表示 `起点 query`，`终点 drawing`，`边 modeled as`，这样一个最简单的图关系(也可以称为图模式)。

Cypher 第一个版本实现了对图的读取，但是需要用户说明从哪些节点开始查询。只有从这些节点开始，才可以支持图的模式匹配。

在后面的版本，2012 年 10 月发布的 Neo4j 1.8 中，Cypher 增加了修改图的能力。但查询还是需要指明从哪些节点开始。

2013 年 12 月，Neo4j 2.0 引入了 label 的概念，label 本质上是个索引。这样，查询引擎就可以利用索引，来选择模式所匹配到的节点，而不需要用户指定开始查询的节点。

Copyright © Yueshu Graph Technology - All rights reserved.

随着 Neo4j 的普及，Cypher 有着广泛的开发者群体，在各行各业的得到广泛的使用。至今仍是最受欢迎的图查询语言。

2015 年 9 月，Neo4j 发起成立了 openCypher Implementors Group（oCIG），将 Cypher 开放为 openCypher，通过开源的方式来治理和推进语言自身的演化。

后续

Cypher 启发了一系列后续的图查询语言，包括

2015 年，Oracle 发布图引擎PGX使用的图语言PGQL。

2016 年，Linked Data Benchmarking Council, LDBC 是一个行业知名的图性能基准评测机构。LDBC 发布 G-CORE

2018 年，基于 Redis 的图库(library) RedisGraph 采用 Cypher 作为其图语言

2019 年，国际标准组织 ISO 启动两个项目，基于 openCypher, PGQL, GSQL[12], and G-CORE 等现有业界成果，启动图语言国际标准的制定过程

2019 年，悦数图数据库以 openCypher 为基础发布其扩展的图语言 NebulaGraph Query Language, nGQL。

分布式图数据库

大约 2005-2010 年，随着 Google 云计算"三驾马车"的发布，各种分布式的架构开始越来越流行，其中就包括以开源方式运作的 Hadoop 和 Cassandra 等。这里包括几个方面的影响：

1. 由于数据量和计算量越来越大，相比于单机(例如 Neo4j)或者小型机这种方案，分布式系统的技术和成本优势更加明显；而同时，分布式系统使得应用程序在访问这成千上万台机器时，就如同访问本地的系统一样，不需要在代码层面进行过多改造；
2. 开源方式使得更多的人（包括代码开发者、数据科学家、产品经理等）以更加低成本和有效的方式参与新兴的技术，并反馈给社区。

严格说，Neo4j 也提供了不少的分布式的能力，但都和业界意义上的（对等、分片的）分布式系统有较大的不同：

- Neo4j 3.X 要求全量数据必须存放在单机中。虽然其也提供多机之间(Master-slave/slave)做全量复制和高可用，但数据不可切分为不同子图存放。

Copyright © Yueshu Graph Technology - All rights reserved.

Cluster architecture

- Neo4j 4.X 允许在不同机器上各存放一部分数据（子图），然后在应用层需通过一定方式拼装后(其称为编织 Fabric)[13]，将读写分发到各个机器上。这种做法需要应用层代码有大量的参与和工作。例如，设计如何把不同子图应该放置在哪些机器上，如何将从各机器获取的部分结果重新编织为最终的结果。

其语法风格大体是

```
USE graphA  # S1.1 从 Shard A 读
MATCH (movie:Movie)
Return movie.title AS title
    UNION   # S2. 在代理服务器 Join 结果
USE graphB  # S1.2 从 Shard B 读
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
MATCH  (move:Movie)
RETURN movie.title AS title
```

第二代（分布式）图数据库：**TITAN** 和其后继者 **JANUSGRAPH**

2011 年，Aurelius 公司成立，致力于开发一个开源的分布式图数据库 Titan[14]。到 2015 年 Titan 的第一个正式版发布，Titan 后端可以支持多种主流的分布式存储架构（例如 Cassandra, HBase, Elasticsearch, BerkeleyDB)，并可以复用 Hadoop 生态的诸多便利，前端以 Gremlin 为统一的查询语言。对于程序员使用、开发和社区参与都很方便。大规模的图，可以分片后存放在 HBase 或者 Cassarndar上(这些当时都已经是相对成熟的分布式存储方案)，Gremlin 语言虽然略微冗长但相对功能完备。整个方案在当时(2011-2015)体现了不错的竞争力。

下图显示了 2012年 - 2015 年，Titan 和 Neo4j 在 GitHub.com 上 star 的增长情况。

2015 年 Aurelius(Titan) 被 DataStax 收购，这之后 Titan 逐渐转变为一个闭源的商业产品 (DataStax Enterprise Graph)。

在 Aurelius(Titan) 被收购后，市场对于开源分布式的图数据库一直仍有比较强烈的需求，而当时市场上成熟和活跃的产品并不多。大数据时代，数据仍在远快于摩尔定律的速度，源源不断的产生。Linux 基金会以及一些技术巨头(Expero, Google, GRAKN.AI, Hortonworks, IBM and

Copyright © Yueshu Graph Technology - All rights reserved.

Amazon) 在2017年，复制并分叉(fork)了原有的Titan项目，并启动为一个新项目 JanusGraph[15]。之后大多数的社区工作，包括开发、测试、发布和推广都逐步转移到了新的 JanusGraph。

下图显示了两个项目2012-2021年日常代码提交(pull request)的变化情况，可以观察到几点：

1. 即使 Aurelius(Titan) 2015 年被收购后，其开源代码仍有一定的活跃度(左侧)，但增速已经明显放缓。这体现了社区的力量。
2. 新项目 JanusGraph 项目在 2017 年 1 月启动后，其社区迅速活跃起来，短短一年时间就超越了 Titan 过去 5 年累计的 pull request 数量。而与此同时，Titan 开源项目就此停滞。

同期知名产品 **ORIENTDB, TIGERGRAPH, ARANGODB,** 和 **DGRAPH**

此后更多的厂商加入整个市场，除了由Linux基金会托管的 JanusGraph，还有一些由商业公司主导开发的分布式图数据，各方采用的数据模型和访问方式也有明显的不同。 本文不做一一介绍，仅简单列出主要区别。

厂商名	创立时间	核心产品名	开源协议	数据模型	查询语言
OrientDB LTD (2017 年 被 SAP 收购)	2011	OrientDB	开源	文档 + KV + 图	OrientDB SQL (基于SQL扩展的图能力)
GraphSQL (后改名 TigerGraph)	2012	TigerGraph	商业版本	图(分析)	GraphSQL (类SQL风格)
ArangoDB GmbH	2014	ArangoDB	Apache License 2.0	文档 + KV + 图	AQL (同时操作文档, KV 和图)
DGraph Labs	2016	DGraph	Apache Public License 2.0 + Dgraph Community License	原 RDF，后改为 GraphQL	GraphQL+-

Copyright © Yueshu Graph Technology - All rights reserved.

传统巨头微软、亚马逊和甲骨文纷纷入场

除了聚焦于图产品的厂商外，传统巨头也纷纷进入这个领域。

Microsoft Azure Cosmos DB[16] 是一个在微软云上的多模数据库云服务，可以提供SQL、文档、图、key-value等多种能力； Amazon AWS Neptune[17] 是一种由 AWS 提供图数据库云服务, 可以提供属性图和 RDF 两种数据模型； Oracle graph[18] 是关系型数据库巨头 Oracle 在图技术与图数据库方向的产品。

分布式图数据库悦数图数据库

在下一章，我们将正式介绍新一代分布式图数据库悦数图数据库。

---

1. https://db-engines.com/en/ranking_categories ↩
2. https://www.yellowfinbi.com/blog/2014/06/yfcommunitynews-big-data-analytics-the-need-for-pragmatism-tangible-benefits-and-real-world-case-165305 ↩
3. https://www.gartner.com/smarterwithgartner/gartner-top-10-data-and-analytics-trends-for-2021/ ↩
4. https://www.verifiedmarketresearch.com/product/graph-database-market/ ↩
5. https://www.globenewswire.com/news-release/2021/01/28/2165742/0/en/Global-Graph-Database-Market-Size-Share-to-Exceed-USD-4-500-Million-By-2026-Facts-Factors.html ↩
6. https://www.marketsandmarkets.com/Market-Reports/graph-database-market-126230231.html ↩
7. https://www.gartner.com/en/newsroom/press-releases/2019-07-01-gartner-says-the-future-of-the-database-market-is-the ↩
8. https://www.amazon.com/Designing-Data-Intensive-Applications-Reliable-Maintainable/dp/1449373321 ↩
9. I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A Graphical Query Language Supporting Recursion. In Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data, pages 323–330. ACM Press, May 1987. ↩↩
10. "An overview of the recent history of Graph Query Languages". Authors: Tobias Lindaaker, U.S. National Expert.Date: 2018-05-14 ↩
11. Gremlin是基于Apache TinkerPop开发的图语言(https://tinkerpop.apache.org/)。 ↩
12. https://docs.tigergraph.com/dev/gsql-ref ↩
13. https://neo4j.com/fosdem20/ ↩
14. https://github.com/thinkaurelius/titan ↩
15. https://github.com/JanusGraph/janusgraph ↩
16. https://azure.microsoft.com/en-us/free/cosmos-db/ ↩
17. https://aws.amazon.com/cn/neptune/ ↩
18. https://www.oracle.com/database/graph/ ↩

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.3 相关技术

本节主要介绍两个和分布式图数据库关系密切的领域，数据库方面和图技术方面。

2.3.1 数据库方面

关系型数据库

关系型数据库，是指采用了关系模型来组织数据的数据库。关系模型为二维表格模型，一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。说到关系型数据库，大多数人都会想到 MySQL。MySQL 是目前最流行的数据库管理系统之一，支持使用最常见的结构化查询语言（SQL）进行数据库操作，并以表格、行、列的形式存储数据。这种存储数据的方法源自埃德加·科德（Edgar Frank Codd）于 1970 年提出的关系型数据模型。

在关系型数据库中，可以为待存储的每种类型的数据创建一个表。例如，球员表用来存储所有的球员信息，球队表用来存储球队信息等。SQL 表中的每行数据都必须包含一个主键（primary key）。主键是该行数据的唯一标识符。一般地，主键作为字段 ID 都是随行数自增的。关系型数据库自问世以来一直为计算机行业提供着非常好的服务，并将未来很长的时间内继续服务下去。

如果你用过 Excel、WPS 或其他类似的应用，你就会大概了解到关系数据库是如何工作的。首先设置好列，然后在对应的列下添加行数据。你可以对某一列数据进行求平均值或其他聚合操作，这与在关系型数据库 MySQL 中求平均值的操作类似。而 Excel 中的数据透视表则相当于在关系型数据库 MySQL 中使用聚合函数和 CASE 语句对数据进行查询。一个 Excel 文件可以有多张表，一张表就相当于 MySQL 的一张表。一个 Excel 文件则类似于一个 MySQL 数据库。

关系型数据库中的关系

与图数据库不同，关系型数据库（或 SQL 型的数据库）中的边也是作为实体存储在专门的边表中的。先创建两个表，球员（player）和球队（team），然后再创建表 player_team 作为边表。边表通常由相关的表 join 而成。例如，此处的边表 player_team 就由球员表和球队表 join 而成。

这种存储边的方式在关联小型数据集时问题并不大，但是当关系型数据库中的关系太多时，问题就出现了。事实上，关系型数据库是非常“反关系的”。具体来说，当你只想查询一个球员的队友时，你必须对表中的所有数据进行 join 操作，然后再过滤掉你不需要的所有数据，当你的数据集达到一定规模时，这将给关系型数据库带来巨大压力。如果你想关联多张不同的表，可能在 join 爆炸（join bombs）前系统就已经无法响应了。

关系型数据库起源

上文提到，关系型数据模型最早是由 IBM 的工程师埃德加·科德（Edgar Frank Codd）于 1970 年提出的。科德写了几篇数据库管理系统方面的论文，论述了关系型数据模型的潜力。关系型数据模型不依赖于数据链接列表（网状数据或层级数据），而是更多依赖于数据集。他使用元组演算（tuple calculus）的数学方法论证了这些数据集能够完成与导航数据库管理系统相同的任务。唯一的要求是，关系型数据模型需要一种合适的查询语

Copyright © Yueshu Graph Technology - All rights reserved.

言，以保证数据库的一致性要求。这就为后来声明型的结构化查询语言（SQL）提供了灵感来源。IBM 的 R 系统是关系型数据模型的最早使用者之一。然而，由前 IBM 员工拉里·埃里森创办的名叫软件开发实验室的小公司在市场上击败了 IBM。该公司的产品就是后来为我们熟知的 Oracle。

由于“关系数据库”在当时是一个比较时髦的词汇，因此许多数据库供应商都喜欢在其产品名称中使用这个词汇，尽管他们的产品实际上并不是关系型的。为了防止这种情况并减少关系型数据模型的错误使用，科德提出了著名科德 12 定律（Codd's 12 rules）。所有关系型数据系统都必须遵循科德 12 定律。

NoSQL 数据库

图数据库并不是可以克服关系型数据库缺点的唯一替代方案。现在市面上还有很多非关系型数据库的产品，这些产品都可以叫做 NoSQL。NoSQL 一词最早于上世纪 90 年代末提出，可以解释为“非 SQL” 或“不仅是 SQL”，具体解释要根据语境判断。为便于理解，这里 NoSQL 可以解释成 “非关系型数据库”。不同于关系型数据库，NoSQL 数据库提供的数据存储、检索机制并不是基于表关系建模的。 NoSQL 数据库可以分为四类：

- 键值存储（key-value stores）
- 列式存储（column-family stores）
- 文件存储（document stores）
- 图数据库（graph databases）

下面将分别介绍这四类数据库。

键值存储

键值存储，顾名思义，就是使用键值对存储数据的数据库。不同于关系型数据库，键值存储是没有表和列的。如果一定要做类比，键值数据库本身就像一张有很多列（也就是键）的大表。在键值存储数据库中，数据（即键值对中的值）都是通过键来存储和查询的，通常用哈希列表来实现。这比传统的 SQL 数据库要简单得多，而且对于某些 web 应用来说，这就足够了。

键值模型对于 IT 系统来说优势在于简单、易部署。多数情况下，这种存储方式对非关联的数据很适用。如是只是存储数据而无需查询的话，使用这种存储方法就没有问题。但是如果 DBA 只对部分值进行查询或更新的时候，键值模型就显得效率低下了。常见的键值存储数据库有：Redis、Voldemort、Oracle BDB。

列式存储

NoSQL 数据库的列式存储与 NoSQL 数据库的键值存储有许多相似之处，因为列式存储仍然在使用键进行存储和检索。区别在于列式存储数据库中，列是最小的存储单元，每一列均由键、值以及用于版本控制和冲突解决的时间戳组成。这在分布式扩展时特别有用，因为在数据库更新时，可以使用时间戳定位过期数据。由于列式存储良好的扩展性，因此适用于非常大的数据集。常见的列式存储数据库有：HBase、Cassandra、HadoopDB 等。

文档存储

准确来说，NoSQL 数据库文档存储实际上也是基于键值的数据库，只不过对功能做了增强。数据仍然以键值的形式存储，但是文档存储中的值是结构化的文档，而不仅仅是一个字符串或单个值。也就是说，由于信息结构的增加，文档存储能够执行更优化的查询，并且使数据检索更加容易。因此，文档存储特别适合存储、索引并管理面向文档的数据或者类似的半结构化数据。

从技术上讲，作为一个半结构化的信息单元，文档存储中的文档可以是任何形式可用的文档，包括 XML、JOSN、YAML 等，这取决于数据库供应商的设计。 比如，JSON 就是一种常见的选择。虽然 JSON 不是结构化数据的最佳选择，但是 JSON 型的数据在前端和后端应用中都可以使用。常见的文档存储数据库有：MongoDB、CouchDB、Terrastore 等。

图存储

最后一类 NoSQL 数据库是图数据库。本书重点讨论的悦数图数据库也是一种图数据库。虽然同为 NoSQL 型数据库，但是图数据库与上述 NoSQL 数据库有本质上的差异。图数据库以点、边、属性的形式存储数据。其优点在于灵活性高，支持复杂的图形算法，可用于构建复杂的关系图谱。我们将在随后的章节中详细讨论图数据库。不过在本章中，你只要知道图数据库是一种 NoSQL 类型的数据库就可以了。常见的图数据库有：悦数图数据库、Neo4j、OrientDB 等。

2.3.2 图技术方面

来看一张 2020 年的图技术全景[1]

Copyright © Yueshu Graph Technology - All rights reserved.

和图有关联的技术有很多，可以大致分为这么几类：

- 基础设施：包括图数据库、图计算(处理)引擎、图深度学习、云服务等。
- 应用：包括可视化、知识图谱、反诈骗、网络安全、社交网络等。
- 开发工具：包括图查询语言、建模工具、开发框架和库。
- 电子书籍和会议等。

图语言

在上一节中，我们介绍了图语言的历史。在这一节中，我们对图语言的功能做一个分类：

- 近邻查询：查询给定点或者边的邻边、邻点，或者是 K 跳的近邻。
- 图模式匹配(Pattern matching)：找到一个/所有的子图，满足给定的图模式；这个问题非常接近于"子图同构映射(subgraph isomorphism)"——虽然两个看上去不同的图，但其实是一模一样的[2]。

Copyright © Yueshu Graph Technology - All rights reserved.

- 可达性（连通性）问题：最常见的可达性问题就是最短路径问题。泛化一些，这类问题通常用正则路径(Regular Path Query)的方式来描述——一系列联通的点组构成了一条路径，而该路径需要满足某种正则表达。
- 分析型问题：通常与一些汇聚型算子相关（平均值、count、最大值、点的出入度），或者度量所有两两点之间的距离、某节点与其他节点之间的互动程度（介数中心性）等。

图数据库**(Graph Database)**与图处理**(Graph processing)**系统

一个图系统通常会涉及到复杂的数据流水线[3]，从数据源(左边)到处理输出(右边)，会经过多个数据加工处理环节和系统；大的模块可以分为 ETL模块，图数据库系统(Graph OLTP)，图处理系统(Graph OLAP)，基于图引擎的应用系统（BI、知识图谱等）。

Copyright © Yueshu Graph Technology - All rights reserved.

虽然这两类系统都是与图数据和图技术相关的系统，也处理类似的目标，但是他们有着不同的起源和特长（及弱点）：

- （在线）图数据库目标是图的持久化存储管理、高效的子图操作。硬盘（及网络）是目标运行设备，物理/逻辑数据映射，数据完整性和（故障）一致性是主要目标。每一个请求通常只会涉及到全图的一小部分，通常可以在一台服务器上完成；单个请求时延通常在毫秒到秒级别，请求并发量通常在几千到几十万。早期的 Neo4j 是图数据库领域的起源之一。
- （离线）图处理系统目标是全图的大批量、并行、迭代、处理与分析，内存（及网络）是目标运行设备。每一个请求会涉及到所有的图节点，需要所有的服务器参与完成；单个请求的时延通常在分钟到小时（天），请求并发量通常为个位数。Google 的 Pregel[4] 是图处理系统的典型起源代表，它的点中心编程抽象与BSP的运行模式构成的编程范式，相比之前 Hadoop Map-Reduce 是更为图友好的 API 抽象。

5

图的分片方式

对于一个大规模的图数据来说，是很难存放在单个服务器的内存中的，即使仅仅存放图结构本身也不够。而且通过增加单服务器的能力，其成本价格通常成指数级别上升。 此外，随着数据量的增加，例如到达千亿级别的时候，已经超过了市面上所有商用服务器的容量能力。

与此对应的，另外一个经常使用的方案，是对数据进行分片，并将每个分片放置在不同的服务器上（并进行冗余备份），以此来增加可靠性和性能。对于一些 NoSQL 型的系统，例如 key-value 或者文档型的系统来说，这个分片方式是比较直观和自然的；通常可以根据 key 或者 docID，来将每个记录或者数据单元(key-value, doc)放在不同的服务器上。

但是图这种数据结构的分片通常不那么直观，这是因为通常图是“全联通”的，每个点通常只要6跳就可以联通到其他任何节点； 而理论上早已证明图的划分问题是 NP 的。 与此同时，当把整个图数据分散到多个服务器时，跨服务器的网络访问时延10倍于同一个服务器内部的硬件(内存)访问时间；因此对于一些深度优先遍历的场景，会发生大量的跨网络访问，导致整体时延极高。

6

Copyright © Yueshu Graph Technology - All rights reserved.

另一方面，通常图有着明显的幂律分布；少量节点的邻边稠密程度远大于平均的节点，虽然处理这些节点通常可以在同一台服务器内，减少了跨网络访问，但这也意味着这些服务器压力会远大于平均。

因此，常见的图分片(Sharding)方式有几类：

- 偏应用层面的分片：应用层感知并控制每个点和边应该落在哪个分片上，一般来说可以根据点和边的类型（比如业务意义来人为）判断。将一组相同类型的点放在一个分片，另一组相同类型的点放在另一个分片。当然，为了高可靠，分片本身还可以做多副本。在应用使用时，从各个分片取回所要的点和边，然后在偏应用侧（或者某个代理服务器端），将取回的数据拼装成最终的结果。其典型代表是 Neo4j 4.x 的 Fabric。

Copyright © Yueshu Graph Technology - All rights reserved.

- 使用分布式的缓存层：在硬盘之上增加内存缓存层，并对重要的部分分片和数据（例如图结构）进行缓存，并预热这部分缓存。
- 增加（只读）副本或视图 (View)：为部分图分片增加只读的副本或者建立一个视图，将较重的读请求负载通过这些分片服务器承担。
- 进行细颗粒度的图划分：例如将点和边组成多个小分片（Partition），而不是一台服务器一个大分片（Sharding），再将关联性较强的 Partition 尽量放置在同一个服务器上[7]。

具体工程实践时，也会混合使用上述几种方式。通常，离线的图处理系统会通过一个ETL过程，将图进行一定程度的预处理以提高局部性；而在线图数据库系统通常会选择周期性的数据再平衡过程来提高数据局部性。

Copyright © Yueshu Graph Technology - All rights reserved.

一些技术上的挑战

在文献[8]中，对于无处不在的大图和挑战做了详尽的调研，下面是其列出的十大图技术挑战：

- 可扩展性(软件可以处理更大的图规模)：包括大图的加载、更新、图计算和图遍历，触发器，超级节点；
- 可视化：可定制布局，大图的渲染，多层次展示，动态（更新）的展示
- 查询语言和编程 API：包括语言表达能力、标准兼容性、与现有系统的兼容性；子查询的设计和跨多图之间的关联查询
- 更快的图（及机器学习）算法
- 易用性（配置和使用）
- 性能指标与测试
- 更通用的图技术软件（例如，处理离线、在线、流式的计算）
- 图清洗（ETL）
- Debug 调试与测试

一些开源的单机图工具

对于图数据库通常会有一个误解，只要涉及到图结构的数据存取就需要存放在图数据库中，这是一种很大的浪费。

这就像也许你只需要一个 SQLite，却用了一个 Oracle。

当数据量并不大时，通常单机内存可以放下，例如数据量几千万的点边关系，使用一些单机的开源工具也可以取得很好的效果。

> **Note**
>
> 下面是一些推荐的单机图库，也可以集成在你的应用程序里面。

- JGraphT[9]：一个知名的开源 Java 图论库(library)，其实现了相当多的高效图算法。
- JUNG[10]是BSD许可下用Java编写的开源图建模和可视化框架。该框架内置了许多布局算法，以及诸如图聚类和节点中心性度量之类的分析算法。
- igraph[11]：一个轻量且功能强大的 Library, 支持R、python、C
- NetworkX[12]：数据科学家做图论分析第一选择， python。
- Cytoscape[13]：功能强大的可视化开源图分析工具。
- Gephi[14]：功能强大的可视化开源图分析工具。
- arrows.app[15]： 非常简单的脑图工具，用于可视化生成 Cypher 语句.

一些行业数据集和 **Benchmark**

LDBC

关联数据基准委员会（LDBC[16]，Linked Data Benchmark Council）是由Oracle、Intel等软硬件巨头和主流图数据库厂商Neo4j和TigerGraph等组成的非赢利机构，是图的基准指南制定者与测试结果发布机构，在行业内有着很高的影响力。

社交网络基准测试（SNB，Social Network Benchmark）是由关联数据基准委员会（LDBC）开发的面向图数据库的基准测试（Benchmark）之一，分为交互式查询（Interactive）和商业智能（BI）两个场景。其作用类似于 TPC-C, TPC-H 等测试在 SQL 型数据库中的功能，可以帮助用户比较多种图数据库产品的功能、性能、容量。

SNB 数据集模拟一个社交网络的人、发帖之间的关系，考虑了社交网络的分布属性、人的活跃度等等社交信息。

Copyright © Yueshu Graph Technology - All rights reserved.

其标准数据量从 0.1 GB (scale factor 0.1) 到 1000 GB (sf1000)，也可以生成 10 TB，100 TB等更大的数据集；其点、边数量如下表。

Scale Factor	0.1	0.3	1	3	10	30	100	300	1000
# of Persons	1.5K	3.5K	11K	27K	73K	182K	499K	1.25M	3.6M
# of nodes	327.6K	908K	3.2M	9.3M	30M	88.8M	282.6M	817.3M	2.7B
# of edges	1.5M	4.6M	17.3M	52.7M	176.6M	540.9M	1.8B	5.3B	17B

Copyright © Yueshu Graph Technology - All rights reserved.

2.3.3 一些趋势

虽然图的各种技术起源和目标并不相同，但在相互借鉴和融合

上云的趋势在加速，对于弹性能力提出更高要求

根据 Gartner 的预计，云服务一直保持较快的增速和渗透率[17]。大量的商业软件，正在从 10 年前完全私有本地逐步转向基于云服务的商业模式。云服务的一大优点是其提供了近乎无限的弹性能力；这也要求各种基于云基础设施的软件必须有更好的快速弹性扩缩容能力。

Copyright © Yueshu Graph Technology - All rights reserved.

硬件趋势，**SSD** 将成为主流的持久化设备

硬件决定了软件的架构——从发现摩尔定律的 50 年代到进入多核的 00 年代，硬件发展趋势和速度一直深刻的决定了软件的架构。数据库类系统大多围绕“硬盘+内存”设计，高性能计算型系统大多围绕“内存 + CPU”设计，分布式系统面对千兆、万兆和 RDMA 网卡的设计也完全不同。

图基于拓扑的遍历有着极其明显的随机访问特点，因此大多数早期图数据库系统都采用了“大内存 + HDD” 的架构————通过设计常驻在内存中的一些数据结构（例如B+树、Hash表等），在内存中实现随机访问目的，以优化图的拓扑遍历，再将这些随机访问转换成 HDD 所适合的顺序读写。整套软件的架构（包括存储和计算层）都必须基于和围绕这样的 IO 流程来展开。随着 SSD 价格的快速下降[18]，SSD 正在替代 HDD 成为持久化设备的主流。SSD 随机访问友好、IO 队列深、按块存取的特点与 HDD 高度顺序、随机时延极高、磁道易损坏的访问特点有着明显的不同。全部的软件架构也需要重新设计，这成为沉重的历史技术负担。

Copyright © Yueshu Graph Technology - All rights reserved.

	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030
SSD $/TB	$2,220	$1,293	$840	$538	$493	$378	$195	$128	$86	$58	$40	$29	$21	$15	$12	$9	$7	$6
HDD $/TB	$60	$53	$45	$35	$30	$26.8	$23.9	$22.2	$21	$19	$18	$17	$16	$15	$14	$14	$13	$13
Ratio SSD/HDD $/TB	37	25	19	15	16	14	8.1	5.8	4.2	3.0	2.2	1.7	1.3	1.0	0.8	0.7	0.54	0.45

Figure 4 – SSD/HDD Pricing Ratio 2013 – 2030

Source: © Wikibon, 2021.

1. https://graphaware.com/graphaware/2020/02/17/graph-technology-landscape-2020.html ↩
2. https://en.wikipedia.org/wiki/Graph_isomorphism ↩
3. The Future is Big Graphs! A Community View on Graph Processing Systems. https://arxiv.org/abs/2012.06171 ↩
4. G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In Proceedings of the International Conference on Management of data (SIGMOD), pages 135–146, New York, NY, USA, 2010. ACM ↩
5. https://neo4j.com/graphacademy/training-iga-40/02-iga-40-overview-of-graph-algorithms/ ↩
6. https://livebook.manning.com/book/graph-powered-machine-learning/welcome/v-8/ ↩
7. https://www.arangodb.com/learn/graphs/using-smartgraphs-arangodb/ ↩
8. https://arxiv.org/abs/1709.03188 ↩
9. https://jgrapht.org/ ↩
10. https://github.com/jrtom/jung ↩
11. https://igraph.org/ ↩
12. https://networkx.org/ ↩
13. https://cytoscape.org/ ↩
14. https://gephi.org/ ↩
15. https://arrows.app/ ↩
16. https://github.com/ldbc/ldbc_snb_docs ↩
17. https://cloudcomputing-news.net/news/2019/apr/15/public-cloud-soaring-to-331b-by-2022-according-to-gartner/ ↩
18. https://blocksandfiles.com/2021/01/25/wikibon-ssds-vs-hard-drives-wrights-law/ ↩

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.4 什么是悦数图数据库

悦数图数据库是一款分布式的、易扩展的原生图数据库，能够承载包含数千亿个点和数万亿条边的超大规模数据集，并且提供毫秒级查询。

2.4.1 什么是图数据库

图数据库是专门存储庞大的图形网络并从中检索信息的数据库。它可以将图中的数据高效存储为点（Vertex）和边（Edge），还可以将属性（Property）附加到点和边上。

图数据库适合存储大多数从现实抽象出的数据类型。世界上几乎所有领域的事物都有内在联系，像关系型数据库这样的建模系统会提取实体之间的关系，并将关系单独存储到表和列中，而实体的类型和属性存储在其他列甚至其他表中，这使得数据管理费时费力。

悦数图数据库作为一个典型的图数据库，可以将丰富的关系通过边及其类型和属性自然地呈现。

2.4.2 悦数图数据库的优势

高性能

基于图数据库的特性使用 C++ 编写的悦数图数据库，可以提供毫秒级查询。众多数据库中，悦数图数据库在图数据服务领域展现了卓越的性能，数据规模越大，悦数图数据库优势就越大。详情请参见 悦数图数据库 benchmarking 页面。

易扩展

悦数图数据库采用 shared-nothing 架构，支持在不停止数据库服务的情况下扩缩容。

Copyright © Yueshu Graph Technology - All rights reserved.

易开发

悦数图数据库提供 Java、Python、C++ 和 Go 等流行编程语言的客户端，更多客户端仍在开发中。详情请参见 悦数图数据库 clients。

高可靠访问控制

悦数图数据库支持严格的角色访问控制和 LDAP（Lightweight Directory Access Protocol）等外部认证服务，能够有效提高数据安全性。详情请参见验证和授权。

生态多样化

悦数图数据库开放了越来越多的原生工具，例如 NebulaGraph Studio、NebulaGraph Console、NebulaGraph Exchange 等，更多工具可以查看生态工具概览。

此外，悦数图数据库还具备与 Spark、Flink、HBase 等产品整合的能力，在这个充满挑战与机遇的时代，大大增强了自身的竞争力。

兼容 **openCypher** 查询语言

悦数图数据库查询语言，简称为 nGQL，是一种声明性的、部分兼容 openCypher 的文本查询语言，易于理解和使用。详细语法请参见 nGQL 指南。

面向未来硬件，读写平衡

闪存型设备有着极高的性能，并且价格快速下降，悦数图数据库是一个面向 SSD 设计的产品，相比于基于 HDD + 大内存的产品，更适合面向未来的硬件趋势，也更容易做到读写平衡。

灵活数据建模

用户可以轻松地在悦数图数据库中建立数据模型，不必将数据强制转换为关系表。而且可以自由增加、更新和删除属性。详情请参见数据模型。

广受欢迎

腾讯、美团、京东、快手、360 等科技巨头都在使用悦数图数据库。详情请参见 悦数图数据库官网。

2.4.3 适用场景

悦数图数据库可用于各种基于图的业务场景。为节约转换各类数据到关系型数据库的时间，以及避免复杂查询，建议使用悦数图数据库。

欺诈检测

金融机构必须仔细研究大量的交易信息，才能检测出潜在的金融欺诈行为，并了解某个欺诈行为和设备的内在关联。这种场景可以通过图来建模，然后借助悦数图数据库，可以很容易地检测出诈骗团伙或其他复杂诈骗行为。

实时推荐

悦数图数据库能够及时处理访问者产生的实时信息，并且精准推送文章、视频、产品和服务。

知识图谱

自然语言可以转化为知识图谱，存储在悦数图数据库中。用自然语言组织的问题可以通过智能问答系统中的语义解析器进行解析并重新组织，然后从知识图谱中检索出问题的可能答案，提供给提问人。

社交网络

人际关系信息是典型的图数据，悦数图数据库可以轻松处理数十亿人和数万亿人际关系的社交网络信息，并在海量并发的情况下，提供快速的好友推荐和工作岗位查询。

Copyright © Yueshu Graph Technology - All rights reserved.

2.4.4 视频

用户也可以通过视频了解什么是图数据。

- 悦数图数据库介绍视频（01 分 39 秒）

2.4.5 主题演讲

查看演讲快速了解图数据库概况。

2.4.6 相关链接

- 官方网站
- 文档首页
- 博客首页
- 论坛
- GitHub

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.5 数据模型

本文介绍悦数图数据库的数据模型。数据模型是一种组织数据并说明它们如何相互关联的模型。

2.5.1 数据模型

悦数图数据库数据模型使用 6 种基本的数据模型：

- 图空间（Space）

  图空间用于隔离不同团队或者项目的数据。不同图空间的数据是相互隔离的，可以指定不同的存储副本数、权限、分片等。

- 点（Vertex）

  点用来保存实体对象，特点如下：

- 点是用点标识符（`VID`）标识的。`VID` 在同一图空间中唯一。VID 是一个 int64，或者 fixed_string(N)。
- 点可以有 0 到多个 Tag。

> **Compatibility**
>
> 悦数图数据库 2.x 及以下版本中的点必须包含至少一个 Tag。

- 边（Edge）

  边是用来连接点的，表示两个点之间的关系或行为，特点如下：

- 两点之间可以有多条边。
- 边是有方向的，不存在无向边。
- 四元组 `<起点 VID、Edge type、边排序值 (rank)、终点 VID>` 用于唯一标识一条边。边没有 EID。
- 一条边有且仅有一个 Edge type。
- 一条边有且仅有一个 Rank，类型为 int64，默认值为 0。

> 关于
>
> Rank 可以用来区分 Edge type、起始点、目的点都相同的边。该值完全由用户自己指定。
> 读取时必须自行取得全部的 Rank 值后排序过滤和拼接。
> 不支持诸如 `next()`, `pre()`, `head()`, `tail()`, `max()`, `min()`, `lessThan()`, `moreThan()` 等函数功能，也不能通过创建索引加速访问或者条件过滤。

- 标签（Tag）

  Tag 由一组事先预定义的属性构成。

- 边类型（Edge type）

  Edge type 由一组事先预定义的属性构成。

- 属性（Property）

  属性是指以键值对（Key-value pair）形式表示的信息。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

Tag 和 Edge type 的作用，类似于关系型数据库中“点表”和“边表”的表结构。

2.5.2 有向属性图

悦数图数据库使用有向属性图模型，指点和边构成的图，这些边是有方向的，点和边都可以有属性。

下表为篮球运动员数据集的结构示例，包括两种类型的点（**player**、**team**）和两种类型的边（**serve**、**follow**）。

类型	名称	属性名（数据类型）	说明
Tag	**player**	name (string) age（int）	表示球员。
Tag	**team**	name (string)	表示球队。
Edge type	**serve**	start_year (int) end_year (int)	表示球员的行为。 该行为将球员和球队联系起来，方向是从球员到球队。
Edge type	**follow**	degree (int)	表示球员的行为。 该行为将两个球员联系起来，方向是从一个球员到另一个球员。

Note

悦数图数据库中没有无向边，只支持有向边。

Compatibility

由于悦数图数据库 3.5.1 的数据模型中，允许存在"悬挂边"，因此在增删时，用户需自行保证“一条边所对应的起点和终点”的存在性。详见 INSERT VERTEX、DELETE VERTEX、INSERT EDGE、DELETE EDGE。

不支持 openCypher 中的 MERGE 语句。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.6 路径

图论中一个非常重要的概念是路径，路径是指一个有限或无限的边序列，这些边连接着一系列的点。

路径的类型分为三种：`walk`、`trail`、`path`。关于路径的详细说明，请参见维基百科。

本文以下图为例进行简单介绍。

2.6.1 walk

`walk` 类型的路径由有限或无限的边序列构成。遍历时点和边可以重复。

查看示例图，由于 C、D、E 构成了一个环，因此该图包含无限个路径，例如 `A->B->C->D->E`、`A->B->C->D->E->C`、`A->B->C->D->E->C->D`。

> **Note**
>
> `GO` 语句采用的是 `walk` 类型路径。

2.6.2 trail

`trail` 类型的路径由有限的边序列构成。遍历时只有点可以重复，边不可以重复。柯尼斯堡七桥问题的路径类型就是 `trail`。

查看示例图，由于边不可以重复，所以该图包含有限个路径，最长路径由 5 条边组成：`A->B->C->D->E->C`。

> **Note**
>
> `MATCH`、`FIND PATH` 和 `GET SUBGRAPH` 语句采用的是 `trail` 类型路径。

在 trail 类型中，还有 `cycle` 和 `circuit` 两种特殊的路径类型，以下图为例对这两种特殊的路径类型进行介绍。

Copyright © Yueshu Graph Technology - All rights reserved.

- cycle

  `cycle` 是封闭的 `trail` 类型的路径，遍历时边不可以重复，起点和终点重复，并且没有其他点重复。在此示例图中，最长路径由三条边组成: `A->B->C->A` 或 `C->D->E->C` .

- circuit

  `circuit` 也是封闭的 `trail` 类型的路径，遍历时边不可以重复，除起点和终点重复外，可能存在其他点重复。在此示例图中，最长路径为: `A->B->C->D->E->C->A` 。

2.6.3 path

`path` 类型的路径由有限的边序列构成。遍历时点和边都不可以重复。

查看示例图，由于点和边都不可以重复，所以该图包含有限个路径，最长路径由 4 条边组成: `A->B->C->D->E` 。

2.6.4 视频

用户也可以观看视频了解路径的相关概念。

Path（03 分 09 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.7 点 VID

在一个图空间中，一个点由点的 ID 唯一标识，即 VID 或 Vertex ID。

2.7.1 VID 的特点

- VID 数据类型只可以为定长字符串 `FIXED_STRING(<N>)` 或 `INT64` 。一个图空间只能选用其中一种 VID 类型。
- VID 在一个图空间中必须唯一，其作用类似于关系型数据库中的主键（索引+唯一约束）。但不同图空间中的 VID 是完全独立无关的。
- 点 VID 的生成方式必须由用户自行指定，系统不提供自增 ID 或者 UUID。
- VID 相同的点，会被认为是同一个点。例如：
- VID 相当于一个实体的唯一标号，例如一个人的身份证号。Tag 相当于实体所拥有的类型，例如"滴滴司机"和"老板"。不同的 Tag 又相应定义了两组不同的属性，例如"驾照号、驾龄、接单量、接单小号"和"工号、薪水、债务额度、商务电话"。
- 同时操作相同 VID 并且相同 Tag 的两条 `INSERT` 语句（均无 `IF NOT EXISTS` 参数），晚写入的 `INSERT` 会覆盖先写入的。
- 同时操作包含相同 VID 但是两个不同 `TAG A` 和 `TAG B` 的两条 `INSERT` 语句，对 `TAG A` 的操作不会影响 `TAG B` 。
- VID 通常会被（LSM-tree 方式）索引并缓存在内存中，因此直接访问 VID 的性能最高。

2.7.2 VID 使用建议

- 悦数图数据库 1.x 只支持 VID 类型为 `INT64` ，从 2.x 开始支持 `INT64` 和 `FIXED_STRING(<N>)` 。在 `CREATE SPACE` 中通过参数 `vid_type` 可以指定 VID 类型。
- 可以使用 `id()` 函数，指定或引用该点的 VID。
- 可以使用 `LOOKUP` 或者 `MATCH` 语句，来通过属性索引查找对应的 VID。
- 性能上，直接通过 VID 找到点的语句性能最高，例如 `DELETE xxx WHERE id(xxx) == "player100"` ，或者 `GO FROM "player100"` 等语句。通过属性先查找 VID，再进行图操作的性能会变差，例如 `LOOKUP | GO FROM $-.ids` 等语句，相比前者多了一次内存或硬盘的随机读（`LOOKUP`）以及一次序列化（`|`）。

2.7.3 VID 生成建议

VID 的生成工作完全交给应用端，有一些通用的建议：

- （最优）通过有唯一性的主键或者属性来直接作为 VID；属性访问依赖于 VID;
- 通过有唯一性的属性组合来生成 VID，属性访问依赖于属性索引。
- 通过 snowflake 等算法生成 VID，属性访问依赖于属性索引。
- 如果个别记录的主键特别长，但绝大多数记录的主键都很短的情况，不要将 `FIXED_STRING(<N>)` 的 `N` 设置成超大，这会浪费大量内存和硬盘，也会降低性能。此时可通过 BASE64，MD5，hash 编码加拼接的方式来生成。
- 如果用 hash 方式生成 int64 VID：在有 10 亿个点的情况下，发生 hash 冲突的概率大约是 1/10。边的数量与碰撞的概率无关。

2.7.4 定义和修改 VID 与其数据类型

VID 的数据类型必须在创建图空间时定义，且一旦定义无法修改。

VID 必须在插入点时设置，且一旦设置无法修改。

2.7.5 "查询起始点"(`start vid`) 与全局扫描

绝大多数情况下，悦数图数据库的查询语句（`MATCH` 、`GO` 、`LOOKUP` ）的执行计划，必须要通过一定方式找到查询起始点的 VID（`start vid` ）。

Copyright © Yueshu Graph Technology - All rights reserved.

定位 `start vid` 只有两种方式:

1. 例如 `GO FROM "player100" OVER` 是在语句中显式的指明 `start vid` 是 "player100";
2. 例如 `LOOKUP ON player WHERE player.name == "Tony Parker"` 或者 `MATCH (v:player {name:"Tony Parker"})`，是通过属性 `player.name` 的索引来定位到 `start vid`；

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.8 服务架构

2.8.1 悦数图数据库架构总览

悦数图数据库由三种服务构成：Graph 服务、Meta 服务和 Storage 服务，是一种存储与计算分离的架构。

每个服务都有可执行的二进制文件和对应进程，用户可以使用这些二进制文件在一个或多个计算机上部署悦数图数据库集群。

下图展示了悦数图数据库集群的经典架构。

Copyright © Yueshu Graph Technology - All rights reserved.

Meta 服务

在悦数图数据库架构中，Meta 服务是由 nebula-metad 进程提供的，负责数据管理，例如 Schema 操作、集群管理和用户权限管理等。

Meta 服务的详细说明，请参见 Meta 服务。

Graph 服务和 Storage 服务

悦数图数据库采用计算存储分离架构。Graph 服务负责处理计算请求，Storage 服务负责存储数据。它们由不同的进程提供，Graph 服务是由 nebula-graphd 进程提供，Storage 服务是由 nebula-storaged 进程提供。计算存储分离架构的优势如下：

- 易扩展

  分布式架构保证了 Graph 服务和 Storage 服务的灵活性，方便扩容和缩容。

- 高可用

  如果提供 Graph 服务的服务器有一部分出现故障，其余服务器可以继续为客户端提供服务，而且 Storage 服务存储的数据不会丢失。服务恢复速度较快，甚至能做到用户无感知。

- 节约成本

  计算存储分离架构能够提高资源利用率，而且可根据业务需求灵活控制成本。

- 更多可能性

  基于分离架构的特性，Graph 服务将可以在更多类型的存储引擎上单独运行，Storage 服务也可以为多种目的计算引擎提供服务。

Graph 服务和 Storage 服务的详细说明，请参见 Graph 服务和 Storage 服务。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.8.2 Meta 服务

本文介绍 Meta 服务的架构和功能。

Meta 服务架构

Meta 服务是由 nebula-metad 进程提供的，用户可以根据场景配置 nebula-metad 进程数量：

- 测试环境中，用户可以在悦数图数据库集群中部署 1 个或 3 个 nebula-metad 进程。如果要部署 3 个，用户可以将它们部署在 1 台机器上，或者分别部署在不同的机器上。
- 生产环境中，建议在悦数图数据库集群中部署 3 个 nebula-metad 进程。请将这些进程部署在不同的机器上以保证高可用。

所有 nebula-metad 进程构成了基于 Raft 协议的集群，其中一个进程是 leader，其他进程都是 follower。

leader 是由多数派选举出来，只有 leader 能够对客户端或其他组件提供服务，其他 follower 作为候补，如果 leader 出现故障，会在所有 follower 中选举出新的 leader。

> **Note**
>
> leader 和 follower 的数据通过 Raft 协议保持一致，因此 leader 故障和选举新 leader 不会导致数据不一致。更多关于 Raft 的介绍见 Storage 服务。

Meta 服务功能

管理用户账号

Meta 服务中存储了用户的账号和权限信息，当客户端通过账号发送请求给 Meta 服务，Meta 服务会检查账号信息，以及该账号是否有对应的请求权限。

Copyright © Yueshu Graph Technology - All rights reserved.

更多悦数图数据库的访问控制说明，请参见身份验证。

管理分片

Meta 服务负责存储和管理分片的位置信息，并且保证分片的负载均衡。

管理图空间

悦数图数据库支持多个图空间，不同图空间内的数据是安全隔离的。Meta 服务存储所有图空间的元数据（非完整数据），并跟踪数据的变更，例如增加或删除图空间。

管理 **SCHEMA** 信息

悦数图数据库是强类型图数据库，它的 Schema 包括 Tag、Edge type、Tag 属性和 Edge type 属性。

Meta 服务中存储了 Schema 信息，同时还负责 Schema 的添加、修改和删除，并记录它们的版本。

更多悦数图数据库的 Schema 信息，请参见数据模型。

管理 **TTL** 信息

Meta 服务存储 TTL（Time To Live）定义信息，可以用于设置数据生命周期。数据过期后，会由 Storage 服务进行处理，具体过程参见 TTL。

管理作业

Meta 服务中的作业管理模块负责作业的创建、排队、查询和删除。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.8.3 Graph 服务

Graph 服务主要负责处理查询请求，包括解析查询语句、校验语句、生成执行计划以及按照执行计划执行四个大步骤，本文将基于这些步骤介绍 Graph 服务。

Graph 服务架构

查询请求发送到 Graph 服务后，会由如下模块依次处理：

1. **Parser**：词法语法解析模块。
2. **Validator**：语义校验模块。
3. **Planner**：执行计划与优化器模块。
4. **Executor**：执行引擎模块。

Parser

Parser 模块收到请求后，通过 Flex（词法分析工具）和 Bison（语法分析工具）生成的词法语法解析器，将语句转换为抽象语法树（AST），在语法解析阶段会拦截不符合语法规则的语句。

例如 `GO FROM "Tim" OVER like WHERE properties(edge).likeness > 8.0 YIELD dst(edge)` 语句转换的 AST 如下。

Copyright © Yueshu Graph Technology - All rights reserved.

Validator

Validator 模块对生成的 AST 进行语义校验，主要包括：

- 校验元数据信息

  校验语句中的元数据信息是否正确。

  例如解析 `OVER` 、`WHERE` 和 `YIELD` 语句时，会查找 Schema 校验 Edge type、Tag 的信息是否存在，或者插入数据时校验插入的数据类型和 Schema 中的是否一致。

- 校验上下文引用信息

  校验引用的变量是否存在或者引用的属性是否属于变量。

  例如语句 `$var = GO FROM "Tim" OVER like YIELD dst(edge) AS ID; GO FROM $var.ID OVER serve YIELD dst(edge)` ，Validator 模块首先会检查变量 `var` 是否定义，其次再检查属性 `ID` 是否属于变量 `var` 。

- 校验类型推断

  推断表达式的结果类型，并根据子句校验类型是否正确。

  例如 `WHERE` 子句要求结果是 `bool` 、`null` 或者 `empty` 。

- 校验 `*` 代表的信息

  查询语句中包含 `*` 时，校验子句时需要将 `*` 涉及的 Schema 都进行校验。

  例如语句 `GO FROM "Tim" OVER * YIELD dst(edge), properties(edge).likeness, dst(edge)` ，校验 `OVER` 子句时需要校验所有的 Edge type，如果 Edge type 包含 `like` 和 `serve` ，该语句会展开为 `GO FROM "Tim" OVER like,serve YIELD dst(edge), properties(edge).likeness, dst(edge)` 。

- 校验输入输出

  校验管道符（|）前后的一致性。

  例如语句 `GO FROM "Tim" OVER like YIELD dst(edge) AS ID | GO FROM $-.ID OVER serve YIELD dst(edge)` ，Validator 模块会校验 `$-.ID` 在管道符左侧是否已经定义。

校验完成后，Validator 模块还会生成一个默认可执行，但是未进行优化的执行计划，存储在目录 `src/planner` 内。

Planner

如果配置文件 `nebula-graphd.conf` 中 `enable_optimizer` 设置为 `false` ，Planner 模块不会优化 Validator 模块生成的执行计划，而是直接交给 Executor 模块执行。

如果配置文件 `nebula-graphd.conf` 中 `enable_optimizer` 设置为 `true` ，Planner 模块会对 Validator 模块生成的执行计划进行优化。如下图所示。

Copyright © Yueshu Graph Technology - All rights reserved.

- 优化前

  如上图右侧未优化的执行计划，每个节点依赖另一个节点，例如根节点 `Project` 依赖 `Filter` 、`Filter` 依赖 `GetNeighbor` ，最终找到叶子节点 `Start` ，才能开始执行（并非真正执行）。

  在这个过程中，每个节点会有对应的输入变量和输出变量，这些变量存储在一个哈希表中。由于执行计划不是真正执行，所以哈希表中每个 key 的 value 值都为空（除了 `Start` 节点，起始数据会存储在该节点的输入变量中）。哈希表定义在仓库 `nebula-graph` 内的 `src/context/ExecutionContext.cpp` 中。

  例如哈希表的名称为 `ResultMap` ，在建立 `Filter` 这个节点时，定义该节点从 `ResultMap["GN1"]` 中读取数据，然后将结果存储在 `ResultMap["Filter2"]` 中，依次类推，将每个节点的输入输出都确定好。

- 优化过程

  Planner 模块目前的优化方式是 RBO（rule-based optimization），即预定义优化规则，然后对 Validator 模块生成的默认执行计划进行优化。新的优化规则 CBO（cost-based optimization）正在开发中。优化代码存储在仓库 `nebula-graph` 的目录 `src/optimizer/` 内。

  RBO 是一个自底向上的探索过程，即对于每个规则而言，都会由执行计划的根节点（示例是 `Project` ）开始，一步步向下探索到最底层的节点，在过程中查看是否可以匹配规则。

  如上图所示，探索到节点 `Filter` 时，发现依赖的节点是 `GetNeighbor` ，匹配预先定义的规则，就会将 `Filter` 融入到 `GetNeighbor` 中，然后移除节点 `Filter` ，继续匹配下一个规则。在执行阶段，当算子 `GetNeighbor` 调用 Storage 服务的接口获取一个点的邻边时，Storage 服务内部会直接将不符合条件的边过滤掉，这样可以极大地减少传输的数据量，该优化称为过滤下推。

Executor

Executor 模块包含调度器（Scheduler）和执行器（Executor），通过调度器调度执行计划，让执行器根据执行计划生成对应的执行算子，从叶子节点开始执行，直到根节点结束。如下图所示。

Copyright © Yueshu Graph Technology - All rights reserved.

每一个执行计划节点都一一对应一个执行算子，节点的输入输出在优化执行计划时已经确定，每个算子只需要拿到输入变量中的值进行计算，最后将计算结果放入对应的输出变量中即可，所以只需要从节点 `Start` 一步步执行，最后一个算子的输出变量会作为最终结果返回给客户端。

代码结构

悦数图数据库的代码层次结构如下：

```
|--src
   |--graph
      |--context    //校验期和执行期上下文
      |--executor   //执行算子
      |--gc         //垃圾收集器
      |--optimizer  //优化规则
      |--planner    //执行计划结构
      |--scheduler  //调度器
      |--service    //对外服务管理
      |--session    //会话管理
      |--stats      //运行指标
      |--util       //基础组件
      |--validator  //语句校验
      |--visitor    //visitor表达式
```

视频

用户也可以通过视频全方位了解悦数图数据库的查询引擎。

- nMeetup·上海 ｜全面解析 Query Engine（33 分 30 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

2.8.4 Storage 服务

悦数图数据库的存储包含两个部分，一个是 Meta 相关的存储，称为 Meta 服务，在前文已有介绍。

另一个是具体数据相关的存储，称为 Storage 服务。其运行在 nebula-storaged 进程中。本文仅介绍 Storage 服务的架构设计。

优势

- 高性能（自研 KVStore）
- 易水平扩展（Shared-nothing 架构，不依赖 NAS 等硬件设备）
- 强一致性（Raft）
- 高可用性（Raft）
- 支持向第三方系统进行同步（例如全文索引）

Storage 服务架构

Storage 服务是由 nebula-storaged 进程提供的，用户可以根据场景配置 nebula-storaged 进程数量，例如测试环境 1 个，生产环境 3 个。

Copyright © Yueshu Graph Technology - All rights reserved.

所有 nebula-storaged 进程构成了基于 Raft 协议的集群，整个服务架构可以分为三层，从上到下依次为：

- Storage interface 层

  Storage 服务的最上层，定义了一系列和图相关的 API。API 请求会在这一层被翻译成一组针对分片的 KV 操作，例如：

- `getNeighbors`：查询一批点的出边或者入边，返回边以及对应的属性，并且支持条件过滤。
- `insert vertex/edge`：插入一条点或者边及其属性。
- `getProps`：获取一个点或者一条边的属性。

  正是这一层的存在，使得 Storage 服务变成了真正的图存储，否则 Storage 服务只是一个 KV 存储服务。

- Consensus 层

  Storage 服务的中间层，实现了 Multi Group Raft，保证强一致性和高可用性。

- Store Engine 层

  Storage 服务的最底层，是一个单机版本地存储引擎，提供对本地数据的 `get`、`put`、`scan` 等操作。相关接口存储在 `KVStore.h` 和 `KVEngine.h` 文件，用户可以根据业务需求定制开发相关的本地存储插件。

下文将基于架构介绍 Storage 服务的部分特性。

Storage 写入流程

KVStore

悦数图数据库使用自行开发的 KVStore，而不是其他开源 KVStore，原因如下：

- 需要高性能 KVStore。
- 需要以库的形式提供，实现高效计算下推。对于强 Schema 的悦数图数据库来说，计算下推时如何提供 Schema 信息，是高效的关键。
- 需要数据强一致性。

Copyright © Yueshu Graph Technology - All rights reserved.

基于上述原因，悦数图数据库使用 RocksDB 作为本地存储引擎，实现了自己的 KVStore，有如下优势：

- 对于多硬盘机器，悦数图数据库只需配置多个不同的数据目录即可充分利用多硬盘的并发能力。
- 由 Meta 服务统一管理所有 Storage 服务，可以根据所有分片的分布情况和状态，手动进行负载均衡。

> **Note**
>
> 不支持自动负载均衡是为了防止自动数据搬迁影响线上业务。

- 定制预写日志（WAL），每个分片都有自己的 WAL。
- 支持多个图空间，不同图空间相互隔离，每个图空间可以设置自己的分片数和副本数。

Copyright © Yueshu Graph Technology - All rights reserved.

数据存储格式

图存储的主要数据是点和边，悦数图数据库将点和边的信息存储为 key，同时将点和边的属性信息存储在 value 中，以便更高效地使用属性过滤。

- 点数据存储格式

  相比悦数图数据库 2.x 版本，3.x 版本在开启无 **Tag** 的点配置后，每个点多了一个不含 TagID 字段并且无 value 的 key。

字段	说明
`Type`	key 类型。长度为 1 字节。
`PartID`	数据分片编号。长度为 3 字节。此字段主要用于 Storage 负载均衡（balance）时方便根据前缀扫描整个分片的数据。
`VertexID`	点 ID。当点 ID 类型为 int 时，长度为 8 字节；当点 ID 类型为 string 时，长度为创建图空间时指定的 `fixed_string` 长度。
`TagID`	点关联的 Tag ID。长度为 4 字节。
`SerializedValue`	序列化的 value，用于保存点的属性信息。

Copyright © Yueshu Graph Technology - All rights reserved.

- 边数据存储格式

Type (1 byte)	PartID (3 bytes)	VertexID (n bytes)	EdgeType (4 bytes)	Rank (8 bytes)	VertexID (n bytes)	PlaceHolder (1 byte)	SerializedValue

字段	说明
`Type`	key 类型。长度为 1 字节。
`PartID`	数据分片编号。长度为 3 字节。此字段主要用于 Storage 负载均衡（balance）时方便根据前缀扫描整个分片的数据。
`VertexID`	点 ID。前一个 `VertexID` 在出边里表示起始点 ID，在入边里表示目的点 ID；后一个 `VertexID` 出边里表示目的点 ID，在入边里表示起始点 ID。
`Edge type`	边的类型。大于 0 表示出边，小于 0 表示入边。长度为 4 字节。
`Rank`	用来处理两点之间有多个同类型边的情况。用户可以根据自己的需求进行设置，例如存放交易时间、交易流水号等。长度为 8 字节，
`PlaceHolder`	预留字段。长度为 1 字节。
`SerializedValue`	序列化的 value，用于保存边的属性信息。

属性说明

悦数图数据库使用强类型 Schema。

对于点或边的属性信息，悦数图数据库会将属性信息编码后按顺序存储。由于定长属性的长度是固定的，查询时可以根据偏移量快速查询。在解码之前，需要先从 Meta 服务中查询具体的 Schema 信息（并缓存）。同时为了支持在线变更 Schema，在编码属性时，会加入对应的 Schema 版本信息。

数据分片

由于超大规模关系网络的节点数量高达百亿到千亿，而边的数量更会高达万亿，即使仅存储点和边两者也远大于一般服务器的容量。因此需要有方法将图元素切割，并存储在不同逻辑分片（Partition）上。悦数图数据库采用边分割的方式。

Copyright © Yueshu Graph Technology - All rights reserved.

切边与存储放大

悦数图数据库中逻辑上的一条边对应着硬盘上的两个键值对（key-value pair），在边的数量和属性较多时，存储放大现象较明显。边的存储方式如下图所示。

上图以最简单的两个点和一条边为例，起点 SrcVertex 通过边 EdgeA 连接目的点 DstVertex，形成路径 `(SrcVertex)-[EdgeA]->(DstVertex)`。这两个点和一条边会以 6 个键值对的形式保存在存储层的两个不同分片，即 Partition x 和 Partition y 中，详细说明如下：

- 点 SrcVertex 的键值保存在 Partition x 中。
- 边 EdgeA 的第一份键值，这里用 EdgeA_Out 表示，与 SrcVertex 一同保存在 Partition x 中。key 的字段有 Type、PartID（x）、VID（Src，即点 SrcVertex 的 ID）、EdgeType（符号为正，代表边方向为出）、Rank（0）、VID（Dst，即点 DstVertex 的 ID）和 PlaceHolder。SerializedValue 即 Value，是序列化的边属性。
- 点 DstVertex 的键值保存在 Partition y 中。
- 边 EdgeA 的第二份键值，这里用 EdgeA_In 表示，与 DstVertex 一同保存在 Partition y 中。key 的字段有 Type、PartID（y）、VID（Dst，即点 DstVertex 的 ID）、EdgeType（符号为负，代表边方向为入）、Rank（0）、VID（Src，即点 SrcVertex 的 ID）和 PlaceHolder。SerializedValue 即 Value，是序列化的边属性，与 EdgeA_Out 中该部分的完全相同。

EdgeA_Out 和 EdgeA_In 以方向相反的两条边的形式存在于存储层，二者组合成了逻辑上的一条边 EdgeA。EdgeA_Out 用于从起点开始的遍历请求，例如 `(a)-[]->()`；EdgeA_In 用于指向目的点的遍历请求，或者说从目的点开始，沿着边的方向逆序进行的遍历请求，例如例如 `()-[]->(a)`。

如 EdgeA_Out 和 EdgeA_In 一样，悦数图数据库冗余了存储每条边的信息，导致存储边所需的实际空间翻倍。因为边对应的 key 占用的硬盘空间较小，但 value 占用的空间与属性值的长度和数量成正比，所以，当边的属性值较大或数量较多时候，硬盘空间占用量会比较大。

分片算法

分片策略采用静态 **Hash** 的方式，即对点 VID 进行取模操作，同一个点的所有 Tag、出边和入边信息都会存储到同一个分片，这种方式极大地提升了查询效率。

Note

创建图空间时需指定分片数量，分片数量设置后无法修改，建议设置时提前满足业务将来的扩容需求。

多机集群部署时，分片分布在集群内的不同机器上。分片数量在 CREATE SPACE 语句中指定，此后不可更改。

如果需要将某些点放置在相同的分片（例如在一台机器上），可以参考公式或代码。

下文用简单代码说明 VID 和分片的关系。

Copyright © Yueshu Graph Technology - All rights reserved.

```
// 如果 ID 长度为 8，为了兼容 1.0，将数据类型视为 int64。
uint64_t vid = 0;
if (id.size() == 8) {
    memcpy(static_cast<void*>(&vid), id.data(), 8);
} else {
    MurmurHash2 hash;
    vid = hash(id.data());
}
PartitionID pId = vid % numParts + 1;
```

简单来说，上述代码是将一个固定的字符串进行哈希计算，转换成数据类型为 int64 的数字（int64 数字的哈希计算结果是数字本身），将数字取模，然后加 1，即：

```
pId = vid % numParts + 1;
```

示例的部分参数说明如下。

参数	说明
`%`	取模运算。
`numParts`	`VID` 所在图空间的分片数，即 CREATE SPACE 语句中的 `partition_num` 值。
`pId`	`VID` 所在分片的 ID。

例如有 100 个分片，`VID` 为 1、101 和 1001 的三个点将会存储在相同的分片。分片 ID 和机器地址之间的映射是随机的，所以不能假定任何两个分片位于同一台机器上。

Raft

关于 **RAFT** 的简单介绍

分布式系统中，同一份数据通常会有多个副本，这样即使少数副本发生故障，系统仍可正常运行。这就需要一定的技术手段来保证多个副本之间的一致性。

基本原理：Raft 就是一种用于保证多副本一致性的协议。Raft 采用多个副本之间竞选的方式，赢得"超过半数"副本投票的（候选）副本成为 Leader，由 Leader 代表所有副本对外提供服务；其他 Follower 作为备份。当该 Leader 出现异常后（通信故障、运维命令等），其余 Follower 进行新一轮选举，投票出一个新的 Leader。Leader 和 Follower 之间通过心跳的方式相互探测是否存活，并以 Raft-wal 的方式写入硬盘，超过多个心跳仍无响应的副本会被认为发生故障。

> **Note**
>
> 因为 Raft-wal 需要定期写硬盘，如果硬盘写能力瓶颈会导致 Raft 心跳失败，导致重新发起选举。硬盘 IO 严重堵塞情况下，会导致长期无法选举出 Leader。

读写流程：对于客户端的每个写入请求，Leader 会将该写入以 Raft-wal 的方式，将该条同步给其他 Follower，并只有在"超过半数"副本都成功收到 Raft-wal 后，才会返回客户端该写入成功。对于客户端的每个读取请求，都直接访问 Leader，而 Follower 并不参与读请求服务。

故障流程：场景 1：考虑一个配置为单副本（图空间）的集群；如果系统只有一个副本时，其自身就是 Leader；如果其发生故障，系统将完全不可用。场景 2：考虑一个配置为 3 副本（图空间）的集群；如果系统有 3 个副本，其中一个副本是 Leader，其他 2 个副本是 Follower；即使原 Leader 发生故障，剩下两个副本仍可投票出一个新的 Leader（以及一个 Follower），此时系统仍可使用；但是当这 2 个副本中任一者再次发生故障后，由于投票人数不足，系统将完全不可用。

> **Note**
>
> Raft 多副本的方式与 HDFS 多副本的方式是不同的，Raft 基于"多数派"投票，因此副本数量不能是偶数。

MULTI GROUP RAFT

由于 Storage 服务需要支持集群分布式架构，所以基于 Raft 协议实现了 Multi Group Raft，即每个分片的所有副本共同组成一个 Raft group，其中一个副本是 leader，其他副本是 follower，从而实现强一致性和高可用性。Raft 的部分实现如下。

Copyright © Yueshu Graph Technology - All rights reserved.

由于 Raft 日志不允许空洞，悦数图数据库使用 Multi Group Raft 缓解此问题，分片数量较多时，可以有效提高悦数图数据库的性能。但是分片数量太多会增加开销，例如 Raft group 内部存储的状态信息、WAL 文件，或者负载过低时的批量操作。

实现 Multi Group Raft 有 2 个关键点：

- 共享 Transport 层

  每一个 Raft group 内部都需要向对应的 peer 发送消息，如果不能共享 Transport 层，会导致连接的开销巨大。

- 共享线程池

  如果不共享一组线程池，会造成系统的线程数过多，导致大量的上下文切换开销。

批量（BATCH）操作

悦数图数据库中，每个分片都是串行写日志，为了提高吞吐，写日志时需要做批量操作，但是由于悦数图数据库利用 WAL 实现一些特殊功能，需要对批量操作进行分组，这是悦数图数据库的特色。

例如无锁 CAS 操作需要之前的 WAL 全部提交后才能执行，如果一个批量写入的 WAL 里包含了 CAS 类型的 WAL，就需要拆分成粒度更小的几个组，还要保证这几组 WAL 串行提交。

LEADER 切换（TRANSFER LEADERSHIP）

leader 切换对于负载均衡至关重要，当把某个分片从一台机器迁移到另一台机器时，首先会检查分片是不是 leader，如果是的话，需要先切换 leader，数据迁移完毕之后，通常还要重新均衡 leader 分布。

对于 leader 来说，提交 leader 切换命令时，就会放弃自己的 leader 身份，当 follower 收到 leader 切换命令时，就会发起选举。

成员变更

为了避免脑裂，当一个 Raft group 的成员发生变化时，需要有一个中间状态，该状态下新旧 group 的多数派需要有重叠的部分，这样就防止了新的 group 或旧的 group 单方面做出决定。为了更加简化，Diego Ongaro 在自己的博士论文中提出每次只增减一个 peer 的方式，以保证新旧 group 的多数派总是有重叠。悦数图数据库也采用了这个方式，只不过增加成员和移除成员的实现有所区别。具体实现方式请参见 Raft Part class 里 addPeer/removePeer 的实现。

与 HDFS 的区别

Storage 服务基于 Raft 协议实现的分布式架构，与 HDFS 的分布式架构有一些区别。例如：

- Storage 服务本身通过 Raft 协议保证一致性，副本数量通常为奇数，方便进行选举 leader，而 HDFS 存储具体数据的 DataNode 需要通过 NameNode 保证一致性，对副本数量没有要求。
- Storage 服务只有 leader 副本提供读写服务，而 HDFS 的所有副本都可以提供读写服务。
- Storage 服务无法修改副本数量，只能在创建图空间时指定副本数量，而 HDFS 可以调整副本数量。
- Storage 服务是直接访问文件系统，而 HDFS 的上层（例如 HBase）需要先访问 HDFS，再访问到文件系统，远程过程调用（RPC）次数更多。

总而言之，Storage 服务更加轻量级，精简了一些功能，架构没有 HDFS 复杂，可以有效提高小块存储的读写性能。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

3. 悦数 License

3.1 悦数 License

3.1.1 什么是悦数 License

悦数 License 指杭州悦数科技有限公司授予用户使用悦数图数据库及附属软件功能的合法权限，由用户联系悦数销售人员购买。

3.1.2 License Key

License Key 是悦数 License 的唯一标识。用户需要将 License Key 加载到 LM 中，悦数图数据库及附属软件会在启动时通过 LM 检查 License Key 的有效性。如果 License Key 有效，则悦数图数据库及附属软件功能正常使用，否则悦数图数据库及附属软件功能无法使用。License Key 由悦数销售人员根据用户提供的 LMID 生成。

3.1.3 授权流程

首次购买悦数 License 后，用户需要提供 LMID 给悦数销售人员以获取 License Key，然后在 LM 中加载 License Key。悦数图数据库及附属软件会在启动时通过 LM 检查 License Key 的有效性。

授权流程图

授权步骤

1. 联系悦数销售人员购买悦数 License。
2. 获取悦数图数库、附属软件，及 LM 的安装包。
3. 安装悦数图数库。
4. 安装 LM。
5. 在 LM 上查看 LMID。
6. 将 LMID 提供给悦数销售人员。
7. 悦数销售人员根据用户提供的 LMID 生成 License Key 并发送给用户。
8. 用户在 LM 上加载 License Key。
9. 在悦数图数库或附属软件中配置 LM 的地址。

Copyright © Yueshu Graph Technology - All rights reserved.

10. 启动悦数图数据库或附属软件。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

3.2 使用 License Manager

License Manager (LM) 是悦数提供给用户在终端侧管理 License 的服务，并且 License 的授权必须依赖于这项服务。通过 LM CLI 命令行工具，用户可在 LM 中加载 License Key 和查看 License 信息，包括 License 的有效期、购买的节点数等。在悦数图数据库及附属软件中配置 LM 地址后，可通过 LM 校验 License 的有效性，从而保证悦数图数据库及附属软件的正常使用。

本文介绍如何在 Linux 环境中部署 LM 及如何在悦数图数据库及附属软件中配置 LM。

3.2.1 前提条件

- 已获取悦数图数库、附属软件（如 Explorer、Dashboard），及 LM 的安装包。
- LM 默认使用 `9119` 端口，确保该端口未被占用。

3.2.2 使用说明

- 已购买 License。
- 由于 LM 是单进程服务，为了确保 LM 的可靠性和连续性，建议使用 Systemd 来管理 LM，并为其设置一个重启策略。
- LM 服务器的时间必需与连接 LM 的产品（包括悦数图数据库和附属软件）服务器的时间保持一致。如果时间不一致，许可证验证将失败，从而导致无法使用产品。

3.2.3 安装和启动 LM

支持在 Linux amd64 或者 arm64 系统上安装和启动 LM 和使用悦数运维监控安装和启动 LM。详情见下文。

TAR 包

1. 解压 LM 的 TAR 包：

```
tar -zxvf <name.tar.gz> -C <path>
```

- `<name.tar.gz>`：LM 的 TAR 包名称。
- `<path>`：指定 LM 包的解压目录。不指定 `-C` 参数时，默认解压到当前目录。

2. 使用 Systemd 启动 LM 服务。

a. 创建 LM 服务文件 `/etc/systemd/system/yueshu-license-manager.service`，内容如下：

```
[Unit]
Description=License Manager

[Service]
Type=simple
ExecStart=<path>/yueshu-license-manager/yueshu-license-manager
WorkingDirectory=<path>/yueshu-license-manager
Restart=always

[Install]
WantedBy=multi-user.target
```

- `<path>`：指 LM 包的解压目录。

b. 启动 LM 服务：

```
sudo systemctl start yueshu-license-manager
```

3. 设置 LM 开机自启动：

```
sudo systemctl enable yueshu-license-manager
```


Copyright © Yueshu Graph Technology - All rights reserved.

RPM 包

1. 解压 LM 的 RPM 包：

```
sudo rpm -ivh <name.rpm>
```

- `<name.rpm>`： LM 的 RPM 包名称。
- 解压路径为 `/usr/local/yueshu-license-manager`， 不可更改。

2. 启动 LM：

```
sudo systemctl start yueshu-license-manager
```

3. 设置 LM 开机自启动：

```
sudo systemctl enable yueshu-license-manager
```

DEB 包

1. 解压 LM 的 DEB 包：

```
sudo dpkg -i <name.deb>
```

- `<name.deb>`： LM 的 DEB 包名称。
- 解压路径为 `/usr/local/yueshu-license-manager`， 不可更改。

2. 启动 LM：

```
sudo systemctl start yueshu-license-manager
```

3. 设置 LM 开机自启动：

```
sudo systemctl enable yueshu-license-manager
```

使用悦数运维监控

悦数运维监控支持快捷安装和启动 LM。详情参见使用悦数运维监控部署 LM。

3.2.4 查看 LM 配置文件

LM 的配置文件名为 `yueshu-license-manager.yaml`。

- RPM 包和 DEB 包的默认路径为 `/usr/local/yueshu-license-manager/etc/yueshu-license-manager.yaml`。
- TAR 包的路径为解压目录下的 `yueshu-license-manager/etc/yueshu-license-manager.yaml`。

LM 配置文件的内容如下：

```
Name: nebula-license-manager
Host: 0.0.0.0                       # LM 绑定的主机地址。
Port: 9119                          # LM 监听的端口号。默认为 9119。
Timeout: 3000                       # LM 等待客户端请求时的超时时间。单位为毫秒，默认为 3000 毫秒。
DataPath: data                      # 存储 LM 状态数据的路径。
Notify:                             # LM 通知相关配置。
  Mail:                             # 邮件通知相关配置。
    Host: ""                        # SMTP 服务器地址。
    Port: 465                       # SMTP 服务器端口号。
    User: ""                        # SMTP 邮箱。
    Password: ""                    # SMTP 邮箱密码。
    To: []                          # 接收通知的邮箱列表。
Log:                                # 日志相关配置。
  Mode: file                        # 日志模式。可选值为 file（输出到文件）和 console（输出到控制台）。默认为 file。
  Path: logs                        # 存储日志文件的路径。
  Level: info                       # 日志级别。可选值为 debug、info、error、severe。默认为 info。
  KeepDays: 30                      # 日志最长保留天数。默认为 30 天。
```


Copyright © Yueshu Graph Technology - All rights reserved.

3.2.5 使用 LM

LM 启动后，可在 LM 的安装路径下通过 LM CLI 工具查看 License 信息。

查看 **LM CLI** 版本

```
./yueshu-license-manager-cli version
```

> **Note**
>
> LM 的版本信息在 LM 启动时，会在日志中打印。

查看 **License** 基本信息

```
./yueshu-license-manager-cli info
```

- 未加载 License Key 时，输出如下：

```
          LMID:  RUZB-XXXX
 LicenseStatus:  NotExist
```

- 已加载 License Key 时，输出如下：

```
          LMID:  RUZB-XXXX
 LicenseStatus:  Normal
    LicenseKey:  MM9OU-9H4QO-WO93M-XXXXX-XXXXX
          Type:  NODE
    Query Node:  3
  Storage Node:  3
      ExpireAt:  2023-06-25 12:00:00 +0800 CST
```

字段说明：

字段	说明
`LMID`	LM 的 ID。获取 License Key 时，需要绑定该 LMID。
`LicenseStatus`	License 的状态。包括： `Normal`： 可正常使用 License。 `NotExist`： License Key 不存在。 `Invalid`： License Key 无效。 `Expiring`： License 即将过期。 `Expired`： License 已过期。
`LicenseKey`	一个包含授权信息的加密字符串，是用户获得悦数图数据库及附属软件功能授权的唯一凭证。
`Type`	购买的资源类型。目前支持购买 `NODE` （节点）和 `CPU` 类型资源。
`Query Node`	购买的查询节点的数量
`Storage Node`	购买的存储节点数量
`ExpireAt`	License 的过期时间。

加载 **License Key**

从销售获取 License Key 后，需要通过 LM CLI 工具加载 License Key。

```
./yueshu-license-manager-cli load --key <license-key> --force
```

- `<license-key>`： License Key 字符串。
- `--force`： 强制加载 License Key。如果不指定该参数，LM 将校验当前 License 的状态。

Copyright © Yueshu Graph Technology - All rights reserved.

查看 **License** 配额使用情况

查看当前的 License 配额（节点数）使用情况，及附属软件的使用状态。

```
./yueshu-license-manager-cli usage
```

查看指定 **LM** 上的 **License** 信息

如果有多个 LM，可以通过以下命令查看指定 LM 上的 License 信息。

```
./yueshu-license-manager-cli <command> --addr <host>:9119
```

- `<command>`：可选值为 `info` 、 `usage` 、 `sync` 、 `load --key <license-key> --force` 。
- `<host>`：指定的 LM 所在的主机 IP 地址。

3.2.6 监控 LM

监控 **LM** 状态

用户可使用监控工具监控 LM 是否处于正常运行的状态。

- 使用悦数运维监控

  当 LM 正常运行时，悦数运维监控许可证管理页面显示 LM 的状态为运行中，否则为已退出。详情参见悦数运维监控许可证管理。

- 使用 Prometheus

  当使用 Prometheus 监控 LM 的状态时，需在 Prometheus 的配置文件中添加设置以下参数。有关 Prometheus 配置文件的更多信息，请参见 Prometheus 官方文档。

  ```
  ...
  - job_name: license-manager
    scrape_interval: 15s                # 拉取数据的时间间隔。
    metrics_path: /metrics              # LM 的监控指标路径，为`/metrics`。
    scheme: http                        # LM 的协议类型，为 HTTP。
    static_configs:                     # LM 的地址和端口（默认 9119）。
    - targets:
      - [<ip:lm_port>]
  ...
  ```

  然后，在 Prometheus 中，可通过其内置指标 `up` 监控 LM 的状态。如果 LM 正常运行， `up` 的值为 `1` ，否则为 `0` 。查看 `up` 指标的状态，命令如下：

  ```
  up{instance="<ip:lm_port>", job="<job_name>"}
  ```

- `<ip:lm_port>`：LM 所在的主机 IP 和端口号。
- `<job_name>`：LM 的作业名称。

  例如， `up{instance="192.168.8.xxx:9119", job="license-manager"}` 。

  > **Note**
  >
  > 默认情况下，LM 使用 `9119` 端口。如果需要修改端口号，可在上文的配置文件中修改 `Port` 字段的值。

查看 **LM** 的监控指标

可通过 `http://<ip:lm_port>/metrics` 方式，例如 `http://192.168.8.xxx:9119/metrics` ，查看 LM 的所有监控指标。

同时，可使用监控工具收集 LM 的指标数据。如果使用 Prometheus，需在 Prometheus 的配置文件中添加参数，详情参见上文监控 **LM** 状态。

Copyright © Yueshu Graph Technology - All rights reserved.

3.2.7 配置 LM 的连接

悦数图数据库

在悦数图数据库 Meta 服务的配置文件（`nebula-metad.conf`）中，设置 `license_manager_url` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.xxx:9119`。详情参见 Meta 服务配置。

配置完成后，重启 Meta 服务。

悦数图探索（**Explorer**）

在 Explorer 安装路径下，进入 `config` 目录，修改 `app-config.yaml` 文件，设置 `LicenseManagerURL` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.xxx:9119`。

配置完成后，重启 Explorer。详情参见部署 Explorer。

悦数运维监控（**Dashboard**）

在 Dashboard 安装路径下，进入 `etc` 目录，修改 `config.yaml` 文件，设置 `LicenseManagerURL` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.xxx:9119`。

配置完成后，重启 Dashboard。详情参见部署 Dashboard。

悦数图计算（**Analytics**）

在 Analytics 安装路径下，修改 `scripts` 目录中的 `analytics.conf` 文件，设置 `license_manager_url` 值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.xxx:9119`。

配置完成后，在 `scripts` 目录下执行 `./run_pagerank.sh` 重启 Analytics。更多信息，参见 NebulaGraph Analytics。

自动化部署运维（**Operator**）

- 使用 Kubectl 部署集群时，在集群配置文件中，通过 `spec.metad.licenseManagerURL` 字段配置 LM 的访问地址和端口。详情参见使用 Kubectl 部署。
- 使用 Helm 部署集群时，通过 `--set nebula.metad.licenseManagerURL` 指定 LM 的地址和端口。详情参见使用 Helm 部署。

3.2.8 FAQ

问：我可以更换 LM 所在的主机吗?

答：不行。如果您需要更改主机，或者主机无法使用，则需要联系销售团队重新绑定 LMID。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4. 快速入门

4.1 从云开始

4.1.1 创建云服务实例

用户可以在云服务上创建悦数图数据库实例，本文介绍如何在阿里云上快速创建悦数图数据库实例。

背景信息

悦数图数据库支持在多个云平台上部署悦数图数据库，本文只介绍如何在悦数图数据库（阿里云版）创建悦数图数据库实例。更多云服务平台的支持情况，参见悦数图数据库（公有云）。

Caution

使用悦数图数据库（阿里云版）创建的实例对应 `3.4.0` 版本的悦数图数据库。

账号与权限

- 准备阿里云账号。
- 如果使用 RAM 用户创建实例，需为其添加以下权限：
- AliyunECSFullAccess
- AliyunVPCFullAccess
- AliyunROSFullAccess
- AliyunCloudMonitorFullAccess
- AliyunComputeNestUserFullAccess

资源与费用

NebulaGraph Cloud 阿里云版支持免费试用和付费使用，二者的详细说明如下表。

项目	免费试用版	付费版
云资源归属	阿里云官方账号	用户账号
云资源费用	阿里云承担	用户承担
云资源使用时长	30 天（不可续期）	用户创建实例时选择
悦数图数据库许可证费用	试用期内免费	用户承担
悦数图数据库服务许可证有效期	30 天	用户创建实例时选择
许可证过期后数据是否保留	否（因云资源会同时到期）	是
云资源到期后数据是否保留	否	否
数据盘快照费用	阿里云承担	用户承担（可关闭功能）

数据备份

在创建服务实例时，系统会默认开启数据盘自动快照备份，用于周期性备份悦数图数据库数据。

创建付费版服务实例前需开通快照。免费试用实例使用的是阿里云官方账号下的资源，已开通快照，无需用户手动开通。

Copyright © Yueshu Graph Technology - All rights reserved.

默认的快照备份策略为每天 00:00 创建快照。

创建快照会产生费用，详情参见快照计费。

如需删除快照、修改或删除自动快照策略，参见快照文档。

套餐版本

NebulaGraph Cloud 阿里云版支持如下套餐版本。

套餐版本	说明
基础版	将所有悦数图数据库服务节点部署在 1 台 ECS 服务器上。
标准版	将 Graph 服务和 Storage 服务分别部署在不同的 ECS 服务器上，每个服务都是单节点（1 台 ECS）。将悦数图探索等生态工具混合部署在 1 台 ECS 服务器上。
高可用版	将 Graph 服务和 Storage 服务分别部署在不同的 ECS 服务器上，每个服务都包含 3 节点（3 台 ECS）。将悦数图探索等生态工具混合部署在 1 台 ECS 服务器上。

付费版和免费试用版服务实例支持的套餐版本有所不同，详情参见创建实例页面。

创建付费版服务实例

1. 登录阿里云控制台。
2. 打开云市场的 悦数图数据库产品页。
3. 选择套餐版本和购买时长（即悦数图数据库服务的许可证有效期），并单击立即购买。

Caution

当前云市场显示的部分价格并非实际售价，购买时需通过右侧的钉钉客服确认实际价格。

4. 在创建服务实例页，保持选择模板处的选择不变。如需切换部署架构，可改变选中的选项，重新选择模板。

Note

改变模板会改变之前选择的套餐版本，软件费用（悦数图数据库许可证费用）和创建服务实例需要的资源也会改变。

5. （可选）设置服务实例名称。默认值为服务实例 ID。
6. 选择要创建服务实例的地域。
7. 在付费模式设置区域，指定 ECS 服务器的付费方式。默认为按量付费。

- 按量付费：按照计费周期计费，在每个结算周期生成账单并从账户中扣除相应费用。详情参见按量付费。
- 包年包月：先付费后使用。详情参见包年包月。选择包年包月模式需要指定购买时长周期和购买时长。购买时长周期当前仅支持 **Month**，即按月购买。

8. 在 悦数图数据库配置区域，完成数据盘和 ECS 服务器密码设置。
9. 在基础设施配置区域，完成以下设置。

a. 选择专有网络 **VPC** 实例 **ID**。

如果下拉列表为空，先单击其右侧的新建专有网络，完成专有网络创建。

b. 选择交换机可用区。

c. 选择业务网络交换机的实例 **ID**。

如果下拉列表为空，先单击其右侧的新建交换机，完成交换机创建。

Copyright © Yueshu Graph Technology - All rights reserved.

10. 完成权限确认，并选中我同意授权服务商（杭州悦数科技有限公司）获取上述权限以提供代运维服务。
11. 在页面底部，单击下一步：确认订单。
12. 在服务条款区域，勾选我已阅读并同意《商品在线协议》《云市场平台服务协议》《计算巢服务协议》。
13. 在页面底部，单击立即支付。
14. 在支付页面，选择支付方式，之后单击页面底部的支付。
15. 在支付完成页面，单击管理控制台。
16. 在已购买的服务页面，单击要创建的实例右侧的立即进入计算巢部署或查看部署详情。

> **Note**
>
> 如果已购买的服务列表中未显示要创建的实例，刷新页面。

17. 在服务实例详情页查看实例的状态，确保状态为部署中。

    部署的平均耗时为 10 分钟。完成后实例的状态变为已部署。

创建免费版试用版服务实例

> **Danger**
>
> 免费试用实例不可在生产环境中使用。

1. 登录阿里云计算巢推荐服务页面，搜索 **NebulaGraph**。
2. 在 悦数图数据库集群版卡片上，单击免费试用。
3. 系统会为首次创建悦数图数据库服务实例的用户弹出申请对话框。在申请权限对话框中填写申请人信息。带有红色星号（*）的为必填项。

> **Note**
>
> 申请通过后需从第 1 步重新开始。

4. （可选）设置服务实例名称。默认值为服务实例 ID。
5. 选择要创建实例的地域。
6. 在 悦数图数据库配置区域，完成数据盘和 ECS 服务器密码设置。
7. 在基础设施配置区域，选择交换机可用区。
8. 完成权限确认，并选中我同意授权服务商（杭州悦数科技有限公司）获取上述权限以提供代运维服务。
9. 在页面底部，单击下一步：确认订单。
10. 在服务条款区域，勾选我已阅读并同意《计算巢服务协议》。
11. 在页面底部，单击开始免费试用。
12. 在提交成功页面，单击去列表查看。
13. 在实例列表中查看目标实例的状态，确保状态为部署中。

    部署的平均耗时为 10 分钟。完成后实例的状态变为已部署。

Copyright © Yueshu Graph Technology - All rights reserved.

常见问题

Q：服务实例的状态显示为部署失败怎么处理?

1. 如果使用 RAM 账号创建的实例，确认为该账号授予了本文前提条件中指定的权限。
2. 如果权限符合要求，删除创建失败的实例，尝试重新创建。
3. 如果仍然创建失败，到 悦数图数据库论坛寻求帮助。

下一步

连接 NebulaGraph

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.1.2 连接悦数图数据库

创建好云服务实例后，可以连接悦数图数据库。本文介绍如何使用悦数图探索快速连接悦数图数据库。

前提条件

创建云服务实例

操作步骤

本文使用 悦数图探索 快速连接悦数图数据库。操作步骤如下：

1. 登录服务实例管理页面。
2. 查看已创建的实例。
- 对于正式版实例，单击私有部署服务页签。
- 对于免费试用版实例，单击试用服务页签。
3. 在实例列表中，单击目标实例的服务实例**ID**，或其右侧操作列的详情。
4. 在概览页签的基本信息区域，查看**nebula_private_ip**及**explorer_portal**信息。
5. 单击**explorer_portal**对应链接，进入悦数图探索登录页面。
6. 填写登录信息，单击登录。
- **Host**：`nebula_private_ip地址:9669`，例如`192.168.98.160:9669`。
- 用户名：`root`。
- 密码：任意密码。

> **Note**
>
> 用户可修改登录用户名及密码，详情参见用户管理。

更多连接方式

用户可根据自己的需求选择其他连接方式。关于更多连接方式，参见连接悦数图数据库。

下一步

管理悦数图数据库数据

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.1.3 管理悦数图数据库数据

nGQL 是悦数图数据库创建的声明式图查询语言。用户可以使用 nGQL 语句对悦数图数据库数据库进行增删改查。本文介绍如何通过悦数图数据库（阿里云版）中悦数图探索的控制台功能快速使用悦数图数据库基本语句。

Note

用户可以在悦数图探索中通过可视化查询悦数图数据库数据。本文介绍通过控制台功能使用 nGQL 语句增删改查悦数图数据库数据。关于如何可视化使用悦数图探索查询数据，参见开始探索和探索拓展，或者观看视频。

前提条件

连接悦数图数据库

基本流程

操作入口

在悦数图探索页面的右上方，单击进入控制台页面。在控制台命令行中执行 nGQL 语句。

使用说明

图空间和 SCHEMA

一个悦数图数据库实例由一个或多个图空间组成。每个图空间都是物理隔离的，用户可以在同一个实例中使用不同的图空间存储不同的数据集。

Copyright © Yueshu Graph Technology - All rights reserved.

为了在图空间中插入数据，需要为图数据库定义一个 Schema。悦数图数据库的 Schema 是由如下几部分组成。

组成部分	说明
点（Vertex）	表示现实世界中的实体。一个点可以有 0 到多个标签。
标签（Tag）	点的类型，定义了一组描述点类型的属性。
边（Edge）	表示两个点之间有方向的关系。
边类型（Edge type）	边的类型，定义了一组描述边的类型的属性。

更多信息，请参见数据结构。

本文将使用下图的数据集演示基础操作的语法。

Copyright © Yueshu Graph Technology - All rights reserved.

异步实现创建和修改

Caution

在悦数图数据库中，下列创建和修改操作是异步实现的。要在下一个心跳周期之后才能生效，否则访问会报错。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。

- `CREATE SPACE`
- `CREATE TAG`
- `CREATE EDGE`
- `ALTER TAG`
- `ALTER EDGE`
- `CREATE TAG INDEX`
- `CREATE EDGE INDEX`

Note

默认心跳周期是 10 秒。修改心跳周期参数 `heartbeat_interval_secs`，请参见配置简介。

第一步：创建和选择图空间

NGQL 语法

- 创建图空间

```
CREATE SPACE [IF NOT EXISTS] <graph_space_name> (
[partition_num = <partition_number>,]
[replica_factor = <replica_number>,]
vid_type = {FIXED_STRING(<N>) | INT64}
)

[COMMENT = '<comment>'];
```

参数详情请参见 CREATE SPACE。

- 列出创建成功的图空间

```
nebula> SHOW SPACES;
```

- 选择数据库

```
USE <graph_space_name>;
```

示例

1. 执行如下语句创建名为 `basketballplayer` 的图空间。

```
nebula> CREATE SPACE basketballplayer(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
```

Note

如果报错提示 `[ERROR (-1005)]: Host not enough!`，请检查是否已添加 Storage 主机。

2. 执行命令 `SHOW HOSTS` 检查分片的分布情况，确保平衡分布。

```
nebula> SHOW HOSTS;
+-------------+-----------+-----------+--------------+---------------------------------+------------------------+---------+
| Host        | Port      | Status    | Leader count | Leader distribution             | Partition distribution | Version |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+------------+-----------+-----------+-------------+-------------------------------+-----------------------+---------+
| "storaged0" | 9779      | "ONLINE"  | 5           | "basketballplayer:5"          | "basketballplayer:5"  | "3.5.1" |
| "storaged1" | 9779      | "ONLINE"  | 5           | "basketballplayer:5"          | "basketballplayer:5"  | "3.5.1" |
| "storaged2" | 9779      | "ONLINE"  | 5           | "basketballplayer:5"          | "basketballplayer:5"  | "3.5.1" |
+------------+-----------+-----------+-------------+-------------------------------+-----------------------+---------+
```

如果 **Leader distribution** 分布不均匀，请执行命令 `BALANCE LEADER` 重新分配。更多信息，请参见 Storage 负载均衡。

3. 选择图空间 `basketballplayer` 。

```
nebula[(none)]> USE basketballplayer;
```

用户可以执行命令 `SHOW SPACES` 查看创建的图空间。

```
nebula> SHOW SPACES;
+--------------------+
| Name               |
+--------------------+
| "basketballplayer" |
+--------------------+
```

第二步：创建 **Tag** 和 **Edge type**

NGQL 语法

```
CREATE {TAG | EDGE} [IF NOT EXISTS] {<tag_name> | <edge_type_name>}
    (
      <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']
      [{, <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']} ...]
    )
    [TTL_DURATION = <ttl_duration>]
    [TTL_COL = <prop_name>]
    [COMMENT = '<comment>'];
```

参数详情请参见 CREATE TAG 和 CREATE EDGE。

示例

创建 Tag: `player` 和 `team` ，以及 Edge type: `follow` 和 `serve` 。说明如下表。

名称	类型	属性
player	Tag	name (string), age (int)
team	Tag	name (string)
follow	Edge type	degree (int)
serve	Edge type	start_year (int), end_year (int)

```
nebula> CREATE TAG player(name string, age int);

nebula> CREATE TAG team(name string);

nebula> CREATE EDGE follow(degree int);

nebula> CREATE EDGE serve(start_year int, end_year int);
```

第三步：插入数据

用户可以使用 `INSERT` 语句，基于现有的 Tag 插入点，或者基于现有的 Edge type 插入边。

NGQL 语法

- 插入点

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES <vid>: ([prop_value_list])

tag_props:
  tag_name ([prop_name_list])

prop_name_list:
   [prop_name [, prop_name] ...]
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
prop_value_list:
   [prop_value [, prop_value] ...]
```

`vid` 是 Vertex ID 的缩写，`vid` 在一个图空间中是唯一的。参数详情请参见 INSERT VERTEX。

- 插入边

```
INSERT EDGE [IF NOT EXISTS] <edge_type> ( <prop_name_list> ) VALUES
<src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> )
[, <src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> ), ...];

<prop_name_list> ::=
[ <prop_name> [, <prop_name> ] ...]

<prop_value_list> ::=
[ <prop_value> [, <prop_value> ] ...]
```

参数详情请参见 INSERT EDGE。

示例

- 插入代表球员和球队的点。

```
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);

nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);

nebula> INSERT VERTEX player(name, age) VALUES "player102":("LaMarcus Aldridge", 33);

nebula> INSERT VERTEX team(name) VALUES "team203":("Trail Blazers"), "team204":("Spurs");
```

- 插入代表球员和球队之间关系的边。

```
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);

nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player102":(90);

nebula> INSERT EDGE follow(degree) VALUES "player102" -> "player100":(75);

nebula> INSERT EDGE serve(start_year, end_year) VALUES "player101" -> "team204":(1999, 2018),"player102" -> "team203":(2006,  2015);
```

第四步：查询数据

- GO 语句可以根据指定的条件遍历数据库。`GO` 语句从一个或多个点开始，沿着一条或多条边遍历，返回 `YIELD` 子句中指定的信息。
- FETCH 语句可以获得点或边的属性。
- LOOKUP 语句是基于索引的，和 `WHERE` 子句一起使用，查找符合特定条件的数据。
- MATCH 语句是查询图数据最常用的，可以灵活的描述各种图模式，但是它依赖索引去匹配悦数图数据库中的数据模型，性能也还需要调优。

NGQL 语法

- `GO`

```
GO [[<M> TO] <N> {STEP|STEPS}] FROM <vertex_list>
OVER <edge_type_list> [{REVERSELY | BIDIRECT}]
[ WHERE <conditions> ]
YIELD [DISTINCT] <return_list>
[{ SAMPLE <sample_list> | <limit_by_list_clause> }]
[| GROUP BY {<col_name> | expression> | <position>} YIELD <col_name>]
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
[| ORDER BY <expression> [{ASC | DESC}]]
[| LIMIT [<offset>,] <number_rows>];
```

- `FETCH`
- 查询 Tag 属性

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *}
<vid> [, vid ...]
YIELD <return_list> [AS <alias>];
```

- 查询边属性

```
FETCH PROP ON <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid> ...]
YIELD <output>;
```

- `LOOKUP`

```
LOOKUP ON {<vertex_tag> | <edge_type>}
[WHERE <expression> [AND <expression> ...]]
YIELD <return_list> [AS <alias>];

<return_list>
    <prop_name> [AS <col_alias>] [, <prop_name> [AS <prop_alias>] ...];
```

- `MATCH`

```
MATCH <pattern> [<clause_1>]  RETURN <output>  [<clause_2>];
```

`GO` 语句示例

- 从 VID 为 `player101` 的球员开始，沿着边 `follow` 找到连接的球员。

```
nebula> GO FROM "player101" OVER follow YIELD id($$);
+-------------+
| id($$)      |
+-------------+
| "player100" |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "player102" |
+-------------+
```

- 从 VID 为 `player101` 的球员开始，沿着边 `follow` 查找年龄大于或等于 35 岁的球员，并返回他们的姓名和年龄，同时重命名对应的列。

```
nebula> GO FROM "player101" OVER follow WHERE properties($$).age >= 35 \
        YIELD properties($$).name AS Teammate, properties($$).age AS Age;
+-----------------+-----+
| Teammate        | Age |
+-----------------+-----+
| "Tim Duncan"    | 42  |
+-----------------+-----+
```

子句/符号	说明
`YIELD`	指定该查询需要返回的值或结果。
`$$`	表示边的终点。
`\`	表示换行继续输入。

- 从 VID 为 `player101` 的球员开始，沿着边 `follow` 查找连接的球员，然后检索这些球员的球队。为了合并这两个查询请求，可以使用管道符或临时变量。

- 使用管道符

```
nebula> GO FROM "player101" OVER follow YIELD dst(edge) AS id | \
        GO FROM $-.id OVER serve YIELD properties($$).name AS Team, \
        properties($^).name AS Player;
+----------------+--------------------+
| Team           | Player             |
+----------------+--------------------+
| "Trail Blazers" | "LaMarcus Aldridge" |
+----------------+--------------------+
```

子句/符号	说明
`$^`	表示边的起点。
`|`	组合多个查询的管道符，将前一个查询的结果集用于后一个查询。
`$-`	表示管道符前面的查询输出的结果集。

- 使用临时变量

> **Note**
>
> 当复合语句作为一个整体提交给服务器时，其中的临时变量会在语句结束时被释放。

```
nebula> $var = GO FROM "player101" OVER follow YIELD dst(edge) AS id; \
        GO FROM $var.id OVER serve YIELD properties($$).name AS Team, \
        properties($^).name AS Player;
+----------------+--------------------+
| Team           | Player             |
+----------------+--------------------+
| "Trail Blazers" | "LaMarcus Aldridge" |
+----------------+--------------------+
```

`FETCH` 语句示例

查询 VID 为 `player100` 的球员的属性。

```
nebula> FETCH PROP ON player "player100" YIELD properties(vertex);
+------------------------------+
| properties(VERTEX)           |
+------------------------------+
| {age: 42, name: "Tim Duncan"} |
+------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

Note

`LOOKUP` 和 `MATCH` 的示例在下文的索引 部分查看。

其他操作

修改点和边

用户可以使用 `UPDATE` 语句或 `UPSERT` 语句修改现有数据。

`UPSERT` 是 `UPDATE` 和 `INSERT` 的结合体。当使用 `UPSERT` 更新一个点或边，如果它不存在，数据库会自动插入一个新的点或边。

Note

每个 partition 内部，`UPSERT` 操作是一个串行操作，所以执行速度比执行 `INSERT` 或 `UPDATE` 慢很多。其仅在多个 partition 之间有并发。

nGQL 语法

- `UPDATE` 点

```
UPDATE VERTEX <vid> SET <properties to be updated>
[WHEN <condition>] [YIELD <columns>];
```

- `UPDATE` 边

```
UPDATE EDGE ON <edge_type> <source vid> -> <destination vid> [@rank]
SET <properties to be updated> [WHEN <condition>] [YIELD <columns to be output>];
```

- `UPSERT` 点或边

```
UPSERT {VERTEX <vid> | EDGE <edge_type>} SET <update_columns>
[WHEN <condition>] [YIELD <columns>];
```

示例

- 用 `UPDATE` 修改 VID 为 `player100` 的球员的 `name` 属性，然后用 `FETCH` 语句检查结果。

```
nebula> UPDATE VERTEX "player100" SET player.name = "Tim";

nebula> FETCH PROP ON player "player100" YIELD properties(vertex);
+------------------------+
| properties(VERTEX)     |
+------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| {age: 42, name: "Tim"} |
+------------------------+
```

- 用 `UPDATE` 修改某条边的 `degree` 属性，然后用 `FETCH` 检查结果。

```
nebula> UPDATE EDGE ON follow "player101" -> "player100" SET degree = 96;

nebula> FETCH PROP ON follow "player101" -> "player100" YIELD properties(edge);
+------------------+
| properties(EDGE) |
+------------------+
| {degree: 96}     |
+------------------+
```

- 用 `INSERT` 插入一个 VID 为 `player111` 的点，然后用 `UPSERT` 更新它。

```
nebula> INSERT VERTEX player(name,age) VALUES "player111":("David West", 38);

nebula> UPSERT VERTEX "player111" SET player.name = "David", player.age = $^.player.age + 11 \
        WHEN $^.player.name == "David West" AND $^.player.age > 20 \
        YIELD $^.player.name AS Name, $^.player.age AS Age;
+---------+-----+
| Name    | Age |
+---------+-----+
| "David" | 49  |
+---------+-----+
```

删除点和边

nGQL 语法

- 删除点

```
DELETE VERTEX <vid1>[, <vid2>...]
```

- 删除边

```
DELETE EDGE <edge_type> <src_vid> -> <dst_vid>[@<rank>]
[, <src_vid> -> <dst_vid>...]
```

示例

- 删除点

```
nebula> DELETE VERTEX "player111", "team203";
```

- 删除边

```
nebula> DELETE EDGE follow "player101" -> "team204";
```

使用索引

用户可以通过 CREATE INDEX 语句为 Tag 和 Edge type 增加索引。

> 使用索引必读
>
> `MATCH` 和 `LOOKUP` 语句的执行都依赖索引，但是索引会导致写性能大幅降低。请不要随意在生产环境中使用索引，除非很清楚使用索引对业务的影响。
>
> 必须为“已写入但未构建索引”的数据重建索引，否则无法在 `MATCH` 和 `LOOKUP` 语句中返回这些数据。参见重建索引。

nGQL 语法

- 创建索引

Copyright © Yueshu Graph Technology - All rights reserved.

```
CREATE {TAG | EDGE} INDEX [IF NOT EXISTS] <index_name>
ON {<tag_name> | <edge_name>} ([<prop_name_list>]) [COMMENT = '<comment>'];
```

- 重建索引

```
REBUILD {TAG | EDGE} INDEX <index_name>;
```

Note

为没有指定长度的变量属性创建索引时，需要指定索引长度。在 utf-8 编码中，一个中文字符占 3 字节，请根据变量属性长度设置合适的索引长度。例如 10 个中文字符，索引长度需要为 30。详情请参见创建索引。

基于索引的 **LOOKUP** 和 **MATCH** 示例

确保 `LOOKUP` 或 `MATCH` 有一个索引可用。如果没有，请先创建索引。

找到 Tag 为 `player` 的点的信息，它的 `name` 属性值为 `Tony Parker` 。

```
// 为 name 属性创建索引 player_index_1。
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_1 ON player(name(20));

// 重建索引确保能对已存在数据生效。
nebula> REBUILD TAG INDEX player_index_1
+------------+
| New Job Id |
+------------+
| 31         |
+------------+

// 使用 LOOKUP 语句检索点的属性。
nebula> LOOKUP ON player WHERE player.name == "Tony Parker" \
        YIELD properties(vertex).name AS name, properties(vertex).age AS age;
+---------------+-----+
| name          | age |
+---------------+-----+
| "Tony Parker" | 36  |
+---------------+-----+

// 使用 MATCH 语句检索点的属性。
nebula> MATCH (v:player{name:"Tony Parker"}) RETURN v;
+----------------------------------------------------+
| v                                                  |
+----------------------------------------------------+
| ("player101" :player{age: 36, name: "Tony Parker"}) |
+----------------------------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.2 本地部署

4.2.1 步骤 1：安装悦数图数据库

RPM 和 DEB 是 Linux 系统下常见的两种安装包格式，本文介绍如何使用 RPM 或 DEB 文件在一台机器上快速安装悦数图数据库。

> **Note**
>
> 部署悦数图数据库集群的方式参见使用 RPM/DEB 包部署集群。

前提条件

- 安装 `wget` 工具。
- 已在 LM 中加载 License Key。

下载安装包

联系我们获取悦数图数据库安装包。

> **Note**
>
> - 当前仅支持在 Linux 系统下安装悦数图数据库，且仅支持 CentOS 7.x、CentOS 8.x、Ubuntu 16.04、Ubuntu 18.04、Ubuntu 20.04 操作系统。
> - 如果用户使用的是国产化的 Linux 操作系统，请安装企业版悦数图数据库。

安装悦数图数据库

- 安装 RPM 包

```
$ sudo rpm -ivh --prefix=<installation_path> <package_name>
```

`--prefix` 为可选项，用于指定安装路径。如不设置，系统会将悦数图数据库安装到默认路径 `/usr/local/nebula/` 。

例如，要在默认路径下安装3.5.1版本的 RPM 包，运行如下命令：

```
sudo rpm -ivh yueshu-graph-3.5.1.el7.x86_64.rpm
```

- 安装 DEB 包

```
$ sudo dpkg -i <package_name>
```

> **Note**
>
> 使用 DEB 包安装悦数图数据库时不支持自定义安装路径。默认安装路径为 `/usr/local/nebula/` 。

例如安装3.5.1版本的 DEB 包：

```
sudo dpkg -i yueshu-graph-3.5.1.ubuntu1804.amd64.deb
```


Copyright © Yueshu Graph Technology - All rights reserved.

配置 **License** 管理工具（**LM**）地址

1. 在悦数图数据库的 Meta 服务配置文件（`nebula-metad.conf`）中，设置 `license_manager_url` 的值为许可证管理工具所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。
2. 将 Meta、Storage 和 Graph 服务的配置文件（`nebula-metad.conf`、`nebula-graphd.conf`、`nebula-storaged.conf`）中的所有 `local_ip`（默认 `127.0.0.1`）替换为各服务所在主机的真实 IP，以及将 `meta_server_addrs` 地址替换为 Meta 服务所在主机 IP 地址和端口号 `9559`。

后续操作

- 启动悦数图数据库
- 连接悦数图数据库

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.2.2 步骤 2：启动悦数图数据库服务

悦数图数据库支持使用脚本管理服务和使用 systemd 管理服务。

Caution

这两种方式互不兼容，选择使用其中一种。

使用脚本管理服务

使用脚本 `nebula.service` 管理服务，包括启动、停止、重启、中止和查看。

Note

`nebula.service` 的默认路径是 `/usr/local/nebula/scripts`，如果修改过安装路径，请使用实际路径。

语法

```
$ sudo /usr/local/nebula/scripts/nebula.service
[-v] [-c <config_file_path>]
<start | stop | restart | kill | status>
<metad | graphd | storaged | all>
```

参数	说明
`-v`	显示详细调试信息。
`-c`	指定配置文件路径，默认路径为 `/usr/local/nebula/etc/` 。
`start`	启动服务。
`stop`	停止服务。
`restart`	重启服务。
`kill`	中止服务。
`status`	查看服务状态。
`metad`	管理 Meta 服务。
`graphd`	管理 Graph 服务。
`storaged`	管理 Storage 服务。
`all`	管理所有服务。

使用 **systemd** 管理服务

为方便使用，悦数图数据库企业版支持用 systemd 管理服务，通过 `systemctl` 启动、停止、重启和查看服务。

Note

- 安装悦数图数据库企业版后，systemd 所需的 `.service` 文件在安装目录的 `etc/unit` 目录内，使用 RPM/DEB 包安装的 NebulaGraph，会自动将这些 `.service` 文件放入 `/usr/lib/systemd/system` 目录内，并且 `ExecStart` 也会根据指定的悦数图数据库安装路径进行生成，因此可以直接使用 `systemctl` 命令。
- 对于使用悦数运维监控安装的悦数图数据库，不支持使用 `systemctl` 管理服务。
- 对于其他方式安装的企业版 NebulaGraph，需要用户手动将 `.service` 文件移动到 `/usr/lib/systemd/system` 目录内，并修改 `.service` 文件内的 `ExecStart` 的文件路径，才可以正常使用 `systemctl` 命令。

Copyright © Yueshu Graph Technology - All rights reserved.

语法

```
$ systemctl <start | stop | restart | status > <nebula | nebula-metad | nebula-graphd | nebula-storaged>
```

参数	说明
`start`	启动服务。
`stop`	停止服务。
`restart`	重启服务。
`status`	查看服务状态。
`nebula`	管理所有服务。
`nebula-metad`	管理 Meta 服务。
`nebula-graphd`	管理 Graph 服务。
`nebula-storaged`	管理 Storage 服务。

启动悦数图数据库服务

执行如下命令启动服务：

```
$ sudo /usr/local/nebula/scripts/nebula.service start all
[INFO] Starting nebula-metad...
[INFO] Done
[INFO] Starting nebula-graphd...
[INFO] Done
[INFO] Starting nebula-storaged...
[INFO] Done
```

或者：

```
$ systemctl start nebula
```

如果需要设置开机自动启动，命令如下：

```
$ systemctl enable nebula
```

停止悦数图数据库服务

Danger

请勿使用 `kill -9` 命令强制终止进程，否则可能较小概率出现数据丢失。

执行如下命令停止悦数图数据库服务：

```
$ sudo /usr/local/nebula/scripts/nebula.service stop all
[INFO] Stopping nebula-metad...
[INFO] Done
[INFO] Stopping nebula-graphd...
[INFO] Done
[INFO] Stopping nebula-storaged...
[INFO] Done
```

或者：

```
$ systemctl stop nebula
```

查看悦数图数据库服务

执行如下命令查看悦数图数据库服务状态：

Copyright © Yueshu Graph Technology - All rights reserved.

```
$ sudo /usr/local/nebula/scripts/nebula.service status all
```

- 如果返回如下结果，表示悦数图数据库服务正常运行。

```
[INFO] nebula-metad(33fd35e): Running as 29020, Listening on 9559
[INFO] nebula-graphd(33fd35e): Running as 29095, Listening on 9669
[WARN] nebula-storaged after v3.0.0 will not start service until it is added to cluster.
[WARN] See Manage Storage hosts:ADD HOSTS in https://docs.nebula-graph.io/
[INFO] nebula-storaged(33fd35e): Running as 29147, Listening on 9779
```

Note

正常启动悦数图数据库后，`nebula-storaged` 进程的端口显示红色。这是因为 `nebula-storaged` 在启动流程中会等待 `nebula-metad` 添加当前 Storage 服务，当前 Storage 服务收到 Ready 信号后才会正式启动服务。从 3.0.0 版本开始，在配置文件中添加的 Storage 节点无法直接读写，配置文件的作用仅仅是将 Storage 节点注册至 Meta 服务中。必须使用 `ADD HOSTS` 命令后，才能正常读写 Storage 节点。更多信息，参见管理 Storage 主机。

- 如果返回类似如下结果，表示悦数图数据库服务异常，可以根据异常服务信息进一步排查，或者在 悦数图数据库社区寻求帮助。

```
[INFO] nebula-metad: Running as 25600, Listening on 9559
[INFO] nebula-graphd: Exited
[INFO] nebula-storaged: Running as 25646, Listening on 9779
```

也可以使用 `systemctl` 命令查看悦数图数据库服务状态:

```
$ systemctl status nebula
● nebula.service
   Loaded: loaded (/usr/lib/systemd/system/nebula.service; disabled; vendor preset: disabled)
   Active: active (exited) since — 2022-03-28 04:13:24 UTC; 1h 47min ago
  Process: 21772 ExecStart=/usr/local/ent-nightly/scripts/nebula.service start all (code=exited, status=0/SUCCESS)
 Main PID: 21772 (code=exited, status=0/SUCCESS)
    Tasks: 325
   Memory: 424.5M
   CGroup: /system.slice/nebula.service
           ├─21789 /usr/local/ent-nightly/bin/nebula-metad --flagfile /usr/local/ent-nightly/etc/nebula-metad.conf
           ├─21827 /usr/local/ent-nightly/bin/nebula-graphd --flagfile /usr/local/ent-nightly/etc/nebula-graphd.conf
           └─21900 /usr/local/ent-nightly/bin/nebula-storaged --flagfile /usr/local/ent-nightly/etc/nebula-storaged.conf

3月 28 04:13:24 xxxxxx systemd[1]: Started nebula.service.
...
```

悦数图数据库服务由 Meta 服务、Graph 服务和 Storage 服务共同提供，这三种服务的配置文件都保存在安装目录的 `etc` 目录内，默认路径为 `/usr/local/nebula/etc/`，用户可以检查相应的配置文件排查问题。

下一步

- 连接悦数图数据库

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.2.3 步骤 3：连接悦数图数据库

本文介绍如何使用原生命令行客户端 Nebula Console 连接悦数图数据库。

Caution

首次连接到悦数图数据库后，必须先注册 Storage 服务，才能正常查询数据。

悦数图数据库支持多种类型的客户端，包括命令行客户端、可视化界面客户端和流行编程语言客户端。详情参见客户端列表。

前提条件

- 悦数图数据库服务已启动。
- 运行 Nebula Console 的机器和运行悦数图数据库的服务器网络互通。
- Nebula Console 的版本兼容悦数图数据库的版本。

Note

版本相同的 Nebula Console 和悦数图数据库兼容程度最高，版本不同的 Nebula Console 连接悦数图数据库时，可能会有兼容问题，或者无法连接并报错 `incompatible version between client and server` 。

操作步骤

1. 在 Nebula Console 下载页面，确认需要的版本，单击 **Assets**。

Note

建议选择最新版本。

2. 在 **Assets** 区域找到机器运行所需的二进制文件，下载文件到机器上。
3. （可选）为方便使用，重命名文件为 `nebula-console` 。

Note

在 Windows 系统中，请重命名为 `nebula-console.exe` 。

4. 在运行 Nebula Console 的机器上执行如下命令，为用户授予 nebula-console 文件的执行权限。

Note

Windows 系统请跳过此步骤。

```
$ chmod 111 nebula-console
```

5. 在命令行界面中，切换工作目录至 nebula-console 文件所在目录。

Copyright © Yueshu Graph Technology - All rights reserved.

6. 执行如下命令连接悦数图数据库。

- Linux 或 macOS

```
$ ./nebula-console -addr <ip> -port <port> -u <username> -p <password>
  [-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

- Windows

```
> nebula-console.exe -addr <ip> -port <port> -u <username> -p <password>
  [-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

参数说明如下。

参数	说明
`-h/-help`	显示帮助菜单。
`-addr/-address`	设置要连接的 Graph 服务的 IP 地址。默认地址为 127.0.0.1。
`-P/-port`	设置要连接的 Graph 服务的端口。默认端口为 9669。
`-u/-user`	设置悦数图数据库账号的用户名。未启用身份认证时，可以使用任意已存在的用户名（默认为 `root`）。
`-p/-password`	设置用户名对应的密码。未启用身份认证时，密码可以填写任意字符。
`-t/-timeout`	设置整数类型的连接超时时间。单位为毫秒，默认值为 120。
`-e/-eval`	设置字符串类型的 nGQL 语句。连接成功后会执行一次该语句并返回结果，然后自动断开连接。
`-f/-file`	设置存储 nGQL 语句的文件的路径。连接成功后会执行该文件内的 nGQL 语句并返回结果，执行完毕后自动断开连接。
`-enable_ssl`	连接悦数图数据库时使用 SSL 加密。
`-ssl_root_ca_path`	指定 CA 证书的存储路径。
`-ssl_cert_path`	指定 CRT 证书的存储路径。
`-ssl_private_key_path`	指定私钥文件的存储路径。

更多参数参见项目仓库。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.2.4 注册 Storage 服务

首次连接到悦数图数据库后，需要先添加 Storage 主机，并确认主机都处于在线状态。

> 版本兼容性
>
> 从悦数图数据库 3.0.0 版本开始，必须先使用 `ADD HOSTS` 添加主机，才能正常通过 Storage 服务读写数据。

前提条件

已连接悦数图数据库服务。

操作步骤

1. 添加 Storage 主机。

   执行如下命令添加主机：

```
ADD HOSTS <ip>:<port> [,<ip>:<port> ...];
```

   示例：

```
nebula> ADD HOSTS 192.168.10.100:9779, 192.168.10.101:9779, 192.168.10.102:9779;
```

> **Caution**
>
> 请确保添加的主机 IP 和配置文件 `nebula-storaged.conf` 中 `local_ip` 配置的 IP 一致，否则会导致添加 Storage 主机失败。关于配置文件的详情，参见配置管理。

2. 检查主机状态，确认全部在线。

```
nebula> SHOW HOSTS;
+-----------------+------+----------+--------------+----------------------  +------------------------+---------+
| Host            | Port | Status   | Leader count | Leader distribution  |   Partition distribution | Version |
+-----------------+------+----------+--------------+----------------------  +------------------------+---------+
| "192.168.10.100" | 9779 | "ONLINE" | 0            | "No valid partition" | "No   valid partition"   | "3.5.1" |
| "192.168.10.101" | 9779 | "ONLINE" | 0            | "No valid partition" | "No   valid partition"   | "3.5.1" |
| "192.168.10.102" | 9779 | "ONLINE" | 0            | "No valid partition" | "No   valid partition"   | "3.5.1" |
+-----------------+------+----------+--------------+----------------------  +------------------------+---------+
```

   在返回结果的 **Status** 列，可以看到所有 Storage 主机都在线。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.2.5 步骤 4：使用常用 nGQL（CRUD 命令）

本文介绍悦数图数据库查询语言的基础语法，包括用于 Schema 创建和常用增删改查操作的语句。

如需了解更多语句的用法，参见 nGQL 指南。

使用说明

图空间和 SCHEMA

一个悦数图数据库实例由一个或多个图空间组成。每个图空间都是物理隔离的，用户可以在同一个实例中使用不同的图空间存储不同的数据集。

为了在图空间中插入数据，需要为图数据库定义一个 Schema。悦数图数据库的 Schema 是由如下几部分组成。

组成部分	说明
点（Vertex）	表示现实世界中的实体。一个点可以有 0 到多个标签。
标签（Tag）	点的类型，定义了一组描述点类型的属性。
边（Edge）	表示两个点之间有方向的关系。
边类型（Edge type）	边的类型，定义了一组描述边的类型的属性。

更多信息，请参见数据结构。

本文将使用下图的数据集演示基础操作的语法。

Copyright © Yueshu Graph Technology - All rights reserved.

Copyright © Yueshu Graph Technology - All rights reserved.

异步实现创建和修改

Caution

在悦数图数据库中，下列创建和修改操作是异步实现的。要在下一个心跳周期之后才能生效，否则访问会报错。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。

- `CREATE SPACE`
- `CREATE TAG`
- `CREATE EDGE`
- `ALTER TAG`
- `ALTER EDGE`
- `CREATE TAG INDEX`
- `CREATE EDGE INDEX`

Note

默认心跳周期是 10 秒。修改心跳周期参数 `heartbeat_interval_secs`，请参见配置简介。

第一步：创建和选择图空间

NGQL 语法

- 创建图空间

```
CREATE SPACE [IF NOT EXISTS] <graph_space_name> (
[partition_num = <partition_number>,]
[replica_factor = <replica_number>,]
vid_type = {FIXED_STRING(<N>) | INT64}
)

[COMMENT = '<comment>'];
```

参数详情请参见 CREATE SPACE。

- 列出创建成功的图空间

```
nebula> SHOW SPACES;
```

- 选择数据库

```
USE <graph_space_name>;
```

示例

1. 执行如下语句创建名为 `basketballplayer` 的图空间。

```
nebula> CREATE SPACE basketballplayer(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
```

Note

如果报错提示 `[ERROR (-1005)]: Host not enough!`，请检查是否已添加 Storage 主机。

2. 执行命令 `SHOW HOSTS` 检查分片的分布情况，确保平衡分布。

```
nebula> SHOW HOSTS;
+-------------+-----------+-----------+--------------+---------------------------------+------------------------+---------+
| Host        | Port      | Status    | Leader count | Leader distribution             | Partition distribution | Version |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+------------+-----------+-----------+-------------+-------------------------------+-----------------------+---------+
| "storaged0" | 9779      | "ONLINE"  | 5           | "basketballplayer:5"          | "basketballplayer:5"  | "3.5.1" |
| "storaged1" | 9779      | "ONLINE"  | 5           | "basketballplayer:5"          | "basketballplayer:5"  | "3.5.1" |
| "storaged2" | 9779      | "ONLINE"  | 5           | "basketballplayer:5"          | "basketballplayer:5"  | "3.5.1" |
+------------+-----------+-----------+-------------+-------------------------------+-----------------------+---------+
```

如果 **Leader distribution** 分布不均匀，请执行命令 `BALANCE LEADER` 重新分配。更多信息，请参见 Storage 负载均衡。

3. 选择图空间 `basketballplayer` 。

```
nebula[(none)]> USE basketballplayer;
```

用户可以执行命令 `SHOW SPACES` 查看创建的图空间。

```
nebula> SHOW SPACES;
+--------------------+
| Name               |
+--------------------+
| "basketballplayer" |
+--------------------+
```

第二步：创建 **Tag** 和 **Edge type**

NGQL 语法

```
CREATE {TAG | EDGE} [IF NOT EXISTS] {<tag_name> | <edge_type_name>}
    (
      <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']
      [{, <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']} ...]
    )
    [TTL_DURATION = <ttl_duration>]
    [TTL_COL = <prop_name>]
    [COMMENT = '<comment>'];
```

参数详情请参见 CREATE TAG 和 CREATE EDGE。

示例

创建 Tag: `player` 和 `team` ，以及 Edge type: `follow` 和 `serve` 。说明如下表。

名称	类型	属性
player	Tag	name (string), age (int)
team	Tag	name (string)
follow	Edge type	degree (int)
serve	Edge type	start_year (int), end_year (int)

```
nebula> CREATE TAG player(name string, age int);

nebula> CREATE TAG team(name string);

nebula> CREATE EDGE follow(degree int);

nebula> CREATE EDGE serve(start_year int, end_year int);
```

第三步：插入数据

用户可以使用 `INSERT` 语句，基于现有的 Tag 插入点，或者基于现有的 Edge type 插入边。

NGQL 语法

- 插入点

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES <vid>: ([prop_value_list])

tag_props:
  tag_name ([prop_name_list])

prop_name_list:
   [prop_name [, prop_name] ...]
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
prop_value_list:
   [prop_value [, prop_value] ...]
```

`vid` 是 Vertex ID 的缩写，`vid` 在一个图空间中是唯一的。参数详情请参见 INSERT VERTEX。

- 插入边

```
INSERT EDGE [IF NOT EXISTS] <edge_type> ( <prop_name_list> ) VALUES
<src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> )
[, <src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> ), ...];

<prop_name_list> ::=
[ <prop_name> [, <prop_name> ] ...]

<prop_value_list> ::=
[ <prop_value> [, <prop_value> ] ...]
```

参数详情请参见 INSERT EDGE。

示例

- 插入代表球员和球队的点。

```
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);

nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);

nebula> INSERT VERTEX player(name, age) VALUES "player102":("LaMarcus Aldridge", 33);

nebula> INSERT VERTEX team(name) VALUES "team203":("Trail Blazers"), "team204":("Spurs");
```

- 插入代表球员和球队之间关系的边。

```
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);

nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player102":(90);

nebula> INSERT EDGE follow(degree) VALUES "player102" -> "player100":(75);

nebula> INSERT EDGE serve(start_year, end_year) VALUES "player101" -> "team204":(1999, 2018),"player102" -> "team203":(2006,  2015);
```

第四步：查询数据

- GO 语句可以根据指定的条件遍历数据库。`GO` 语句从一个或多个点开始，沿着一条或多条边遍历，返回 `YIELD` 子句中指定的信息。
- FETCH 语句可以获得点或边的属性。
- LOOKUP 语句是基于索引的，和 `WHERE` 子句一起使用，查找符合特定条件的数据。
- MATCH 语句是查询图数据最常用的，可以灵活的描述各种图模式，但是它依赖索引去匹配悦数图数据库中的数据模型，性能也还需要调优。

NGQL 语法

- `GO`

```
GO [[<M> TO] <N> {STEP|STEPS}] FROM <vertex_list>
OVER <edge_type_list> [{REVERSELY | BIDIRECT}]
[ WHERE <conditions> ]
YIELD [DISTINCT] <return_list>
[{ SAMPLE <sample_list> | <limit_by_list_clause> }]
[| GROUP BY {<col_name> | expression> | <position>} YIELD <col_name>]
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
[| ORDER BY <expression> [{ASC | DESC}]]
[| LIMIT [<offset>,] <number_rows>];
```

- `FETCH`
- 查询 Tag 属性

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *}
<vid> [, vid ...]
YIELD <return_list> [AS <alias>];
```

- 查询边属性

```
FETCH PROP ON <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid> ...]
YIELD <output>;
```

- `LOOKUP`

```
LOOKUP ON {<vertex_tag> | <edge_type>}
[WHERE <expression> [AND <expression> ...]]
YIELD <return_list> [AS <alias>];

<return_list>
    <prop_name> [AS <col_alias>] [, <prop_name> [AS <prop_alias>] ...];
```

- `MATCH`

```
MATCH <pattern> [<clause_1>]  RETURN <output>  [<clause_2>];
```

`GO` 语句示例

- 从 VID 为 `player101` 的球员开始，沿着边 `follow` 找到连接的球员。

```
nebula> GO FROM "player101" OVER follow YIELD id($$);
+-------------+
| id($$)      |
+-------------+
| "player100" |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "player102" |
+-------------+
```

- 从 VID 为 `player101` 的球员开始，沿着边 `follow` 查找年龄大于或等于 35 岁的球员，并返回他们的姓名和年龄，同时重命名对应的列。

```
nebula> GO FROM "player101" OVER follow WHERE properties($$).age >= 35 \
        YIELD properties($$).name AS Teammate, properties($$).age AS Age;
+-----------------+-----+
| Teammate        | Age |
+-----------------+-----+
| "Tim Duncan"    | 42  |
+-----------------+-----+
```

子句/符号	说明
`YIELD`	指定该查询需要返回的值或结果。
`$$`	表示边的终点。
`\`	表示换行继续输入。

- 从 VID 为 `player101` 的球员开始，沿着边 `follow` 查找连接的球员，然后检索这些球员的球队。为了合并这两个查询请求，可以使用管道符或临时变量。

- 使用管道符

```
nebula> GO FROM "player101" OVER follow YIELD dst(edge) AS id | \
        GO FROM $-.id OVER serve YIELD properties($$).name AS Team, \
        properties($^).name AS Player;
+-----------------+---------------------+
| Team            | Player              |
+-----------------+---------------------+
| "Trail Blazers" | "LaMarcus Aldridge" |
+-----------------+---------------------+
```

子句/符号	说明
`$^`	表示边的起点。
`|`	组合多个查询的管道符，将前一个查询的结果集用于后一个查询。
`$-`	表示管道符前面的查询输出的结果集。

- 使用临时变量

> **Note**
>
> 当复合语句作为一个整体提交给服务器时，其中的临时变量会在语句结束时被释放。

```
nebula> $var = GO FROM "player101" OVER follow YIELD dst(edge) AS id; \
        GO FROM $var.id OVER serve YIELD properties($$).name AS Team, \
        properties($^).name AS Player;
+-----------------+---------------------+
| Team            | Player              |
+-----------------+---------------------+
| "Trail Blazers" | "LaMarcus Aldridge" |
+-----------------+---------------------+
```

`FETCH` 语句示例

查询 VID 为 `player100` 的球员的属性。

```
nebula> FETCH PROP ON player "player100" YIELD properties(vertex);
+------------------------------+
| properties(VERTEX)           |
+------------------------------+
| {age: 42, name: "Tim Duncan"} |
+------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

Note

`LOOKUP` 和 `MATCH` 的示例在下文的索引 部分查看。

其他操作

修改点和边

用户可以使用 `UPDATE` 语句或 `UPSERT` 语句修改现有数据。

`UPSERT` 是 `UPDATE` 和 `INSERT` 的结合体。当使用 `UPSERT` 更新一个点或边，如果它不存在，数据库会自动插入一个新的点或边。

Note

每个 partition 内部，`UPSERT` 操作是一个串行操作，所以执行速度比执行 `INSERT` 或 `UPDATE` 慢很多。其仅在多个 partition 之间有并发。

nGQL 语法

- `UPDATE` 点

```
UPDATE VERTEX <vid> SET <properties to be updated>
[WHEN <condition>] [YIELD <columns>];
```

- `UPDATE` 边

```
UPDATE EDGE ON <edge_type> <source vid> -> <destination vid> [@rank]
SET <properties to be updated> [WHEN <condition>] [YIELD <columns to be output>];
```

- `UPSERT` 点或边

```
UPSERT {VERTEX <vid> | EDGE <edge_type>} SET <update_columns>
[WHEN <condition>] [YIELD <columns>];
```

示例

- 用 `UPDATE` 修改 VID 为 `player100` 的球员的 `name` 属性，然后用 `FETCH` 语句检查结果。

```
nebula> UPDATE VERTEX "player100" SET player.name = "Tim";

nebula> FETCH PROP ON player "player100" YIELD properties(vertex);
+------------------------+
| properties(VERTEX)     |
+------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| {age: 42, name: "Tim"} |
+------------------------+
```

- 用 `UPDATE` 修改某条边的 `degree` 属性，然后用 `FETCH` 检查结果。

```
nebula> UPDATE EDGE ON follow "player101" -> "player100" SET degree = 96;

nebula> FETCH PROP ON follow "player101" -> "player100" YIELD properties(edge);
+------------------+
| properties(EDGE) |
+------------------+
| {degree: 96}     |
+------------------+
```

- 用 `INSERT` 插入一个 VID 为 `player111` 的点，然后用 `UPSERT` 更新它。

```
nebula> INSERT VERTEX player(name,age) VALUES "player111":("David West", 38);

nebula> UPSERT VERTEX "player111" SET player.name = "David", player.age = $^.player.age + 11 \
        WHEN $^.player.name == "David West" AND $^.player.age > 20 \
        YIELD $^.player.name AS Name, $^.player.age AS Age;
+---------+-----+
| Name    | Age |
+---------+-----+
| "David" | 49  |
+---------+-----+
```

删除点和边

nGQL 语法

- 删除点

```
DELETE VERTEX <vid1>[, <vid2>...]
```

- 删除边

```
DELETE EDGE <edge_type> <src_vid> -> <dst_vid>[@<rank>]
[, <src_vid> -> <dst_vid>...]
```

示例

- 删除点

```
nebula> DELETE VERTEX "player111", "team203";
```

- 删除边

```
nebula> DELETE EDGE follow "player101" -> "team204";
```

使用索引

用户可以通过 CREATE INDEX 语句为 Tag 和 Edge type 增加索引。

> 使用索引必读
>
> `MATCH` 和 `LOOKUP` 语句的执行都依赖索引，但是索引会导致写性能大幅降低。请不要随意在生产环境中使用索引，除非很清楚使用索引对业务的影响。
>
> 必须为“已写入但未构建索引”的数据重建索引，否则无法在 `MATCH` 和 `LOOKUP` 语句中返回这些数据。参见重建索引。

nGQL 语法

- 创建索引

Copyright © Yueshu Graph Technology - All rights reserved.

```
CREATE {TAG | EDGE} INDEX [IF NOT EXISTS] <index_name>
ON {<tag_name> | <edge_name>} ([<prop_name_list>]) [COMMENT = '<comment>'];
```

- 重建索引

```
REBUILD {TAG | EDGE} INDEX <index_name>;
```

Note

为没有指定长度的变量属性创建索引时，需要指定索引长度。在 utf-8 编码中，一个中文字符占 3 字节，请根据变量属性长度设置合适的索引长度。例如 10 个中文字符，索引长度需要为 30。详情请参见创建索引。

基于索引的 **LOOKUP** 和 **MATCH** 示例

确保 `LOOKUP` 或 `MATCH` 有一个索引可用。如果没有，请先创建索引。

找到 Tag 为 `player` 的点的信息，它的 `name` 属性值为 `Tony Parker` 。

```
// 为 name 属性创建索引 player_index_1。
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_1 ON player(name(20));

// 重建索引确保能对已存在数据生效。
nebula> REBUILD TAG INDEX player_index_1
+------------+
| New Job Id |
+------------+
| 31         |
+------------+

// 使用 LOOKUP 语句检索点的属性。
nebula> LOOKUP ON player WHERE player.name == "Tony Parker" \
        YIELD properties(vertex).name AS name, properties(vertex).age AS age;
+---------------+-----+
| name          | age |
+---------------+-----+
| "Tony Parker" | 36  |
+---------------+-----+

// 使用 MATCH 语句检索点的属性。
nebula> MATCH (v:player{name:"Tony Parker"}) RETURN v;
+--------------------------------------------------+
| v                                                |
+--------------------------------------------------+
| ("player101" :player{age: 36, name: "Tony Parker"}) |
+--------------------------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

4.3 nGQL 命令汇总

4.3.1 函数

Copyright © Yueshu Graph Technology - All rights reserved.

- 数学函数

函数	说明
double abs(double x)	返回 x 的绝对值。
double floor(double x)	返回小于或等于 x 的最大整数。
double ceil(double x)	返回大于或等于 x 的最小整数。
double round(double x)	返回离 x 最近的整数值，如果 x 恰好在中间，则返回离 0 较远的整数。
double sqrt(double x)	返回 x 的平方根。
double cbrt(double x)	返回 x 的立方根。
double hypot(double x, double y)	返回直角三角形（直角边长为 x 和 y）的斜边长。
double pow(double x, double y)	返回 x^y 的值。
double exp(double x)	返回 e^x 的值。
double exp2(double x)	返回 2^x 的值。
double log(double x)	返回以自然数 e 为底 x 的对数。
double log2(double x)	返回以 2 为底 x 的对数。
double log10(double x)	返回以 10 为底 x 的对数。
double sin(double x)	返回 x 的正弦值。
double asin(double x)	返回 x 的反正弦值。
double cos(double x)	返回 x 的余弦值。
double acos(double x)	返回 x 的反余弦值。
double tan(double x)	返回 x 的正切值。
double atan(double x)	返回 x 的反正切值。
double rand()	返回 [0,1) 内的随机浮点数。
int rand32(int min, int max)	返回 `[min, max)` 内的一个随机 32 位整数。 用户可以只传入一个参数，该参数会判定为 `max`，此时 `min` 默认为 `0`。 如果不传入参数，此时会从带符号的 32 位 int 范围内随机返回。
int rand64(int min, int max)	返回 `[min, max)` 内的一个随机 64 位整数。 用户可以只传入一个参数，该参数会判定为 `max`，此时 `min` 默认为 `0`。 如果不传入参数，此时会从带符号的 64 位 int 范围内随机返回。
bit_and()	逐位做 AND 操作。
bit_or()	逐位做 OR 操作。
bit_xor()	逐位做 XOR 操作。
int size()	返回列表或映射中元素的数量，或字符串的长度。
int range(int start, int end, int step)	返回 `[start,end]` 中指定步长的值组成的列表。步长 `step` 默认为 1。
int sign(double x)	返回 x 的正负号。 如果 x 为 `0`，则返回 `0`。 如果 x 为负数，则返回 `-1`。 如果 x 为正数，则返回 `1`。
double e()	返回自然对数的底 e（2.718281828459045）。
double pi()	返回数学常数π（3.141592653589793）。
double radians()	将角度转换为弧度。 `radians(180)` 返回 `3.141592653589793`。

Copyright © Yueshu Graph Technology - All rights reserved.

- 聚合函数

函数	说明
avg()	返回参数的平均值。
count()	语法：`count({expr | *})` 。 `count()` 返回总行数（包括 NULL）。 `count(expr)` 返回满足表达式的非空值的总数。 `count()` 和 `size()` 是不同的。
max()	返回参数的最大值。
min()	返回参数的最小值。
collect()	`collect()` 函数返回一个符合表达式返回结果的列表。该函数可以将多条记录或值合并进一个列表，实现数据聚合。
std()	返回参数的总体标准差。
sum()	返回参数的和。

Copyright © Yueshu Graph Technology - All rights reserved.

- 字符串函数

函数	说明
int strcasecmp(string a, string b)	比较两个字符串（不区分大小写）。当 a=b 时，返回 0，当 a>b 是，返回大于 0 的数，当 a<b 时，返回小于 0 的数。
string lower(string a)	返回小写形式的字符串。
string toLower(string a)	和 `lower()` 相同。
string upper(string a)	返回大写形式的字符串。
string toUpper(string a)	和 `upper()` 相同。
int length(a)	返回给定字符串的长度或路径的长度，单位分别是字节和跳数。
string trim(string a)	删除字符串头部和尾部的空格。
string ltrim(string a)	删除字符串头部的空格。
string rtrim(string a)	删除字符串尾部的空格。
string left(string a, int count)	返回字符串左侧 `count` 个字符组成的子字符串。如果 `count` 超过字符串 a 的长度，则返回字符串 a。
string right(string a, int count)	返回字符串右侧 `count` 个字符组成的子字符串。如果 `count` 超过字符串 a 的长度，则返回字符串 a。
string lpad(string a, int size, string letters)	在字符串 a 的左侧填充 `letters` 字符串，并返回 `size` 长度的字符串。
string rpad(string a, int size, string letters)	在字符串 a 的右侧填充 `letters` 字符串，并返回 `size` 长度的字符串。
string substr(string a, int pos, int count)	从字符串 a 的指定位置 `pos` 开始（不包括 `pos` 位置的字符），提取右侧的 `count` 个字符，组成新的字符串并返回。
string substring(string a, int pos, int count)	和 `substr()` 相同。
string reverse(string)	逆序返回字符串。
string replace(string a, string b, string c)	将字符串 a 中的子字符串 b 替换为字符串 c。
list split(string a, string b)	在子字符串 b 处拆分字符串 a，返回一个字符串列表。
concat()	`concat()` 函数至少需要两个或以上字符串参数，并将所有参数连接成一个字符串。 语法：`concat(string1,string2,...)`
concat_ws()	`concat_ws()` 函数将两个或以上字符串参数与预定义的分隔符（separator）相连接。
extract()	`extract()` 从指定字符串中提取符合正则表达式的子字符串。
json_extract()	`json_extract()` 将指定 JSON 字符串转换为 map 类型。

Copyright © Yueshu Graph Technology - All rights reserved.

- 日期时间函数

函数	说明
int now()	根据当前系统返回当前时间戳。
timestamp timestamp()	根据当前系统返回当前时间戳。
date date()	根据当前系统返回当前日期（UTC 时间）。
time time()	根据当前系统返回当前时间（UTC 时间）。
datetime datetime()	根据当前系统返回当前日期和时间（UTC 时间）。

- Schema 相关函数
- 原生 nGQL 语句适用

函数	说明
id(vertex)	返回点 ID。数据类型和点 ID 的类型保持一致。
map properties(vertex)	返回点的所有属性。
map properties(edge)	返回边的所有属性。
string type(edge)	返回边的 Edge type。
src(edge)	返回边的起始点 ID。数据类型和点 ID 的类型保持一致。
dst(edge)	返回边的目的点 ID。数据类型和点 ID 的类型保持一致。
int rank(edge)	返回边的 rank。
vertex	返回点的信息。包括点 ID、Tag、属性和值。
edge	返回边的信息。包括 Edge type、起始点 ID、目的点 ID、rank、属性和值。
vertices	返回子图中的点的信息。详情参见 GET SUBGRAPH。
edges	返回子图中的边的信息。详情参见 GET SUBGRAPH。
path	返回路径信息。详情参见 FIND PATH。

- openCypher 兼容语句适用

函数	说明
id(<vertex>)	返回点 ID。数据类型和点 ID 的类型保持一致。
list tags(<vertex>)	返回点的 Tag，与 labels() 作用相同。
list labels(<vertex>)	返回点的 Tag，与 tags() 作用相同，用于兼容 openCypher 语法。
map properties(<vertex_or_edge>)	返回点或边的所有属性。
string type(<edge>)	返回边的 Edge type。
src(<edge>)	返回边的起始点 ID。数据类型和点 ID 的类型保持一致。
dst(<edge>)	返回边的目的点 ID。数据类型和点 ID 的类型保持一致。
vertex startNode(<path>)	获取一条边或一条路径并返回它的起始点 ID。
string endNode(<path>)	获取一条边或一条路径并返回它的目的点 ID。
int rank(<edge>)	返回边的 rank。

Copyright © Yueshu Graph Technology - All rights reserved.

- 列表函数

函数	说明
keys(expr)	返回一个列表，包含字符串形式的点、边或映射的所有属性。
labels(vertex)	返回点的 Tag 列表。
nodes(path)	返回路径中所有点的列表。
range(start, end [, step])	返回 `[start,end]` 范围内固定步长的列表，默认步长 `step` 为 1。
relationships(path)	返回路径中所有关系的列表。
reverse(list)	返回将原列表逆序排列的新列表。
tail(list)	返回不包含原列表第一个元素的新列表。
head(list)	返回列表的第一个元素。
last(list)	返回列表的最后一个元素。
reduce()	将表达式逐个应用于列表中的元素，然后和累加器中的当前结果累加，最后返回完整结果。

- 类型转换函数

函数	说明
bool toBoolean()	将字符串转换为布尔。
float toFloat()	将整数或字符串转换为浮点数。
string toString()	将任意数据类型转换为字符串类型。
int toInteger()	将浮点或字符串转换为整数。
set toSet()	将列表或集合转换为集合。
int hash()	`hash()` 函数返回参数的哈希值。其参数可以是数字、字符串、列表、布尔值、NULL 等类型的值，或者计算结果为这些类型的表达式。

- 谓词函数

谓词函数只返回 `true` 或 `false`，通常用于 `WHERE` 子句中。

```
<predicate>(<variable> IN <list> WHERE <condition>)
```

函数	说明
exists()	如果指定的属性在点、边或映射中存在，则返回 `true`，否则返回 `false`。
any()	如果指定的谓词适用于列表中的至少一个元素，则返回 `true`，否则返回 `false`。
all()	如果指定的谓词适用于列表中的每个元素，则返回 `true`，否则返回 `false`。
none()	如果指定的谓词不适用于列表中的任何一个元素，则返回 `true`，否则返回 `false`。
single()	如果指定的谓词适用于列表中的唯一一个元素，则返回 `true`，否则返回 `false`。

Copyright © Yueshu Graph Technology - All rights reserved.

- 条件表达式函数

函数	说明
CASE	使用条件来过滤 nGQL 查询语句的结果，常用于 `YIELD` 和 `RETURN` 子句中。 `CASE` 表达式会遍历所有条件，并在满足第一个条件时停止读取后续条件，然后返回结果。 如果不满足任何条件，将通过 `ELSE` 子句返回结果。如果没有 `ELSE` 子句且不满足任何条件，则返回 `NULL` 。
coalesce()	返回所有表达式中第一个非空元素。

4.3.2 通用查询语句

- MATCH

Copyright © Yueshu Graph Technology - All rights reserved.

```
MATCH <pattern> [<clause_1>]  RETURN <output>  [<clause_2>];
```


Copyright © Yueshu Graph Technology - All rights reserved.

模式	示例	说明
匹配点	`(v)`	用户可以在一对括号中使用自定义变量来表示模式中的点。例如 `(v)` 。
匹配 Tag	`MATCH (v:player) RETURN v`	用户可以在点的右侧用 `:<tag_name>` 表示模式中的 Tag。
匹配多 Tag	`MATCH (v:player:team) RETURN v`	用户可以用英文冒号（`:`）匹配多 Tag 的点。

匹配点的属性	`MATCH (v:player{name:"Tim Duncan"}) RETURN v`
	
	`MATCH (v) WITH v, properties(v) as props, keys(properties(v)) as kk WHERE [i in kk where props[i] == "Tim Duncan"] RETURN v`
		用户可以在 Tag 的右侧用 `{<prop_name>: <prop_value>}` 表示模式中点的属性；或者不指定 Tag 直接匹配点的属性。

匹配单点 ID	`MATCH (v) WHERE id(v) == 'player101' RETURN v`	用户可以使用点 ID 去匹配点。 `id()` 函数可以检索点的 ID。
匹配多点 ID	`MATCH (v:player { name: 'Tim Duncan' })--(v2) WHERE id(v2) IN ["player101", "player102"] RETURN v2`	要匹配多个点的 ID，可以用 `WHERE id(v) IN [vid_list]` 。
匹配连接的点	`MATCH (v:player{name:"Tim Duncan"})--(v2) RETURN v2.player.name AS Name`	用户可以使用 `--` 符号表示两个方向的边，并匹配这些边连接的点。用户可以在 `--` 符号上增加 `<` 或 `>` 符号指定边的方向。
匹配路径	`MATCH p=(v:player{name:"Tim Duncan"})-->(v2) RETURN p`	连接起来的点和边构成了路径。用户可以使用自定义变量命名路径。
匹配边	`MATCH (v:player{name:"Tim Duncan"})-[e]-(v2) RETURN e` `MATCH ()<-[e]-()  RETURN e`	除了用 `--` 、 `-->` 、 `<--` 表示未命名的边之外，用户还可以在方括号中使用自定义变量命名边。例如 `-[e]-` 。
匹配 Edge type	`MATCH ()-[e:follow]->() RETURN e`	和点一样，用户可以用 `:<edge_type>` 表示模式中的 Edge type，例如 `-[e:follow]-` 。

匹配边的属性	`MATCH (v:player{name:"Tim Duncan"})-[e:follow{degree:95}]->(v2) RETURN e`
	
	`MATCH ()-[e]->() WITH e, properties(e) as props, keys(properties(e)) as kk WHERE [i in kk where props[i] == 90] RETURN e`
		用户可以用 `{<prop_name>: <prop_value>}` 表示模式中 Edge type 的属性，例如 `[e:follow{likeness:95}]` ；或者不指定 Edge type 直接匹配边的属性。

匹配多个 Edge type	`MATCH (v:player{name:"Tim Duncan"})-[e:follow | :serve]->(v2) RETURN e`	使用 `|` 可以匹配多个 Edge type，例如 `[e:follow | :serve]` 。第一个 Edge type 前的英文冒号（`:`）不可省略，后续 Edge type 前的英文冒号可以省略，例如 `[e:follow | serve]` 。
匹配多条边	`MATCH (v:player{name:"Tim Duncan"})-[]->(v2)<-[e:serve]-(v3) RETURN v2, v3`	用户可以扩展模式，匹配路径中的多条边。
匹配定长路径	`MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) RETURN DISTINCT v2 AS Friends`	用户可以在模式中使用 `:<edge_type>*<hop>` 匹配定长路径。 `hop` 必须是一个非负整数。 `e` 的数据类型是列表。
匹配变长路径	`MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*1..3]->(v2) RETURN v2 AS Friends`	`minHop` ：可选项。表示路径的最小长度。 `minHop` 必须是一个非负整数，默认值为 1。 `maxHop` ：可选项。表示路径的最大长度。 `maxHop` 必须是一个非负整数，默认为无穷大。 `e` 的数据类型是列表。
匹配多个 Edge type 的变长路径	`MATCH p=(v:player{name:"Tim Duncan"})-[e:follow | serve*2]->(v2) RETURN DISTINCT v2`	用户可以在变长或定长模式中指定多个 Edge type。 `hop` 、 `minHop` 和 `maxHop` 对所有 Edge type 都生效。 `e` 的数据类型是列表。
检索点或边的信息	`MATCH (v:player{name:"Tim Duncan"}) RETURN v` `MATCH (v:player{name:"Tim Duncan"})-[e]->(v2) RETURN e`	使用 `RETURN {<vertex_name> | <edge_name>}` 检索点或边的所有信息。
检索点 ID		使用 `id()` 函数检索点 ID。

Copyright © Yueshu Graph Technology - All rights reserved.

模式	示例	说明
	`MATCH (v:player{name:"Tim Duncan"}) RETURN id(v)`	
检索 Tag	`MATCH (v:player{name:"Tim Duncan"}) RETURN labels(v)`	使用 `labels()` 函数检索点上的 Tag 列表。 检索列表 `labels(v)` 中的第 N 个元素，可以使用 `labels(v)[n-1]` 。
检索点或边的单个属性	`MATCH (v:player{name:"Tim Duncan"}) RETURN v.player.age`	使用 `RETURN {<vertex_name> | <edge_name>}.<property>` 检索单个属性。 使用 `AS` 设置属性的别名。
检索点或边的所有属性	`MATCH p=(v:player{name:"Tim Duncan"})-[]->(v2) RETURN properties(v2)`	使用 `properties()` 函数检索点或边的所有属性。
检索 Edge type	`MATCH p=(v:player{name:"Tim Duncan"})-[e]->() RETURN DISTINCT type(e)`	使用 `type()` 函数检索匹配的 Edge type。
检索路径	`MATCH p=(v:player{name:"Tim Duncan"})-[*3]->() RETURN p`	使用 `RETURN <path_name>` 检索匹配路径的所有信息。
检索路径中的点	`MATCH p=(v:player{name:"Tim Duncan"})-[]->(v2) RETURN nodes(p)`	使用 `nodes()` 函数检索路径中的所有点。
检索路径中的边	`MATCH p=(v:player{name:"Tim Duncan"})-[]->(v2) RETURN relationships(p)`	使用 `relationships()` 函数检索路径中的所有边。
检索路径长度	`MATCH p=(v:player{name:"Tim Duncan"})-[*..2]->(v2) RETURN p AS Paths, length(p) AS Length`	使用 `length()` 函数检索路径的长度。

- OPTIONAL MATCH

模式	示例	说明
作为 `MATCH` 语句的可选项去匹配图数据库中的模式	`MATCH (m)-[]->(n) WHERE id(m)=="player100" OPTIONAL MATCH (n)-[]->(l) RETURN id(m),id(n),id(l)`	如果图数据库中没有对应的模式，对应的列返回 `NULL` 。

- LOOKUP

```
LOOKUP ON {<vertex_tag> | <edge_type>}
[WHERE <expression> [AND <expression> ...]]
YIELD <return_list> [AS <alias>]
```

模式	示例	说明
检索点	`LOOKUP ON player WHERE player.name == "Tony Parker" YIELD player.name AS name, player.age AS age`	返回 Tag 为 `player` 且 `name` 为 `Tony Parker` 的点。
检索边	`LOOKUP ON follow WHERE follow.degree == 90 YIELD follow.degree`	返回 Edge type 为 `follow` 且 `degree` 为 `90` 的边。
通过 Tag 列出所有点	`LOOKUP ON player YIELD properties(vertex),id(vertex)`	查找所有 Tag 为 `player` 的点 VID。
通过 Edge type 列出边	`LOOKUP ON follow YIELD edge AS e`	查找 Edge type 为 `follow` 的所有边的信息。
统计点	`LOOKUP ON player YIELD id(vertex)| YIELD COUNT(*) AS Player_Count`	统计 Tag 为 `player` 的点。
统计边	`LOOKUP ON follow YIELD edge as e | YIELD COUNT(*) AS Like_Count`	统计 Edge type 为 `follow` 的边。

Copyright © Yueshu Graph Technology - All rights reserved.

- GO

```
GO [[<M> TO] <N> {STEP|STEPS} ] FROM <vertex_list>
OVER <edge_type_list> [{REVERSELY | BIDIRECT}]
[ WHERE <conditions> ]
YIELD [DISTINCT] <return_list>
[{SAMPLE <sample_list> | LIMIT <limit_list>}]
[| GROUP BY {col_name | expr | position} YIELD <col_name>]
[| ORDER BY <expression> [{ASC | DESC}]]
[| LIMIT [<offset_value>,] <number_rows>]
```

示例	说明
`GO FROM "player102" OVER serve YIELD dst(edge)`	返回 player102 所属队伍。
`GO 2 STEPS FROM "player102" OVER follow YIELD dst(edge)`	返回距离 player102 两跳的朋友。
`GO FROM "player100", "player102" OVER serve WHERE properties(edge).start_year > 1995 YIELD DISTINCT properties($$).name AS team_name, properties(edge).start_year AS start_year, properties($^).name AS player_name`	添加过滤条件。
`GO FROM "player100" OVER follow, serve YIELD properties(edge).degree, properties(edge).start_year`	遍历多个 Edge type。属性没有值时，会显示 `NULL` 。
`GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS destination`	返回 player100 入方向的邻居点。
`GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS id | GO FROM $-.id OVER serve WHERE properties($^).age > 20 YIELD properties($^).name AS FriendOf, properties($$).name AS Team`	查询 player100 的朋友和朋友所属队伍。
`GO FROM "player102" OVER follow YIELD dst(edge) AS both`	返回 player102 所有邻居点。
`GO 2 STEPS FROM "player100" OVER follow YIELD src(edge) AS src, dst(edge) AS dst, properties($$).age AS age | GROUP BY $-.dst YIELD $-.dst AS dst, collect_set($-.src) AS src, collect($-.age) AS age`	根据年龄分组。

Copyright © Yueshu Graph Technology - All rights reserved.

- FETCH
- 获取点的属性值

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *}
<vid> [, vid ...]
YIELD <return_list> [AS <alias>]
```

示例	说明
`FETCH PROP ON player "player100" YIELD properties(vertex)`	在 `FETCH` 语句中指定 Tag 获取对应点的属性值。
`FETCH PROP ON player "player100" YIELD player.name AS name`	使用 `YIELD` 子句指定返回的属性。
`FETCH PROP ON player "player101", "player102", "player103" YIELD properties(vertex)`	指定多个点 ID 获取多个点的属性值，点之间用英文逗号（,）分隔。
`FETCH PROP ON player, t1 "player100", "player103" YIELD properties(vertex)`	在 `FETCH` 语句中指定多个 Tag 获取属性值。Tag 之间用英文逗号（,）分隔。
`FETCH PROP ON * "player100", "player106", "team200" YIELD properties(vertex)`	在 `FETCH` 语句中使用 `*` 获取当前图空间所有标签里，点的属性值。

- 获取边的属性值

```
FETCH PROP ON <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid> ...]
YIELD <output>;
```

示例	说明
`FETCH PROP ON serve "player100" -> "team204" YIELD properties(edge)`	获取连接 player100 和 team204 的边 serve 的所有属性值。
`FETCH PROP ON serve "player100" -> "team204" YIELD serve.start_year`	使用 `YIELD` 子句指定返回的属性。
`FETCH PROP ON serve "player100" -> "team204", "player133" -> "team202" YIELD properties(edge)`	指定多个边模式（`<src_vid> -> <dst_vid>[@<rank>]`）获取多个边的属性值。模式之间用英文逗号（,）分隔。
`FETCH PROP ON serve "player100" -> "team204"@1 YIELD properties(edge)`	要获取 rank 不为 0 的边，请在 FETCH 语句中设置 rank。
`GO FROM "player101" OVER follow YIELD follow._src AS s, follow._dst AS d | FETCH PROP ON follow $-.s -> $-.d YIELD follow.degree`	返回从点 player101 开始的 follow 边的 degree 值。
`$var = GO FROM "player101" OVER follow YIELD follow._src AS s, follow._dst AS d; FETCH PROP ON follow $var.s -> $var.d YIELD follow.degree`	自定义变量构建查询。

Copyright © Yueshu Graph Technology - All rights reserved.

- SHOW

语句	语法	示例	说明
SHOW CHARSET	`SHOW CHARSET`	`SHOW CHARSET`	显示当前的字符集。
SHOW COLLATION	`SHOW COLLATION`	`SHOW COLLATION`	显示当前的排序规则。
SHOW CREATE SPACE	`SHOW CREATE SPACE <space_name>`	`SHOW CREATE SPACE basketballplayer`	显示指定图空间的创建语句。
SHOW CREATE TAG/EDGE	`SHOW CREATE {TAG <tag_name> | EDGE <edge_name>}`	`SHOW CREATE TAG player`	显示指定 Tag/Edge type 的基本信息。
SHOW HOSTS	`SHOW HOSTS [GRAPH | STORAGE | META]`	`SHOW HOSTS` `SHOW HOSTS GRAPH`	显示 Graph、Storage、Meta 服务主机信息、版本信息。
SHOW INDEX STATUS	`SHOW {TAG | EDGE} INDEX STATUS`	`SHOW TAG INDEX STATUS`	重建原生索引的作业状态，以便确定重建索引是否成功。
SHOW INDEXES	`SHOW {TAG | EDGE} INDEXES`	`SHOW TAG INDEXES`	列出当前图空间内的所有 Tag 和 Edge type（包括属性）的索引。
SHOW PARTS	`SHOW PARTS [<part_id>]`	`SHOW PARTS`	显示图空间中指定分片或所有分片的信息。
SHOW ROLES	`SHOW ROLES IN <space_name>`	`SHOW ROLES in basketballplayer`	显示分配给用户的角色信息。
SHOW SNAPSHOTS	`SHOW SNAPSHOTS`	`SHOW SNAPSHOTS`	显示所有快照信息。
SHOW SPACES	`SHOW SPACES`	`SHOW SPACES`	显示现存的图空间。
SHOW STATS	`SHOW STATS`	`SHOW STATS`	显示最近 `STATS` 作业收集的图空间统计信息。
SHOW TAGS/EDGES	`SHOW TAGS | EDGES`	`SHOW TAGS` 、 `SHOW EDGES`	显示当前图空间内的所有 Tag/Edge type。
SHOW USERS	`SHOW USERS`	`SHOW USERS`	显示用户信息。
SHOW SESSIONS	`SHOW SESSIONS`	`SHOW SESSIONS`	显示所有会话信息。
SHOW SESSIONS	`SHOW SESSION <Session_Id>`	`SHOW SESSION 1623304491050858`	指定会话 ID 进行查看。
SHOW QUERIES	`SHOW [ALL] QUERIES`	`SHOW QUERIES`	查看当前 Session 中正在执行的查询请求信息。
SHOW META LEADER	`SHOW META LEADER`	`SHOW META LEADER`	显示当前 Meta 集群的 leader 信息。

Copyright © Yueshu Graph Technology - All rights reserved.

4.3.3 子句和选项

子句	语法	示例	说明
GROUP BY	`GROUP BY <var> YIELD <var>, <aggregation_function(var)>`	`GO FROM "player100" OVER follow BIDIRECT YIELD $$.player.name as Name | GROUP BY $-.Name YIELD $-.Name as Player, count(*) AS Name_Count`	查找所有连接到 player100 的点，并根据他们的姓名进行分组，返回姓名的出现次数。
LIMIT	`YIELD <var> [| LIMIT [<offset_value>,] <number_rows>]`	`GO FROM "player100" OVER follow REVERSELY YIELD $$.player.name AS Friend, $$.player.age AS Age | ORDER BY $-.Age, $-.Friend | LIMIT 1, 3`	从排序结果中返回第 2 行开始的 3 行数据。
SKIP	`RETURN <var> [SKIP <offset>] [LIMIT <number_rows>]`	`MATCH (v:player{name:"Tim Duncan"}) --> (v2) RETURN v2.player.name AS Name, v2.player.age AS Age ORDER BY Age DESC SKIP 1`	用户可以单独使用 `SKIP <offset>` 设置偏移量，后面不需要添加 `LIMIT <number_rows>` 。
SAMPLE	`<go_statement> SAMPLE <sample_list>;`	`GO 3 STEPS FROM "player100" OVER * YIELD properties($$).name AS NAME, properties($$).age AS Age SAMPLE [1,2,3];`	在结果集中均匀取样并返回指定数量的数据。
ORDER BY	`<YIELD clause> ORDER BY <expression> [ASC | DESC] [, <expression> [ASC | DESC] ...]`	`FETCH PROP ON player "player100", "player101", "player102", "player103" YIELD player.age AS age, player.name AS name | ORDER BY $-.age ASC, $-.name DESC`	`ORDER BY` 子句指定输出结果的排序规则。
RETURN	`RETURN {<vertex_name>|<edge_name>| <vertex_name>.<property>| <edge_name>.<property>|...}`	`MATCH (v:player) RETURN v.player.name, v.player.age LIMIT 3`	返回点的属性为 `name` 和 `age` 的前三行值。
TTL	`CREATE TAG <tag_name>(<property_name_1> <property_value_1>, <property_name_2> <property_value_2>, ...) ttl_duration= <value_int>, ttl_col = <property_name>`	`CREATE TAG t2(a int, b int, c string) ttl_duration= 100, ttl_col = "a"`	创建 Tag 并设置 TTL 选项。
WHERE	`WHERE {<vertex|edge_alias>.<property_name> {>|==|<|...} <value>...}`	`MATCH (v:player) WHERE v.player.name == "Tim Duncan" XOR (v.player.age < 30 AND v.player.name == "Yao Ming") OR NOT (v.player.name == "Yao Ming" OR v.player.name == "Tim Duncan") RETURN v.player.name, v.player.age`	`WHERE` 子句可以根据条件过滤输出结果，通常用于 `GO` 和 `LOOKUP` 语句， `MATCH` 和 `WITH` 语句。
YIELD	`YIELD [DISTINCT] <col> [AS <alias>] [, <col> [AS <alias>] ...] [WHERE <conditions>];`	`GO FROM "player100" OVER follow YIELD dst(edge) AS ID | FETCH PROP ON player $-.ID YIELD player.age AS Age \| YIELD AVG($-.Age) as Avg_age, count(*)as Num_friends`	查找 player100 关注的 player，并计算他们的平均年龄。
WITH	`MATCH $expressions WITH {nodes()| labels()|...}`	`MATCH p=(v:player{name:"Tim Duncan"})--() WITH nodes(p) AS n UNWIND n AS n1 RETURN DISTINCT n1`	`WITH` 子句可以获取并处理查询前半部分的结果，并将处理结果作为输入传递给查询的后半部分。
UNWIND	`UNWIND <list> AS <alias> <RETURN clause>`	`UNWIND [1,2,3] AS n RETURN n`	拆分列表。

Copyright © Yueshu Graph Technology - All rights reserved.

4.3.4 图空间语句

语句	语法	示例	说明
CREATE SPACE	`CREATE SPACE [IF NOT EXISTS] <graph_space_name> ( [partition_num = <partition_number>,] [replica_factor = <replica_number>,] vid_type = {FIXED_STRING(<N>) | INT[64]} ) [COMMENT = '<comment>']`	`CREATE SPACE my_space_1 (vid_type=FIXED_STRING(30))`	创建一个新的图空间。
CREATE SPACE	`CREATE SPACE <new_graph_space_name> AS <old_graph_space_name>`	`CREATE SPACE my_space_4 as my_space_3`	克隆现有图空间的 Schema。
USE	`USE <graph_space_name>`	`USE space1`	指定一个图空间，或切换到另一个图空间，将其作为后续查询的工作空间。
SHOW SPACES	`SHOW SPACES`	`SHOW SPACES`	列出悦数图数据库示例中的所有图空间。
DESCRIBE SPACE	`DESC[RIBE] SPACE <graph_space_name>`	`DESCRIBE SPACE basketballplayer`	显示指定图空间的信息。
CLEAR SPACE	`CLEAR SPACE [IF EXISTS] <graph_space_name>`	清空图空间中的点和边，但不会删除图空间本身以及其中的 Schema 信息。	
DROP SPACE	`DROP SPACE [IF EXISTS] <graph_space_name>`	`DROP SPACE basketballplayer`	删除指定图空间的所有内容。

4.3.5 TAG 语句

语句	语法	示例	说明
CREATE TAG	`CREATE TAG [IF NOT EXISTS] <tag_name> (<prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>'] [{, <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']} ...]) [TTL_DURATION = <ttl_duration>] [TTL_COL = <prop_name>] [COMMENT = '<comment>']`	`CREATE TAG woman(name string, age int, married bool, salary double, create_time timestamp) TTL_DURATION = 100, TTL_COL = "create_time"`	通过指定名称创建一个 Tag。
DROP TAG	`DROP TAG [IF EXISTS] <tag_name>`	`DROP TAG test;`	删除当前工作空间内所有点上的指定 Tag。
ALTER TAG	`ALTER TAG <tag_name> <alter_definition> [, alter_definition] ...] [ttl_definition [, ttl_definition] ... ] [COMMENT = '<comment>']`	`ALTER TAG t1 ADD (p3 int, p4 string)`	修改 Tag 的结构。例如增删属性、修改数据类型，也可以为属性设置、修改 TTL（Time-To-Live）。
SHOW TAGS	`SHOW TAGS`	`SHOW TAGS`	显示当前图空间内的所有 Tag 名称。
DESCRIBE TAG	`DESC[RIBE] TAG <tag_name>`	`DESCRIBE TAG player`	查看指定 Tag 的详细信息，例如字段名称、数据类型等。
DELETE TAG	`DELETE TAG <tag_name_list> FROM <VID>`	`DELETE TAG test1 FROM "test"`	删除指定点上的指定 Tag。

Copyright © Yueshu Graph Technology - All rights reserved.

4.3.6 Edge type 语句

语句	语法	示例	说明
CREATE EDGE	`CREATE EDGE [IF NOT EXISTS] <edge_type_name> (<prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>'] [{, <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']} ...]) [TTL_DURATION = <ttl_duration>] [TTL_COL = <prop_name>] [COMMENT = '<comment>']`	`CREATE EDGE e1(p1 string, p2 int, p3 timestamp) TTL_DURATION = 100, TTL_COL = "p2"`	指定名称创建一个 Edge type。
DROP EDGE	`DROP EDGE [IF EXISTS] <edge_type_name>`	`DROP EDGE e1`	删除当前工作空间内的指定 Edge type。
ALTER EDGE	`ALTER EDGE <edge_type_name> <alter_definition> [, alter_definition] ...] [ttl_definition [, ttl_definition] ... ] [COMMENT = '<comment>']`	`ALTER EDGE e1 ADD (p3 int, p4 string)`	修改 Edge type 的结构。
SHOW EDGES	`SHOW EDGES`	`SHOW EDGES`	显示当前图空间内的所有 Edge type 名称。
DESCRIBE EDGE	`DESC[RIBE] EDGE <edge_type_name>`	`DESCRIBE EDGE follow`	查看指定 Edge type 的详细信息，例如字段名称、数据类型等。

4.3.7 点语句

语句	语法	示例	说明
INSERT VERTEX	`INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...] VALUES <vid>: ([prop_value_list])`	`INSERT VERTEX t2 (name, age) VALUES "13":("n3", 12), "14": ("n4", 8)`	在悦数图数据库实例的指定图空间中插入一个或多个点。
DELETE VERTEX	`DELETE VERTEX <vid> [, <vid> ...]`	`DELETE VERTEX "team1"`	删除点，以及点关联的出边和入边。
UPDATE VERTEX	`UPDATE VERTEX ON <tag_name> <vid> SET <update_prop> [WHEN <condition>] [YIELD <output>]`	`UPDATE VERTEX ON player "player101" SET age = age + 2`	修改点上 Tag 的属性值。
UPSERT VERTEX	`UPSERT VERTEX ON <tag> <vid> SET <update_prop> [WHEN <condition>] [YIELD <output>]`	`UPSERT VERTEX ON player "player667" SET age = 31`	结合 `UPDATE` 和 `INSERT`，如果点存在，会修改点的属性值；如果点不存在，会插入新的点。

Copyright © Yueshu Graph Technology - All rights reserved.

4.3.8 边语句

语句	语法	示例	说明
INSERT EDGE	`INSERT EDGE [IF NOT EXISTS] <edge_type> ( <prop_name_list> ) VALUES <src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> ) [, <src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> ), ...]`	`INSERT EDGE e2 (name, age) VALUES "11"->"13": ("n1", 1)`	在悦数图数据库实例的指定图空间中插入一条或多条边。
DELETE EDGE	`DELETE EDGE <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid>[@<rank>] ...]`	`DELETE EDGE serve "player100" -> "team204"@0`	删除边。一次可以删除一条或多条边。
UPDATE EDGE	`UPDATE EDGE ON <edge_type> <src_vid> -> <dst_vid> [@<rank>] SET <update_prop> [WHEN <condition>] [YIELD <output>]`	`UPDATE EDGE ON serve "player100" -> "team204"@0 SET start_year = start_year + 1`	修改边上 Edge type 的属性。
UPSERT EDGE	`UPSERT EDGE ON <edge_type> <src_vid> -> <dst_vid> [@rank] SET <update_prop> [WHEN <condition>] [YIELD <properties>]`	`UPSERT EDGE on serve "player666" -> "team200"@0 SET end_year = 2021`	结合 `UPDATE` 和 `INSERT`，如果边存在，会更新边的属性；如果边不存在，会插入新的边。

Copyright © Yueshu Graph Technology - All rights reserved.

4.3.9 索引

- 原生索引

  索引配合 `LOOKUP` 和 `MATCH` 语句使用。

语句	语法	示例	说明
CREATE INDEX	`CREATE {TAG | EDGE} INDEX [IF NOT EXISTS] <index_name> ON {<tag_name> | <edge_name>} ([<prop_name_list>]) [COMMENT = '<comment>']`	`CREATE TAG INDEX player_index on player()`	对 Tag、EdgeType 或其属性创建原生索引。
SHOW CREATE INDEX	`SHOW CREATE {TAG | EDGE} INDEX <index_name>`	`show create tag index index_2`	创建 Tag 或者 Edge type 时使用的 nGQL 语句，其中包含索引的详细信息，例如其关联的属性。
SHOW INDEXES	`SHOW {TAG | EDGE} INDEXES`	`SHOW TAG INDEXES`	列出当前图空间内的所有 Tag 和 Edge type（包括属性）的索引。
DESCRIBE INDEX	`DESCRIBE {TAG | EDGE} INDEX <index_name>`	`DESCRIBE TAG INDEX player_index_0`	查看指定索引的信息，包括索引的属性名称（Field）和数据类型（Type）。
REBUILD INDEX	`REBUILD {TAG | EDGE} INDEX [<index_name_list>]`	`REBUILD TAG INDEX single_person_index`	重建索引。索引功能不会自动对其创建之前已存在的存量数据生效。
SHOW INDEX STATUS	`SHOW {TAG | EDGE} INDEX STATUS`	`SHOW TAG INDEX STATUS`	查看索引名称和对应的状态。
DROP INDEX	`DROP {TAG | EDGE} INDEX [IF EXISTS] <index_name>`	`DROP TAG INDEX player_index_0`	删除当前图空间中已存在的索引。

- 全文索引

语法	示例	说明
`SIGN IN TEXT SERVICE [(<elastic_ip:port> [,<username>, <password>]), (<elastic_ip:port>), ...]`	`SIGN IN TEXT SERVICE (127.0.0.1:9200)`	悦数图数据库的全文索引是基于 Elasticsearch 实现，部署 Elasticsearch 集群之后，可以使用 `SIGN IN` 语句登录 Elasticsearch 客户端。
`SHOW TEXT SEARCH CLIENTS`	`SHOW TEXT SEARCH CLIENTS`	列出文本搜索客户端。
`SIGN OUT TEXT SERVICE`	`SIGN OUT TEXT SERVICE`	退出所有文本搜索客户端。
`CREATE FULLTEXT {TAG | EDGE} INDEX <index_name> ON {<tag_name> |  <edge_name>} ([<prop_name_list>])`	`CREATE FULLTEXT TAG INDEX nebula_index_1 ON player(name)`	创建全文索引。
`SHOW FULLTEXT INDEXES`	`SHOW FULLTEXT INDEXES`	显示全文索引。
`REBUILD FULLTEXT INDEX`	`REBUILD FULLTEXT INDEX`	重建全文索引。
`DROP FULLTEXT INDEX <index_name>`	`DROP FULLTEXT INDEX nebula_index_1`	删除全文索引。
`LOOKUP ON {<tag> | <edge_type>} WHERE <expression> [YIELD <return_list>]`	`LOOKUP ON player WHERE FUZZY(player.name, "Tim Dunncan", AUTO, OR) YIELD player.name`	使用查询选项。

Copyright © Yueshu Graph Technology - All rights reserved.

4.3.10 子图和路径

类型	语法	示例	说明
子图	`GET SUBGRAPH [WITH PROP] [<step_count> {STEP|STEPS}] FROM {<vid>, <vid>...} [{IN | OUT | BOTH} <edge_type>, <edge_type>...] YIELD [VERTICES AS <vertex_alias>] [,EDGES AS <edge_alias>]`	`GET SUBGRAPH 1 STEPS FROM "player100" YIELD VERTICES AS nodes, EDGES AS relationships`	指定 Edge type 的起始点可以到达的点和边的信息，返回子图信息。
路径	`FIND { SHORTEST | ALL | NOLOOP } PATH [WITH PROP] FROM <vertex_id_list> TO <vertex_id_list> OVER <edge_type_list> [REVERSELY | BIDIRECT] [<WHERE clause>] [UPTO <N> {STEP|STEPS}] YIELD path as <alias> [| ORDER BY $-.path] [| LIMIT <M>]`	`FIND SHORTEST PATH FROM "player102" TO "team204" OVER * YIELD path as p`	查找指定起始点和目的点之间的路径。返回的路径格式类似于 `(<vertex_id>)-[:<edge_type_name>@<rank>]->(<vertex_id)` 。

4.3.11 查询调优

类型	语法	示例	说明
EXPLAIN	`EXPLAIN [format="row" | "dot"] <your_nGQL_statement>`	`EXPLAIN format="row" SHOW TAGS` `EXPLAIN format="dot" SHOW TAGS`	输出 nGQL 语句的执行计划，但不会执行 nGQL 语句。
PROFILE	`PROFILE [format="row" | "dot"] <your_nGQL_statement>`	`PROFILE format="row" SHOW TAGS` `EXPLAIN format="dot" SHOW TAGS`	执行 nGQL 语句，然后输出执行计划和执行概要。

4.3.12 运维

- SUBMIT JOB BALANCE

语法	说明
`SUBMIT JOB BALANCE LEADER`	启动任务均衡分布所有图空间中的 leader。该命令会返回任务 ID。

Copyright © Yueshu Graph Technology - All rights reserved.

| `SUBMIT JOB BALANCE DATA` |启动任务均衡分布当前图空间中的所有分片。该命令会返回任务 ID（`job_id`）。| |
`SUBMIT JOB BALANCE DATA REMOVE <ip:port> [,<ip>:<port> ...]` |启动任务迁空当前图空间指定的 Storage 服务中的分片。端口默认为 `9779`。|

- 作业管理

语法	说明
`SUBMIT JOB COMPACT`	触发 RocksDB 的长耗时 `compact` 操作。
`SUBMIT JOB FLUSH`	将内存中的 RocksDB memfile 写入硬盘。
`SUBMIT JOB STATS`	启动一个作业，该作业对当前图空间进行统计。作业完成后，用户可以使用 `SHOW STATS` 语句列出统计结果。
`SHOW JOB <job_id>`	显示当前图空间内指定作业和相关任务的信息。Meta 服务将 `SUBMIT JOB` 请求解析为多个任务，然后分配给进程 nebula-storaged。
`SHOW JOBS`	列出当前图空间内所有未过期的作业。
`STOP JOB`	停止当前图空间内未完成的作业。
`RECOVER JOB`	重新执行当前图空间内失败的作业，并返回已恢复的作业数量。

- 终止查询

语法	示例	说明
`KILL QUERY (session=<session_id>, plan=<plan_id>)`	`KILL QUERY(SESSION=1625553545984255,PLAN=163)`	在一个会话中执行命令终止另一个会话中的查询。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5. nGQL 指南

5.1 nGQL 概述

5.1.1 什么是 nGQL

nGQL是悦数图数据库使用的的声明式图查询语言，支持灵活高效的图模式，而且 nGQL 是为开发和运维人员设计的类 SQL 查询语言，易于学习。

nGQL 是一个进行中的项目，会持续发布新特性和优化，因此可能会出现语法和实际操作不一致的问题，如果遇到此类问题，请提交 issue 通知悦数图数据库团队。悦数图数据库 3.0 及更新版本正在支持 openCypher 9。

nGQL 可以做什么

- 支持图遍历
- 支持模式匹配
- 支持聚合
- 支持修改图
- 支持访问控制
- 支持聚合查询
- 支持索引
- 支持大部分 openCypher 9 图查询语法（不支持修改和控制语法）

示例数据 **Basketballplayer**

用户可以下载悦数图数据库示例数据 basketballplayer 文件，然后使用 NebulaGraph Console，使用选项 `-f` 执行脚本。

> **Note**
>
> 导入示例数据前，确保已执行 `ADD HOSTS` 命令将 Storage 主机增加至集群中。更多信息，请参见管理 Storage 主机。

占位标识符和占位符值

悦数图数据库查询语言 nGQL 参照以下标准设计：

- (Draft) ISO/IEC JTC1 N14279 SC 32 - Database_Languages - GQL
- (Draft) ISO/IEC JTC1 SC32 N3228 - SQL_Property_Graph_Queries - SQLPGQ
- OpenCypher 9

在模板代码中，任何非关键字、字面值或标点符号的标记都是占位符标识符或占位符值。

Copyright © Yueshu Graph Technology - All rights reserved.

本文中 nGQL 语法符号的说明如下。

符号	含义
< >	语法元素的名称。
:	定义元素的公式。
[ ]	可选元素。
{ }	显式的指定元素。
|	所有可选的元素。
...	可以重复多次。

例如创建点的 nGQL 语法:

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES <vid>: ([prop_value_list])

tag_props:
  tag_name ([prop_name_list])

prop_name_list:
   [prop_name [, prop_name] ...]

prop_value_list:
   [prop_value [, prop_value] ...]
```

示例语句:

```
nebula> CREATE TAG IF NOT EXISTS player(name string, age int);
```

关于 openCypher 兼容性

原生 NGQL 和 OPENCYPHER 的关系

原生 nGQL 是由悦数图数据库自行创造和实现的图查询语言。openCypher 是由 openCypher Implementers Group 组织所开源和维护的图查询语言，最新版本为 openCypher 9。

由于 nGQL 语言部分兼容了 openCypher，这个部分在本文中称为 openCypher 兼容语句。

> **Note**
>
> `nGQL 语言` = `原生 nGQL 语句` + `openCypher 兼容语句`

NGQL 完全兼容 OPENCYPHER 9 吗?

不。

> **openCypher 兼容性**
>
> nGQL 设计目标仅为兼容部分的 DQL 语句(match, optional match, with等)。
>
> 不计划兼容任何 DDL，DML，DCL;
>
> 不计划兼容 Bolt 协议;
>
> 不计划兼容 APOC 与 GDS。
>
> 在本文搜索 "compatibility" 或者 “兼容性” 查看具体不兼容的细节。
>
> 在 Issues 中已经列出已知的兼容错误。如果发现这种类型的新问题，请提交问题并附带 `incompatible` 标签。

Copyright © Yueshu Graph Technology - All rights reserved.

NGQL 和 OPENCYPHER 9 的 主要差异有哪些?

类别	**openCypher 9**	**nGQL**
Schema	弱 Schema	强 Schema
相等运算符	`=`	`==`
数学求幂	`^`	使用 `pow(x, y)` 替代 `^` 。
边 Rank	无此概念	用 `@rank` 设置。
语句	-	不支持 openCypher 9 的所有 DML 语句（如 `CREATE` 、 `MERGE` 等），不支持所有的 DCL，和支持部分 `MATCH` , `OPTIONAL MATCH` 语法和函数。
语句文本换行	换行符	`\` + 换行符
Label 与 Tag 是不同的概念	Label 用于寻找点（点的索引）。	Tag 用于定义点的一种类型及相应的属性，无索引功能。
预编译与参数化查询	支持	仅支持参数化查询。

Compatibility

请注意 openCypher 9 和 Cypher 在语法和许可上有不同：

1. Cypher 要求所有 Cypher 语句必须“显式地在一个事务中”执行，而 openCypher 没有这样的要求。另外，nGQL 没有事务及隔离性。
2. Cypher 企业版功能有多种的约束（constraints），包括 Unique node property constraints、Node property existence constraints、Relationship property existence constraints、Node key constraints。 OpenCypher 标准中没有约束。 而 nGQL 是强 Schema 系统，前述的约束大多通过 Schema 定义可实现（包括 NOT NULL），唯一不能支持的功能是“属性值唯一性”（UNIQUE constraint）。
3. Cypher 有 APoC，openCypher 9 没有 APoC。Cypher 有 Bolt 协议支持要求， openCypher 9 没有。

哪里可以找到更多 NGQL 的示例?

用户可以在 GitHub 的 features 目录内查看超过 2500 条 nGQL 示例。

features 目录内包含很多。features 格式的文件，每个文件都记录了使用 nGQL 的场景和示例。例如：

```
Feature: Basic match

  Background:
    Given a graph with space named "basketballplayer"

  Scenario: Single node
    When executing query:
      """
      MATCH (v:player {name: "Yao Ming"}) RETURN v;
      """
    Then the result should be, in any order, with relax comparison:
      | v                                      |
      | ("player133" :player{age: 38, name: "Yao Ming"}) |

  Scenario: One step
    When executing query:
      """
      MATCH (v1:player{name: "LeBron James"}) -[r]-> (v2)
      RETURN type(r) AS Type, v2.player.name AS Name
      """
    Then the result should be, in any order:

      | Type     | Name        |
      | "follow" | "Ray Allen" |
      | "serve"  | "Lakers"    |
      | "serve"  | "Heat"      |
      | "serve"  | "Cavaliers" |

Feature:  Comparison of where clause

  Background:
    Given a graph with space named "basketballplayer"

    Scenario: push edge props filter down
      When profiling query:
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    """
    GO FROM "player100" OVER follow
    WHERE properties(edge).degree IN [v IN [95,99] WHERE v > 0]
    YIELD dst(edge), properties(edge).degree
    """
  Then the result should be, in any order:
    | follow._dst | follow.degree |
    | "player101" | 95            |
    | "player125" | 95            |
  And the execution plan should be:
    | id | name         | dependencies | operator info                                                    |
    | 0  | Project      | 1            |                                                                  |
    | 1  | GetNeighbors | 2            | {"filter": "(properties(edge).degree IN [v IN [95,99] WHERE (v>0)])"} |
    | 2  | Start        |              |                                                                  |
```

示例中的关键字说明如下。

关键字	说明
`Feature`	描述当前文档的主题。
`Background`	描述当前文档的背景信息。
`Given`	描述执行示例语句的前提条件。
`Scenario`	描述具体场景。如果场景之前有 `@skip` 标识，表示这个场景下示例语句可能无法正常工作，请不要在生产环境中使用该示例语句。
`When`	描述要执行的 nGQL 示例语句。可以是 `executing query` 或 `profiling query` 。
`Then`	描述执行 `When` 内语句的预期返回结果。如果返回结果和文档不同，请提交 issue 通知悦数图数据库团队。
`And`	描述执行 `When` 内语句的副作用或执行计划。
`@skip`	跳过这个示例。通常表示测试代码还没有准备好。

欢迎增加更多 tck case，在 CI/CD 中自动回归所使用的语句。

是否支持 TINKERPOP GREMLIN?

不支持。也没有计划。

是否支持 W3C 的 RDF（SPARQL） 或 GRAPHQL 等?

不支持。也没有计划。

悦数图数据库的数据模型是属性图，是一个强 Schema 系统，不支持 RDF 标准。

nGQL 也不支持 SPARQL 和 GraphQL。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.1.2 模式

模式（pattern）和图模式匹配，是图查询语言的核心功能，本文介绍悦数图数据库设计的各种模式，部分还未实现。

单点模式

点用一对括号来描述，通常包含一个名称。例如：

```
(a)
```

示例为一个简单的模式，描述了单个点，并使用变量 `a` 命名该点。

多点关联模式

多个点通过边相连是常见的结构，模式用箭头来描述两个点之间的边。例如：

```
(a)-[]->(b)
```

示例为一个简单的数据结构：两个点和一条连接两个点的边，两个点分别为 `a` 和 `b` ，边是有方向的，从 `a` 到 `b` 。

这种描述点和边的方式可以扩展到任意数量的点和边，例如：

```
(a)-[]->(b)<-[]-(c)
```

这样的一系列点和边称为 `路径` （path）。

只有在涉及某个点时，才需要命名这个点。如果不涉及这个点，则可以省略名称，例如：

```
(a)-[]->()<-[]-(c)
```

Tag 模式

> **Note**
>
> nGQL 中的 `Tag` 概念与 openCypher 中的 `Label` 有一些不同。例如，必须创建一个 `Tag` 之后才能使用它，而且 `Tag` 还定义了属性的类型。

模式除了简单地描述图中的点之外，还可以描述点的 Tag。例如：

```
(a:User)-[]->(b)
```

模式也可以描述有多个 Tag 的点，例如：

```
(a:User:Admin)-[]->(b)
```

属性模式

点和边是图的基本结构。nGQL 在这两种结构上都可以增加属性，方便实现更丰富的模型。

在模式中，属性的表示方式为：用花括号括起一些键值对，用英文逗号分隔，并且需要指定属性所属的 Tag 或者 Edge type。

例如一个点有两个属性：

```
(a:player{name: "Tim Duncan", age: 42})
```

在这个点上可以有一条边是：

```
(a)-[e:follow{degree: 95}]->(b)
```


Copyright © Yueshu Graph Technology - All rights reserved.

边模式

描述一条边最简单的方法是使用箭头连接两个点。

可以用以下方式描述边以及它的方向性。如果不关心边的方向，可以省略箭头，例如：

```
(a)-[]-(b)
```

和点一样，边也可以命名。一对方括号用于分隔箭头，变量放在两者之间。例如：

```
(a)-[r]->(b)
```

和点上的 Tag 一样，边也可以有类型。描述边的类型，例如：

```
(a)-[r:REL_TYPE]->(b)
```

和点上的 Tag 不同，一条边只能有一种 Edge type。但是如果我们想描述多个可选 Edge type，可以用管道符号（|）将可选值分开，例如：

```
(a)-[r:TYPE1|TYPE2]->(b)
```

和点一样，边的名称可以省略，例如：

```
(a)-[:REL_TYPE]->(b)
```

变长模式

在图中指定边的长度来描述多条边（以及中间的点）组成的一条长路径，不需要使用多个点和边来描述。例如：

```
(a)-[*2]->(b)
```

该模式描述了 3 点 2 边组成的图，它们都在一条路径上（长度为 2），等价于：

```
(a)-[]->()-[]->(b)
```

也可以指定长度范围，这样的边模式称为 `variable-length edges`，例如：

```
(a)-[*3..5]->(b)
```

`*3..5` 表示最小长度为 3，最大长度为 5。

该模式描述了 4 点 3 边、5 点 4 边或 6 点 5 边组成的图。

也可以忽略最小长度，只指定最大长度，例如：

```
(a)-[*..5]->(b)
```

Note

必须指定最大长度，不支持仅指定最小长度（`(a)-[*3..]->(b)`）或都不指定（`(a)-[*]->(b)`）。

路径变量

一系列连接的点和边称为 `路径` 。nGQL 允许使用变量来命名路径，例如：

```
p = (a)-[*3..5]->(b)
```

可以在 MATCH 语句中使用路径变量。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.1.3 注释

本文介绍 nGQL 中的注释方式。

Examples

```
nebula> # 这行什么都不做。
nebula> RETURN 1+1;     # 这条注释延续到行尾。
nebula> RETURN 1+1;     // 这条注释延续到行尾。
nebula> RETURN 1 /* 这是一条行内注释 */ + 1 == 2;
nebula> RETURN 11 +            \
/* 多行注释       \
用反斜线来换行。   \
*/ 12;
```

nGQL 语句中的反斜线（\）代表换行。

OpenCypher 兼容性

- 在 nGQL 中，用户必须在行末使用反斜线（\）来换行，即使是在使用 `/* */` 符号的多行注释内。
- 在 openCypher 中不需要使用反斜线换行。

```
/* openCypher 风格：
这条注释
延续了不止
一行 */
MATCH (n:label)
RETURN n;
```

```
/* 原生 nGQL 风格：  \
这条注释       \
延续了不止     \
一行 */       \
MATCH (n:tag) \
RETURN n;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.1.4 大小写区分

标识符区分大小写

以下语句会出现错误，因为 `my_space` 和 `MY_SPACE` 是两个不同的图空间。

```
nebula> CREATE SPACE IF NOT EXISTS my_space (vid_type=FIXED_STRING(30));
nebula> use MY_SPACE;
[ERROR (-1005)]: SpaceNotFound:
```

关键字不区分大小写

以下语句是等价的，因为 `show` 和 `spaces` 是关键字。

```
nebula> show spaces;
nebula> SHOW SPACES;
nebula> SHOW spaces;
nebula> show SPACES;
```

函数不区分大小写

函数名称不区分大小写，例如 `count()` 、 `COUNT()` 、 `couNT()` 是等价的。

```
nebula> WITH [NULL, 1, 1, 2, 2] As a \
        UNWIND a AS b \
        RETURN count(b), COUNT(*), couNT(DISTINCT b);
+----------+----------+------------------+
| count(b) | COUNT(*) | couNT(distinct b) |
+----------+----------+------------------+
| 4        | 5        | 2                |
+----------+----------+------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.1.5 关键字

关键字在 nGQL 中有重要意义，分为保留关键字和非保留关键字。建议不要在 Schema 中使用关键字。

如果必须使用关键字：

- 当非保留关键字均为小写字母时，如果将其作为标识符时可以不使用引号；当非保留关键字包含大写字母时并将其作为标识符时，需要用反引号（`）包围，例如 `Comment`。
- 保留关键字或特殊字符作为标识符时，需要用反引号（`）包围，例如 `AND`。

Note

关键字不区分大小写。

```
nebula> CREATE TAG TAG(name string);
[ERROR (-1004)]: SyntaxError: syntax error near `TAG'

nebula> CREATE TAG `TAG` (name string);
Execution succeeded

nebula> CREATE TAG SPACE(name string);
Execution succeeded

nebula> CREATE TAG 中文(简体 string);
Execution succeeded

nebula> CREATE TAG `￥%特殊 字符&*+-*/` (`q~！（）=  wer` string);
Execution succeeded
```

保留关键字

```
ACROSS
ADD
ALTER
AND
AS
ASC
ASCENDING
BALANCE
BOOL
BY
CASE
CHANGE
COMPACT
CREATE
DATE
DATETIME
DELETE
DESC
DESCENDING
DESCRIBE
DISTINCT
DOUBLE
DOWNLOAD
DROP
DURATION
EDGE
EDGES
EXISTS
EXPLAIN
FETCH
FIND
FIXED_STRING
FLOAT
FLUSH
FORMAT
FROM
GET
GO
GRANT
IF
IGNORE_EXISTED_INDEX
IN
INDEX
INDEXES
INGEST
INSERT
INT
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
INT16
INT32
INT64
INT8
INTERSECT
IS
LIMIT
LIST
LOOKUP
MAP
MATCH
MINUS
NO
NOT
NOT_IN
NULL
OF
OFFSET
ON
OR
ORDER
OVER
OVERWRITE
PROFILE
PROP
REBUILD
RECOVER
REMOVE
RESTART
RETURN
REVERSELY
REVOKE
SET
SHOW
STEP
STEPS
STOP
STRING
SUBMIT
TAG
TAGS
TIME
TIMESTAMP
TO
UNION
UPDATE
UPSERT
UPTO
USE
VERTEX
VERTICES
WHEN
WHERE
WITH
XOR
YIELD
```

非保留关键字

```
ACCOUNT
ADMIN
ALL
ANY
ATOMIC_EDGE
AUTO
BIDIRECT
BOTH
CHARSET
CLIENTS
COLLATE
COLLATION
COMMENT
CONFIGS
CONTAINS
DATA
DBA
DEFAULT
ELASTICSEARCH
ELSE
END
ENDS
ENDS_WITH
FORCE
FULLTEXT
FUZZY
GOD
GRAPH
GROUP
GROUPS
GUEST
HDFS
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
HOST
HOSTS
INTO
IS_EMPTY
IS_NOT_EMPTY
IS_NOT_NULL
IS_NULL
JOB
JOBS
KILL
LEADER
LISTENER
META
NOLOOP
NONE
NOT_CONTAINS
NOT_ENDS_WITH
NOT_STARTS_WITH
OPTIONAL
OUT
PART
PARTITION_NUM
PARTS
PASSWORD
PATH
PLAN
PREFIX
QUERIES
QUERY
REDUCE
REGEXP
REPLICA_FACTOR
RESET
ROLE
ROLES
SAMPLE
SEARCH
SERVICE
SESSION
SESSIONS
SHORTEST
SIGN
SINGLE
SKIP
SNAPSHOT
SNAPSHOTS
SPACE
SPACES
STARTS
STARTS_WITH
STATS
STATUS
STORAGE
SUBGRAPH
TEXT
TEXT_SEARCH
THEN
TOP
TTL_COL
TTL_DURATION
UNWIND
USER
USERS
UUID
VALUE
VALUES
VID_TYPE
WILDCARD
ZONE
ZONES
FALSE
TRUE
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.1.6 nGQL 风格指南

nGQL 没有严格的构建格式要求，但根据恰当而统一的风格创建 nGQL 语句有利于提高可读性、避免歧义。在同一组织或项目中使用相同的 nGQL 风格有利于降低维护成本，规避因格式混乱或误解造成的问题。本文为写作 nGQL 语句提供了风格参考。

> **Compatibility**
>
> nGQL 风格与 Cypher Style Guide 不同。

换行

1. 换行写子句。

   不推荐：

```
GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS id;
```

   推荐：

```
GO FROM "player100" \
OVER follow REVERSELY \
YIELD src(edge) AS id;
```

2. 换行写复合语句中的不同语句。

   不推荐：

```
GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS id | GO FROM $-.id \
OVER serve WHERE properties($^).age > 20 YIELD properties($^).name AS FriendOf, properties($$).name AS Team;
```

   推荐：

```
GO FROM "player100" \
OVER follow REVERSELY \
YIELD src(edge) AS id | \
GO FROM $-.id OVER serve \
WHERE properties($^).age > 20 \
YIELD properties($^).name AS FriendOf, properties($$).name AS Team;
```

3. 子句长度超过 80 个字符时，在合适的位置换行。

   不推荐：

```
MATCH (v:player{name:"Tim Duncan"})-[e]->(v2) \
WHERE (v2.player.name STARTS WITH "Y" AND v2.player.age > 35 AND v2.player.age < v.player.age) OR (v2.player.name STARTS WITH "T" AND v2.player.age < 45 AND v2.player.age > v.player.age) \
RETURN v2;
```

   推荐：

```
MATCH (v:player{name:"Tim Duncan"})-[e]->(v2) \
WHERE (v2.player.name STARTS WITH "Y" AND v2.player.age > 35 AND v2.player.age < v.player.age) \
OR (v2.player.name STARTS WITH "T" AND v2.player.age < 45 AND v2.player.age > v.player.age) \
RETURN v2;
```

> **Note**
>
> 即使子句不超过 80 个字符，如需换行后有助于理解，也可将子句再次分行。

Copyright © Yueshu Graph Technology - All rights reserved.

标识符命名

在 nGQL 语句中，关键字、标点符号、空格以外的字符内容都是标识符。推荐的标识符命名方式如下。

1. 使用单数名词命名 Tag，用原型动词或动词短语构成 Edge type。

   不推荐：

```
MATCH p=(v:players)-[e:are_following]-(v2) \
RETURN nodes(p);
```

   推荐：

```
MATCH p=(v:player)-[e:follow]-(v2) \
RETURN nodes(p);
```

2. 标识符用蛇形命名法，以下划线（_）连接单词，且所有字母小写。

   不推荐：

```
MATCH (v:basketballTeam) \
RETURN v;
```

   推荐：

```
MATCH (v:basketball_team) \
RETURN v;
```

3. 语法关键词大写，变量小写。

   不推荐：

```
match (V:player) return V limit 5;
```

   推荐：

```
MATCH (v:player) RETURN v LIMIT 5;
```

Pattern

1. 分行写 Pattern 时，在表示边的箭头右侧换行，而不是左侧。

   不推荐：

```
MATCH (v:player{name: "Tim Duncan", age: 42}) \
-[e:follow]->()-[e2:serve]->()<--(v2) \
RETURN v, e, v2;
```

   推荐：

```
MATCH (v:player{name: "Tim Duncan", age: 42})-[e:follow]-> \
()-[e2:serve]->()<--(v2) \
RETURN v, e, v2;
```

2. 将无需查询的点和边匿名化。

   不推荐：

```
MATCH (v:player)-[e:follow]->(v2) \
RETURN v;
```

   推荐：

```
MATCH (v:player)-[:follow]->() \
RETURN v;
```

3. 将非匿名点放在匿名点的前面。

   不推荐：

Copyright © Yueshu Graph Technology - All rights reserved.

```
MATCH ()-[:follow]->(v) \
RETURN v;
```

推荐：

```
MATCH (v)<-[:follow]-() \
RETURN v;
```

字符串

字符串用双引号包围。

不推荐：

```
RETURN 'Hello Nebula!';
```

推荐：

```
RETURN "Hello Nebula!\"123\"";
```

Note

字符串中需要嵌套单引号或双引号时，用反斜线（\）转义。例如：

```
RETURN "\"The database is amazing,\" the user says.";
```

结束语句

1. 用英文分号（;）结束 nGQL 语句。

   不推荐：

```
FETCH PROP ON player "player100" YIELD properties(vertex)
```

   推荐：

```
FETCH PROP ON player "player100" YIELD properties(vertex);
```

2. 使用管道符（|）分隔的复合语句，仅在最后一行末用英文分号结尾。在管道符前使用英文分号会导致语句执行失败。

   不支持：

```
GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id; | \
GO FROM $-.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```

   支持：

```
GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id | \
GO FROM $-.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```

3. 在包含自定义变量的复合语句中，用英文分号结束定义变量的语句。不按规则加分号或使用管道符结束该语句会导致执行失败。

   不支持：

```
$var = GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id \
GO FROM $var.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```


Copyright © Yueshu Graph Technology - All rights reserved.

也不支持:

```
$var = GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id | \
GO FROM $var.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```

支持:

```
$var = GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id; \
GO FROM $var.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2 数据类型

5.2.1 数值

nGQL 支持整数和浮点数。

整数

nGQL 支持带符号的 64 位整数（INT64）、32 位整数（INT32）、16 位整数（INT16）和 8 位整数（INT8）。

类型	声明关键字	范围
INT64	`INT64` 或 `INT`	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
INT32	`INT32`	-2,147,483,648 ~ 2,147,483,647
INT16	`INT16`	-32,768 ~ 32,767
INT8	`INT8`	-128 ~ 127

浮点数

nGQL 支持单精度浮点（FLOAT）和双精度浮点（DOUBLE）。

类型	声明关键字	范围	精度
FLOAT	`FLOAT`	3.4E +/- 38	6~7 位
DOUBLE	`DOUBLE`	1.7E +/- 308	15~16 位

nGQL 支持科学计数法，例如 `1e2`、`1.1e2`、`.3e4`、`1.e4`、`-1234E-10`。

Note

不支持 MySQL 中的 DECIMAL 数据类型。

数值的读写

在写入和读取不同类型的数据时，nGQL 的行为遵守以下规则：

数值类型	设置为 **VID**	设置为属性类型	读取该类型的属性值得到的类型
INT64	支持	支持	INT64
INT32	不支持	支持	INT64
INT16	不支持	支持	INT64
INT8	不支持	支持	INT64
FLOAT	不支持	支持	DOUBLE
DOUBLE	不支持	支持	DOUBLE

Copyright © Yueshu Graph Technology - All rights reserved.

例如，nGQL 不支持设置 INT8 类型的 VID，但支持将 TAG 或 Edge type 的某个属性类型设置为 INT8。当使用 nGQL 语句读取 INT8 类型的属性时，获取到的值的类型为 INT64。

- 悦数图数据库支持写入多种进制的数值：
- 十进制，例如 `123456` 。
- 十六进制，例如 `0x1e240` 。
- 八进制，例如 `0361100` 。

  但悦数图数据库会将写入的非十进制数值解析为十进制的值保存。读取到的值为十进制。

  例如，属性 `score` 的类型为 `INT` ，通过 INSERT 语句为其赋值 `0xb` ，使用 FETCH 等语句查询该属性值获取到的结果是 `11` ，即将十六进制的 `0xb` 转换为十进制后的值。

- 将 FLOAT/DOUBLE 类型的数值插入 INT 类型的列，会将数值四舍五入取整。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.2 布尔

悦数图数据库使用关键字 `BOOL` 声明布尔数据类型，可选值为 `true` 或 `false` 。

nGQL 支持以如下方式使用布尔值:

- 将属性值的数据类型定义为布尔。
- 在 `WHERE` 子句中用布尔值作为判断条件。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.3 字符串

悦数图数据库支持定长字符串和变长字符串。

声明与表示方式

nGQL 中的字符串声明方式如下：

- 使用关键字 `STRING` 声明变长字符串。
- 使用关键字 `FIXED_STRING(<length>)` 声明定长字符串，`<length>` 为字符串长度，例如 `FIXED_STRING(32)` 。

字符串的表示方式为用双引号或单引号包裹，例如 `"Hello, Cooper"` 或 `'Hello, Cooper'` 。

字符串读写

nGQL 支持以如下方式使用字符串：

- 将 VID 的数据类型定义为定长字符串。
- 将变长字符串设置为 Schema 名称，包括图空间、Tag、Edge type 和属性的名称。
- 将属性值的数据类型定义为定长或变长字符串。

例如：

- 将属性值的类型定义为定长字符串

```
nebula> CREATE TAG IF NOT EXISTS t1 (p1 FIXED_STRING(10));
```

- 将属性值的类型定义为变长字符串

```
nebula> CREATE TAG IF NOT EXISTS t2 (p2 STRING);
```

如果尝试写入的定长字符串超出长度限制：

- 当该定长字符串为属性值时，写入会成功，悦数图数据库将截断字符串，仅存入符合长度限制的部分。
- 当该定长字符串为 VID 时，写入会失败，悦数图数据库将报错。

转义字符

字符串中不支持直接换行，可以使用转义字符实现，例如：

- `"\n\t\r\b\f"`
- `"\110ello world"`

OpenCypher 兼容性

openCypher、Cypher 和 nGQL 之间有一些细微区别，例如下面 openCypher 的示例，不能将单引号替换为双引号。

```
# File: Literals.feature
Feature: Literals

Background:
    Given any graph
 Scenario: Return a single-quoted string
    When executing query:
      """
      RETURN '' AS literal
      """
    Then the result should be, in any order:
      | literal |
      | ''      |    # Note: it should return single-quotes as openCypher required.
    And no side effects
```


Copyright © Yueshu Graph Technology - All rights reserved.

Cypher 的返回结果同时支持单引号和双引号，nGQL 遵循 Cypher 的方式。

```
nebula > YIELD '' AS quote1, "" AS quote2, "'" AS quote3, '"' AS quote4
+--------+--------+--------+--------+
| quote1 | quote2 | quote3 | quote4 |
+--------+--------+--------+--------+
| ""     | ""     | "'"    | """    |
+--------+--------+--------+--------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.4 日期和时间类型

本文介绍日期和时间的类型，包括 `DATE` 、 `TIME` 、 `DATETIME` 、 `TIMESTAMP` 和 `DURATION` 。

注意事项

- 在插入时间类型的属性值时，悦数图数据库会根据配置文件中 `timezone_name` 参数指定的时区，将该 `DATE` 、 `TIME` 、 `DATETIME` 转换成相应的世界协调时间（UTC）时间。

  > **Note**
  >
  > 如需修改当前时区，请同时修改所有服务的配置文件中的 `timezone_name` 参数。

- 函数 `date()` 、 `time()` 和 `datetime()` 可以指定时区进行转换，例如 `datetime("2017-03-04 22:30:40.003000+08:00")` 或 `datetime("2017-03-04T22:30:40.003000[Asia/Shanghai]")` 。
- 函数 `date()` 、 `time()` 、 `datetime()` 和 `timestamp()` 可以用空值获取当前的日期或时间。
- 函数 `date()` 、 `time()` 、 `datetime()` 和 `duration()` 可以用属性名称获取自身的某一个具体属性值，例如 `date().month` 获取当前月份、 `time("02:59:40").minute` 获取传入时间的分钟数。
- 进行时间运算时建议使用 `duration()` 计算时刻的偏移。此外还支持 `date()` 和 `date()` 的加减、 `timestamp()` 和 `timestamp()` 的加减。
- 设置时间的年份为负数时，需要使用 Map 类型数据。

openCypher 兼容性

- 支持年、月、日、时、分、秒、毫秒、微秒，不支持纳秒。
- 不支持函数 `localdatetime()` 。
- 不支持大部分字符串时间格式，支持 `YYYY-MM-DDThh:mm:ss` 和 `YYYY-MM-DD hh:mm:ss` 。
- 支持单个数字的字符串时间格式，例如 `time("1:1:1")` 。

DATE

`DATE` 包含日期，但是不包含时间。悦数图数据库检索和显示 `DATE` 的格式为 `YYYY-MM-DD` 。支持的范围是 `-32768-01-01` 到 `32767-12-31` 。

`date()` 支持的属性名称包括 `year` 、 `month` 和 `day` 。 `date()` 支持输入 `YYYY` 、 `YYYY-MM` 或 `YYYY-MM-DD` ，未输入的月份或日期默认为 `01` 。

```
nebula> RETURN DATE({year:-123, month:12, day:3});
+-----------------------------------+
| date({year:-(123),month:12,day:3}) |
+-----------------------------------+
| -123-12-03                        |
+-----------------------------------+

nebula> RETURN DATE("23333");
+---------------+
| date("23333") |
+---------------+
| 23333-01-01   |
+---------------+

nebula> RETURN DATE("2023-12-12") - DATE("2023-12-11");
+----------------------------------------+
| (date("2023-12-12")-date("2023-12-11")) |
+----------------------------------------+
| 1                                      |
+----------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

TIME

`TIME` 包含时间，但是不包含日期。悦数图数据库检索和显示 `TIME` 的格式为 `hh:mm:ss.msmsmsususus` 。支持的范围是 `00:00:00.000000` 到 `23:59:59.999999` 。

`time()` 支持的属性名称包括 `hour` 、 `minute` 和 `second` 。

DATETIME

`DATETIME` 包含日期和时间。悦数图数据库检索和显示 `DATETIME` 的格式为 `YYYY-MM-DDThh:mm:ss.msmsmsususus` 。支持的范围是 `-32768-01-01T00:00:00.000000` 到 `32767-12-31T23:59:59.999999` 。

- `datetime()` 支持的属性名称包括 `year` 、 `month` 、 `day` 、 `hour` 、 `minute` 和 `second` 。
- `datetime()` 可将 `TIMESTAMP` 类型的日期值转换成 `DATETIME` 类型的日期值。 `TIMESTAMP` 类型的日期值取值范围： `0~9223372036` 。
- `datetime()` 支持 `int` 类型的参数，该 `int` 参数表示时间戳。

```
# 获取当前时间。
nebula> RETURN datetime();
+----------------------------+
| datetime()                 |
+----------------------------+
| 2022-08-29T06:37:08.933000 |
+----------------------------+

# 获取当前时间的小时。
nebula> RETURN datetime().hour;
+-----------------+
| datetime().hour |
+-----------------+
| 6               |
+-----------------+

# 将时间戳转换成 DATETIME 类型的格式。
nebula> RETURN datetime(timestamp(1625469277));
+---------------------------------+
| datetime(timestamp(1625469277)) |
+---------------------------------+
| 2021-07-05T07:14:37.000000      |
+---------------------------------+

nebula> RETURN datetime(1625469277);
+----------------------------+
| datetime(1625469277)       |
+----------------------------+
| 2021-07-05T07:14:37.000000 |
+----------------------------+
```

TIMESTAMP

`TIMESTAMP` 包含日期和时间。支持的范围是 UTC 时间的 `1970-01-01T00:00:01` 到 `2262-04-11T23:47:16` 。

`TIMESTAMP` 还有以下特点：

- 以时间戳形式存储和显示。例如 `1615974839` ，表示 `2021-03-17T17:53:59` 。
- 查询 `TIMESTAMP` 类型属性值的方式包括时间戳整数和 `timestamp()` 函数。
- 插入 `TIMESTAMP` 类型属性值的方式包括时间戳整数、 `timestamp()` 函数和 `now()` 函数。
- `timestamp()` 函数支持传入空值获取当前时间戳；同时支持传入整数以标识该整数为时间戳，整数取值： `0~9223372036` 。
- `timestamp()` 函数可将 `DATETIME` 类型的日期值转换成 `TIMESTAMP` 类型的日期值，且传入的 `DATETIME` 类型的日期值为 `string` 类型。
- 底层存储的数据格式为 **64** 位 **int**。

```
# 传入当前时间。
nebula> RETURN timestamp();
+-------------+
| timestamp() |
+-------------+
| 1625469277  |
+-------------+

# 传入指定时间。
nebula> RETURN timestamp("2022-01-05T06:18:43");
+---------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| timestamp("2022-01-05T06:18:43") |
+----------------------------------+
| 1641363523                       |
+----------------------------------+

# 传入 datetime()。
nebula> RETURN timestamp(datetime("2022-08-29T07:53:10.939000"));
+-------------------------------------------------+
| timestamp(datetime("2022-08-29T07:53:10.939000")) |
+-------------------------------------------------+
| 1661759590                                      |
+-------------------------------------------------+
```

Note

传入 `timestamp()` 函数的时间字符串不支持包含毫秒和微秒，但是通过 `timestamp(datetime())` 传入的时间字符串支持包含毫秒和微秒。

DURATION

`DURATION` 是一段连续的时间，由 `years` 、 `months` 、 `days` 、 `hours` 、 `minutes` 、 `seconds` 六个Key自由组合成的Map类型数据表示。例如 `duration({years: 12, months: 5, days: 14, hours: 16, minutes: 12, seconds: 70})` 。

`DURATION` 还有以下特点：

- 不支持为 `DURATION` 类型数据创建索引。
- 可以用于对指定时间进行计算。

示例

1. 创建 Tag，名称为 `date1` ，包含 `DATE` 、 `TIME` 和 `DATETIME` 三种类型。

```
nebula> CREATE TAG IF NOT EXISTS date1(p1 date, p2 time, p3 datetime);
```

2. 插入点，名称为 `test1` 。

```
nebula> INSERT VERTEX date1(p1, p2, p3) VALUES "test1":(date("2021-03-17"), time("17:53:59"), datetime("2017-03-04T22:30:40.003000[Asia/Shanghai]"));
```

3. 查询 `test1` 的属性 `p1` 是否为 `2021-03-17` 。

```
nebula> MATCH (v:date1) RETURN v.date1.p1 == date("2021-03-17");
+---------------------------------+
| (v.date1.p1==date("2021-03-17")) |
+---------------------------------+
| true                            |
+---------------------------------+
```

4. 获取 `test1` 的属性 `p1` 的月份。

```
nebula> CREATE TAG INDEX IF NOT EXISTS date1_index ON date1(p1);
nebula> REBUILD TAG INDEX date1_index;
nebula> MATCH (v:date1) RETURN v.date1.p1.month;
+------------------+
| v.date1.p1.month |
+------------------+
| 3                |
+------------------+
```

5. 查找 Tag `date1` 中属性 `p3` 小于 `2023-01-01T00:00:00.000000` 的值。

```
nebula> MATCH (v:date1)  \
WHERE v.date1.p3 < datetime("2023-01-01T00:00:00.000000") \
RETURN v.date1.p3;
+---------------------------+
| v.date1.p3                |
+---------------------------+
| 2017-03-04T14:30:40.003000 |
+---------------------------+
```

6. 创建 Tag，名称为 `school` ，包含 `TIMESTAMP` 类型。

```
nebula> CREATE TAG IF NOT EXISTS school(name string , found_time timestamp);
```


Copyright © Yueshu Graph Technology - All rights reserved.

7. 插入点，名称为 `DUT`，存储时间为 `"1988-03-01T08:00:00"`。

```
# 时间戳形式插入，1988-03-01T08:00:00 对应的时间戳为 573177600，转换为 UTC 时间为 573206400。
nebula> INSERT VERTEX school(name, found_time) VALUES "DUT":("DUT", 573206400);

# 日期和时间格式插入。
nebula> INSERT VERTEX school(name, found_time) VALUES "DUT":("DUT", timestamp("1988-03-01T08:00:00"));
```

8. 插入点，名称为 `dut`，用 `now()` 或 `timestamp()` 函数存储时间。

```
# 用 now() 函数存储时间
nebula> INSERT VERTEX school(name, found_time) VALUES "dut":("dut", now());

# 用 timestamp() 函数存储时间
nebula> INSERT VERTEX school(name, found_time) VALUES "dut":("dut", timestamp());
```

还可以使用 `WITH` 语句设置具体日期时间或进行计算，例如：

```
nebula> WITH time({hour: 12, minute: 31, second: 14, millisecond:111, microsecond: 222}) AS d RETURN d;
+-----------------+
| d               |
+-----------------+
| 12:31:14.111222 |
+-----------------+

nebula> WITH date({year: 1984, month: 10, day: 11}) AS x RETURN x + 1;
+------------+
| (x+1)      |
+------------+
| 1984-10-12 |
+------------+

nebula> WITH date('1984-10-11') as x, duration({years: 12, days: 14, hours: 99, minutes: 12}) as d \
        RETURN x + d AS sum, x - d AS diff;
+------------+------------+
| sum        | diff       |
+------------+------------+
| 1996-10-29 | 1972-09-23 |
+------------+------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.5 NULL

默认情况下，插入点或边时，属性值可以为 `NULL`，用户也可以设置属性值不允许为 `NULL`（`NOT NULL`），即插入点或边时必须设置该属性的值，除非创建属性时已经设置默认值。

NULL 的逻辑操作

`AND`、`OR`、`XOR` 和 `NOT` 的真值表如下。

a	b	a AND b	a OR b	a XOR b	NOT a
false	false	false	false	false	true
false	null	false	null	null	true
false	true	false	true	true	true
true	false	false	true	true	false
true	null	null	true	null	false
true	true	true	true	false	false
null	false	false	null	null	null
null	null	null	null	null	null
null	true	null	true	null	null

OpenCypher 兼容性

悦数图数据库中，NULL 的比较和操作与 openCypher 不同，后续也可能会有变化。

NULL 的比较

悦数图数据库中，NULL 的比较操作不兼容 openCypher。

NULL 的操作和返回

悦数图数据库中，对 NULL 的操作以及返回结果不兼容 openCypher。

示例

使用 NOT NULL

创建 Tag，名称为 `player`，指定属性 `name` 为 `NOT NULL`。

```
nebula> CREATE TAG IF NOT EXISTS player(name string NOT NULL, age int);
```

使用 `SHOW` 命令查看创建 Tag 语句，属性 `name` 为 `NOT NULL`，属性 `age` 为默认的 `NULL`。

```
nebula> SHOW CREATE TAG player;
+-----------+----------------------------------+
| Tag       | Create Tag                       |
+-----------+----------------------------------+
| "student" | "CREATE TAG `player` (           |
|           |  `name` string NOT NULL,         |
|           |  `age` int64 NULL                |
|           | ) ttl_duration = 0, ttl_col = """ |
+-----------+----------------------------------+
```

插入点 `Kobe`，属性 `age` 可以为 `NULL`。

```
nebula> INSERT VERTEX player(name, age) VALUES "Kobe":("Kobe",null);
```


Copyright © Yueshu Graph Technology - All rights reserved.

使用 **NOT NULL** 并设置默认值

创建 Tag，名称为 `player`，指定属性 `age` 为 `NOT NULL`，并设置默认值 `18`。

```
nebula> CREATE TAG IF NOT EXISTS player(name string, age int NOT NULL DEFAULT 18);
```

插入点 `Kobe`，只设置属性 `name`。

```
nebula> INSERT VERTEX player(name) VALUES "Kobe":("Kobe");
```

查询点 `Kobe`，属性 `age` 为默认值 `18`。

```
nebula> FETCH PROP ON player "Kobe" YIELD properties(vertex);
+-------------------------+
| properties(VERTEX)      |
+-------------------------+
| {age: 18, name: "Kobe"} |
+-------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.6 列表

列表（List）是复合数据类型，一个列表是一组元素的序列，可以通过元素在序列中的位置访问列表中的元素。

列表用左方括号（[）和右方括号（]）包裹多个元素，各个元素之间用英文逗号（,）隔开。元素前后的空格在列表中被忽略，因此可以使用换行符、制表符和空格调整格式。

OpenCypher 兼容性

复合数据类型（例如 List、Set、Map）不能存储为点或边的属性。

列表操作

对列表进行操作可以使用预设的列表函数，也可以使用下标表达式过滤列表内的元素。

下标表达式语法

```
[M]
[M..N]
[M..]
[..N]
```

nGQL 的下标支持从前往后查询，从 0 开始，0 表示第一个元素，1 表示第二个元素，以此类推；也支持从后往前查询，从-1 开始，-1 表示最后一个元素，-2 表示倒数第二个元素，以此类推。

- [M]：表示下标为 M 的元素。
- [M..N]：表示 `M ≤ 下标 < N` 的元素。`N` 为 0 时，返回为空。
- [M..]：表示 `M ≤ 下标` 的元素。
- [..N]：表示 `下标 < N` 的元素。`N` 为 0 时，返回为空。

Note

- 越界的下标返回为空，未越界的可以正常返回。
- `M ≥ N` 时，返回为空。
- 查询单个元素时，如果 `M` 为 null，返回报错 `BAD_TYPE`；范围查询时，`M` 或 `N` 为 null，返回为 `null`。

示例

```
# 返回列表 [1,2,3]
nebula> RETURN list[1, 2, 3] AS a;
+-----------+
| a         |
+-----------+
| [1, 2, 3] |
+-----------+

# 返回列表 [1,2,3,4,5] 中位置下标为 3 的元素。列表的位置下标是从 0 开始，因此返回的元素为 4。
nebula> RETURN range(1,5)[3];
+---------------+
| range(1,5)[3] |
+---------------+
| 4             |
+---------------+

# 返回列表 [1,2,3,4,5] 中位置下标为-2 的元素。列表的最后一个元素的位置下标是-1，因此-2 是指倒数第二个元素，即 4。
nebula> RETURN range(1,5)[-2];
+------------------+
| range(1,5)[-(2)] |
+------------------+
| 4                |
+------------------+

# 返回列表 [1,2,3,4,5] 中下标位置从 0 到 3（不包括 3）的元素。
nebula> RETURN range(1,5)[0..3];
+------------------+
| range(1,5)[0..3] |
+------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| [1, 2, 3]       |
+-----------------+

# 返回列表 [1,2,3,4,5] 中位置下标大于 2 的元素。
nebula> RETURN range(1,5)[3..] AS a;
+--------+
| a      |
+--------+
| [4, 5] |
+--------+

# 返回列表内下标小于 3 的元素。
nebula> WITH list[1, 2, 3, 4, 5] AS a \
        RETURN a[..3] AS r;
+-----------+
| r         |
+-----------+
| [1, 2, 3] |
+-----------+

# 筛选列表 [1,2,3,4,5] 中大于 2 的元素，将这些元素分别做运算并返回。
nebula> RETURN [n IN range(1,5) WHERE n > 2 | n + 10] AS a;
+--------------+
| a            |
+--------------+
| [13, 14, 15] |
+--------------+

# 返回列表内第一个至倒数第二个（包括）的元素。
nebula> YIELD list[1, 2, 3][0..-1] AS a;
+--------+
| a      |
+--------+
| [1, 2] |
+--------+

# 返回列表内倒数第三个至倒数第一个（不包括）的元素。
nebula> YIELD list[1, 2, 3, 4, 5][-3..-1] AS a;
+--------+
| a      |
+--------+
| [3, 4] |
+--------+

# 设置变量，返回列表内下标为 1、2 的元素。
nebula> $var = YIELD 1 AS f, 3 AS t; \
        YIELD list[1, 2, 3][$var.f..$var.t] AS a;
+--------+
| a      |
+--------+
| [2, 3] |
+--------+

# 越界的下标返回为空，未越界的可以正常返回。
nebula> RETURN list[1, 2, 3, 4, 5] [0..10] AS a;
+-----------------+
| a               |
+-----------------+
| [1, 2, 3, 4, 5] |
+-----------------+

nebula> RETURN list[1, 2, 3] [-5..5] AS a;
+-----------+
| a         |
+-----------+
| [1, 2, 3] |
+-----------+

# [0..0] 时返回为空。
nebula> RETURN list[1, 2, 3, 4, 5] [0..0] AS a;
+----+
| a  |
+----+
| [] |
+----+

# M ≥ N 时，返回为空。
nebula> RETURN list[1, 2, 3, 4, 5] [3..1] AS a;
+----+
| a  |
+----+
| [] |
+----+

# 范围查询时，下标有 null 时，返回为 null。
nebula> WITH list[1,2,3] AS a \
        RETURN a[0..null] as r;
+----------+
| r        |
+----------+
| __NULL__ |
+----------+

# 将列表 [1,2,3,4,5] 中的元素分别做运算，然后将列表去掉表头并返回。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> RETURN tail([n IN range(1, 5) | 2 * n - 10]) AS a;
+-----------------+
| a               |
+-----------------+
| [-6, -4, -2, 0] |
+-----------------+

# 将列表 [1,2,3] 中的元素判断为真，然后返回。
nebula> RETURN [n IN range(1, 3) WHERE true | n] AS r;
+-----------+
| r         |
+-----------+
| [1, 2, 3] |
+-----------+

# 返回列表 [1,2,3] 的长度。
nebula> RETURN size(list[1,2,3]);
+---------------+
| size([1,2,3]) |
+---------------+
| 3             |
+---------------+

# 将列表 [92,90] 中的元素做运算，然后在 where 子句中进行条件判断。
nebula> GO FROM "player100" OVER follow WHERE properties(edge).degree NOT IN [x IN [92, 90] | x + $$.player.age] \
        YIELD dst(edge) AS id, properties(edge).degree AS degree;
+-------------+--------+
| id          | degree |
+-------------+--------+
| "player101" | 95     |
| "player102" | 90     |
+-------------+--------+

# 将 MATCH 语句的查询结果作为列表中的元素进行运算并返回。
nebula> MATCH p = (n:player{name:"Tim Duncan"})-[:follow]->(m) \
        RETURN [n IN nodes(p) | n.player.age + 100] AS r;
+------------+
| r          |
+------------+
| [142, 136] |
| [142, 141] |
+------------+
```

OpenCypher 兼容性

- 在 openCypher 中，查询越界元素时返回 `null`，而在 nGQL 中，查询单个越界元素时返回 `OUT_OF_RANGE`。

```
nebula> RETURN range(0,5)[-12];
+-------------------+
| range(0,5)[-(12)] |
+-------------------+
| OUT_OF_RANGE      |
+-------------------+
```

- 复合数据类型（例如 set、map、list）不能存储为点或边的属性。
- 建议修改图建模方式：将复合数据类型建模为点的邻边，而不是该点的自身属性，每条邻边可以动态增删，并且可以设置邻边的 Rank 值来控制邻边的顺序。
- List 中不支持 pattern，例如 `[(src)-[]->(m) | m.name]`。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.7 集合

集合（Set）是复合数据类型，集合中是一组元素，与列表（List）不同的是，集合中的元素是无序的，且不允许重复。

集合用左花括号（{）和右花括号（}）包裹多个元素，各个元素之间用英文逗号（,）隔开。元素前后的空格在集合中被忽略，因此可以使用换行符、制表符和空格调整格式。

OpenCypher 兼容性

- 复合数据类型（例如 List、Set、Map）不能存储为点或边的属性。
- 在 OpenCypher 中，集合不是一个数据类型，而在 nGQL 中，用户可以使用集合。

示例

```
# 返回集合 {1,2,3}。
nebula> RETURN set{1, 2, 3} AS a;
+-----------+
| a         |
+-----------+
| {3, 2, 1} |
+-----------+

# 返回集合 {1,2,1}，因为集合不允许重复元素，会返回 {1,2}，且顺序是无序的。
nebula> RETURN set{1, 2, 1} AS a;
+--------+
| a      |
+--------+
| {2, 1} |
+--------+

# 判断集合中是否有指定元素 1。
nebula> RETURN 1 IN set{1, 2} AS a;
+------+
| a    |
+------+
| true |
+------+

# 计算集合中的元素数量。
nebula> YIELD size(set{1, 2, 1}) AS a;
+---+
| a |
+---+
| 2 |
+---+

# 返回目标点属性值组成的集合。
nebula> GO FROM "player100" OVER follow \
        YIELD set{properties($$).name,properties($$).age} as a;
+-----------------------+
| a                     |
+-----------------------+
| {36, "Tony Parker"}   |
| {41, "Manu Ginobili"} |
+-----------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.8 映射

映射（Map）是复合数据类型。一个映射是一组键值对（Key-Value）的无序集合。在映射中，Key 是字符串类型，Value 可以是任何数据类型。用户可以通过 `map['<key>']` 的方法获取映射中的元素。

映射用左花括号（{）和右花括号（}）包裹多个键值对，各个键值对之间用英文逗号（,）隔开。键值对前后的空格在映射中被忽略，因此可以使用换行符、制表符和空格调整格式。

OpenCypher 兼容性

- 复合数据类型（例如 List、Set、Map）不能存储为点或边的属性。
- 不支持映射投影（map projection）。

示例

```
# 返回简单的映射。
nebula> YIELD map{key1: 'Value1', Key2: 'Value2'} as a;
+----------------------------------+
| a                                |
+----------------------------------+
| {Key2: "Value2", key1: "Value1"} |
+----------------------------------+

# 返回列表类型的映射。
nebula> YIELD map{listKey: [{inner: 'Map1'}, {inner: 'Map2'}]} as a;
+---------------------------------------------+
| a                                           |
+---------------------------------------------+
| {listKey: [{inner: "Map1"}, {inner: "Map2"}]} |
+---------------------------------------------+

# 返回混合类型的映射。
nebula> RETURN map{a: LIST[1,2], b: SET{1,2,1}, c: "hee"} as a;
+---------------------------------+
| a                               |
+---------------------------------+
| {a: [1, 2], b: {2, 1}, c: "hee"} |
+---------------------------------+

# 返回映射中的指定元素。
nebula> RETURN map{a: LIST[1,2], b: SET{1,2,1}, c: "hee"}["b"] AS b;
+--------+
| b      |
+--------+
| {2, 1} |
+--------+

# 判断映射中是否有指定key，暂不支持判断value。
nebula> RETURN "a" IN MAP{a:1, b:2} AS a;
+------+
| a    |
+------+
| true |
+------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.9 类型转换

类型转换是指将表达式的类型转换为另一个类型。

悦数图数据库支持显式地转换类型。详情参见类型转换函数。

示例

```
nebula> UNWIND [true, false, 'true', 'false', NULL] AS b \
        RETURN toBoolean(b) AS b;
+----------+
| b        |
+----------+
| true     |
+----------+
| false    |
+----------+
| true     |
+----------+
| false    |
+----------+
| __NULL__ |
+----------+

nebula> RETURN toFloat(1), toFloat('1.3'), toFloat('1e3'), toFloat('not a number');
+------------+----------------+----------------+-------------------------+
| toFloat(1) | toFloat("1.3") | toFloat("1e3") | toFloat("not a number") |
+------------+----------------+----------------+-------------------------+
| 1.0        | 1.3            | 1000.0         | __NULL__                |
+------------+----------------+----------------+-------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.2.10 地理空间

地理空间（GEOGRAPHY）是由经纬度构成的表示地理空间信息的数据类型。悦数图数据库当前支持简单地理要素中的 Point、LineString 和 Polygon 三种地理形状。支持 SQL-MM 3 中的部分核心 geo 解析、构造、格式设置、转换、谓词和度量等函数。

GEOGRAPHY

GEOGRAPHY 的基本类型是点，由经纬度确定一个点，例如 `"POINT(3 8)"` 表示经度为 `3°`，纬度为 `8°`。多个点可以构成线段或多边形。

类型	示例	说明
Point	`"POINT(3 8)"`	点类型
LineString	`"LINESTRING(3 8, 4.7 73.23)"`	线段类型
Polygon	`"POLYGON((0 1, 1 2, 2 3, 0 1))"`	多边形类型

Note

请勿直接插入上述类型的 GEOGRAPHY 数据，例如 `INSERT VERTEX any_shape(geo) VALUES "1":("POINT(1 1)")`，需要使用 geo 函数指定数据类型后才能插入，例如 `INSERT VERTEX any_shape(geo) VALUES "1":(ST_GeogFromText("POINT(1 1)"));`。

示例

```
//创建 Tag，允许存储任意形状地理空间数据类型。
nebula> CREATE TAG IF NOT EXISTS any_shape(geo geography);

//创建 Tag，只允许存储点形状地理空间数据类型。
nebula> CREATE TAG IF NOT EXISTS only_point(geo geography(point));

//创建 Tag，只允许存储线段形状地理空间数据类型。
nebula> CREATE TAG IF NOT EXISTS only_linestring(geo geography(linestring));

//创建 Tag，只允许存储多边形形状地理空间数据类型。
nebula> CREATE TAG IF NOT EXISTS only_polygon(geo geography(polygon));

//创建 Edge type，允许存储任意形状地理空间数据类型。
nebula> CREATE EDGE IF NOT EXISTS any_shape_edge(geo geography);

//创建存储多边形地理空间的点。
nebula> INSERT VERTEX any_shape(geo) VALUES "103":(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"));

//创建存储多边形地理空间的边。
nebula> INSERT EDGE any_shape_edge(geo) VALUES "201"->"302":(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"));

//查询点 103 的属性 geo。
nebula> FETCH PROP ON any_shape "103" YIELD ST_ASText(any_shape.geo);
+--------------------------------+
| ST_ASText(any_shape.geo)       |
+--------------------------------+
| "POLYGON((0 1, 1 2, 2 3, 0 1))" |
+--------------------------------+

//查询边 201->302 的属性 geo。
nebula> FETCH PROP ON any_shape_edge "201"->"302" YIELD ST_ASText(any_shape_edge.geo);
+--------------------------------+
| ST_ASText(any_shape_edge.geo)  |
+--------------------------------+
| "POLYGON((0 1, 1 2, 2 3, 0 1))" |
+--------------------------------+

//为 geo 属性创建索引并使用 LOOKUP 查询。
nebula> CREATE TAG INDEX IF NOT EXISTS any_shape_geo_index ON any_shape(geo);
nebula> REBUILD TAG INDEX any_shape_geo_index;
nebula> LOOKUP ON any_shape YIELD ST_ASText(any_shape.geo);
+--------------------------------+
| ST_ASText(any_shape.geo)       |
+--------------------------------+
| "POLYGON((0 1, 1 2, 2 3, 0 1))" |
+--------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

为 geo 属性创建索引时，还可以指定 geo 索引的参数。说明如下。

参数	默认值	说明
`s2_max_level`	`30`	S2 cell 用于填充的最大等级。取值：`1` ~ `30`。设置为小于默认值时，意味着会使用较大的单元格进行填充。
`s2_max_cells`	`8`	S2 cell 用于填充的最大数量，可以限制填充时的工作量。取值：`1` ~ `30`。对于复杂形状的区域（例如细矩形），可以使用更大的值。

Note

指定如上两个参数对 Point 类型属性没有影响，Point 类型属性的 `s2_max_level` 强制为 `30`。

```
nebula> CREATE TAG INDEX IF NOT EXISTS any_shape_geo_index ON any_shape(geo) with (s2_max_level=30, s2_max_cells=8);
```

更多索引说明请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.3 变量和复合查询

5.3.1 复合查询（子句结构）

复合查询将来自不同请求的数据放在一起，然后进行过滤、分组或者排序等，最后返回结果。

悦数图数据库支持三种方式进行复合查询（或子查询）：

- （opencypher 兼容语句）连接各个子句，让它们在彼此之间提供中间结果集。
- （原生 nGQL）多个查询可以合并处理，以英文分号（;）分隔，返回最后一个查询的结果。
- （原生 nGQL）可以用管道符（|）将多个查询连接起来，上一个查询的结果可以作为下一个查询的输入。

OpenCypher 兼容性

在复合查询中，请不要混用 opencypher 兼容语句和原生 nGQL 语句，例如 `MATCH ... | GO ... | YIELD ...` 。

- 如果使用 openCypher 兼容语句（`MATCH` 、 `RETURN` 、 `WITH` 等），请不要使用管道符或分号组合子句。
- 如果使用原生 nGQL 语句（`FETCH` 、 `GO` 、 `LOOKUP` 等），必须使用管道符或分号组合子句。

复合查询不支持事务

例如一个查询由三个子查询 A、B、C 组成，A 是一个读操作，B 是一个计算操作，C 是一个写操作，如果在执行过程中，任何一个操作执行失败，则整个结果是未定义的：没有回滚，而且写入的内容取决于执行程序。

Note

openCypher 没有事务要求。

示例

- opencypher 兼容语句

```
# 子句连接多个查询。
nebula> MATCH p=(v:player{name:"Tim Duncan"})--() \
        WITH nodes(p) AS n \
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
        UNWIND n AS n1 \
        RETURN DISTINCT n1;
```

- 原生 nGQL（分号）

```
# 只返回边。
nebula> SHOW TAGS; SHOW EDGES;

# 插入多个点。
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42); \
        INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36); \
        INSERT VERTEX player(name, age) VALUES "player102":("LaMarcus Aldridge", 33);
```

- 原生 nGQL（管道符）

```
# 管道符连接多个查询。
nebula> GO FROM "player100" OVER follow YIELD dst(edge) AS id | \
        GO FROM $-.id OVER serve YIELD properties($$).name AS Team, \
        properties($^).name AS Player;
+-----------+-----------------+
| Team      | Player          |
+-----------+-----------------+
| "Spurs"   | "Tony Parker"   |
| "Hornets" | "Tony Parker"   |
| "Spurs"   | "Manu Ginobili" |
+-----------+-----------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.3.2 自定义变量

悦数图数据库允许将一条语句的结果作为自定义变量传递给另一条语句。

OpenCypher 兼容性

当引用一个变量的点、边或路径，需要先给它命名。例如：

```
nebula> MATCH (v:player{name:"Tim Duncan"}) RETURN v;
+--------------------------------------------------+
| v                                                |
+--------------------------------------------------+
| ("player100" :player{name: "Tim Duncan", age: 42}) |
+--------------------------------------------------+
```

示例中的 `v` 就是自定义变量。

Caution

在同一个 MATCH 语句的模式中，不能重复使用边变量。例如 `e` 不能重复被写在模式 `p=(v1)-[e*2..2]->(v2)-[e*2..2]->(v3)` 中。

原生 nGQL

nGQL 扩展的自定义变量可以表示为 `$var_name`， `var_name` 由字母、数字或下划线（_）构成，不允许使用其他字符。

自定义变量仅在当前执行（本复合查询中）有效，执行结束后变量也会释放，不能在其他客户端、执行、session 中使用之前的自定义变量。

用户可以在复合查询中使用自定义变量。复合查询的详细信息请参见复合查询。

Note

- 自定义变量区分大小写。
- 在包含扩展的自定义变量的复合语句中，用英文分号 `;` 结束定义变量的语句。详情参见 nGQL 风格指南。

示例

```
nebula> $var = GO FROM "player100" OVER follow YIELD dst(edge) AS id; \
        GO FROM $var.id OVER serve YIELD properties($$).name AS Team, \
        properties($^).name AS Player;
+-----------+----------------+
| Team      | Player         |
+-----------+----------------+
| "Spurs"   | "Tony Parker"  |
| "Hornets" | "Tony Parker"  |
| "Spurs"   | "Manu Ginobili" |
+-----------+----------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.3.3 引用属性

用户可以在 `WHERE` 和 `YIELD` 子句中引用点或边的属性。

> **Note**
>
> 本功能仅适用于原生 nGQL 的 GO 语句。

引用点的属性

起始点

```
$^.<tag_name>.<prop_name>
```

参数	说明
`$^`	起始点
`tag_name`	点的 Tag 名称
`prop_name`	Tag 内的属性名称

目的点

```
$$.<tag_name>.<prop_name>
```

参数	说明
`$$`	目的点
`tag_name`	点的 Tag 名称
`prop_name`	Tag 内的属性名称

引用边的属性

引用自定义的边属性

```
<edge_type>.<prop_name>
```

参数	说明
`edge_type`	Edge type
`prop_name`	Edge type 的属性名称

引用内置的边属性

除了自定义的边属性，每条边还有如下四种内置属性：

参数	说明
`_src`	边的起始点
`_dst`	边的目的点
`_type`	边的类型内部编码，正负号表示方向：正数为正向边，负数为逆向边
`_rank`	边的 rank 值

示例

Copyright © Yueshu Graph Technology - All rights reserved.

```
# 返回起始点的 Tag player 的 name 属性值和目的点的 Tag player 的 age 属性值。
nebula> GO FROM "player100" OVER follow YIELD $^.player.name AS startName, $$.player.age AS endAge;
+--------------+--------+
| startName    | endAge |
+--------------+--------+
| "Tim Duncan" | 36     |
| "Tim Duncan" | 41     |
+--------------+--------+

# 返回 Edge type follow 的 degree 属性值。
nebula> GO FROM "player100" OVER follow YIELD follow.degree;
+---------------+
| follow.degree |
+---------------+
| 95            |
+---------------+

# 返回 EdgeType 是 follow 的起始点 VID、目的点 VID、EdgeType 编码（正数为正向边，负数为逆向边），和边的 rank 值。
nebula> GO FROM "player100" OVER follow YIELD follow._src, follow._dst, follow._type, follow._rank;
+-------------+-------------+--------------+--------------+
| follow._src | follow._dst | follow._type | follow._rank |
+-------------+-------------+--------------+--------------+
| "player100" | "player101" | 17           | 0            |
| "player100" | "player125" | 17           | 0            |
+-------------+-------------+--------------+--------------+
```

版本兼容性

从悦数图数据库 2.6.0 起支持了新的 Schema 相关函数。以上示例在悦数图数据库 3.5.1 中的近似写法如下：

```
GO FROM "player100" OVER follow YIELD properties($^).name AS startName, properties($$).age AS endAge;
GO FROM "player100" OVER follow YIELD properties(edge).degree;
GO FROM "player100" OVER follow YIELD src(edge), dst(edge), type(edge), rank(edge);
```

悦数图数据库 3.5.1 兼容新旧语法。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4 运算符

5.4.1 比较符

悦数图数据库支持的比较符如下。

符号	说明
`==`	相等
`!=`, `<>`	不等于
`>`	大于
`>=`	大于等于
`<`	小于
`<=`	小于等于
`IS NULL`	为 NULL
`IS NOT NULL`	不为 NULL
`IS EMPTY`	不存在
`IS NOT EMPTY`	存在

比较操作的结果是 `true` 或者 `false` 。

Note

- 比较不同类型的值通常没有定义，结果可能是 `NULL` 或其它。
- `EMPTY` 当前仅用于判断，不支持函数或者运算操作，包括且不限于 `GROUP BY` 、 `count()` 、 `sum()` 、 `max()` 、 `hash()` 、 `collect()` 、 `+` 、 `*` 。

OpenCypher 兼容性

openCypher 中没有 `EMPTY` ，因此不支持在 MATCH 语句中使用 `EMPTY` 。

示例

`==`

字符串比较时，会区分大小写。不同类型的值不相等。

Note

nGQL 中的相等符号是 `==` ，openCypher 中的相等符号是 `=` 。

```
nebula> RETURN 'A' == 'a', toUpper('A') == toUpper('a'), toLower('A') == toLower('a');
+------------+-----------------------------+-----------------------------+
| ("A"=="a") | (toUpper("A")==toUpper("a")) | (toLower("A")==toLower("a")) |
+------------+-----------------------------+-----------------------------+
| false      | true                        | true                        |
+------------+-----------------------------+-----------------------------+

nebula> RETURN '2' == 2, toInteger('2') == 2;
+----------+---------------------+
| ("2"==2) | (toInteger("2")==2) |
+----------+---------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| false    | true                |
+----------+---------------------+
```

`>`

```
nebula> RETURN 3 > 2;
+-------+
| (3>2) |
+-------+
| true  |
+-------+

nebula> WITH 4 AS one, 3 AS two \
        RETURN one > two AS result;
+--------+
| result |
+--------+
| true   |
+--------+
```

`>=`

```
nebula> RETURN 2 >= "2", 2 >= 2;
+----------+--------+
| (2>="2") | (2>=2) |
+----------+--------+
| __NULL__ | true   |
+----------+--------+
```

`<`

```
nebula> YIELD 2.0 < 1.9;
+---------+
| (2<1.9) |
+---------+
| false   |
+---------+
```

`<=`

```
nebula> YIELD 0.11 <= 0.11;
+--------------+
| (0.11<=0.11) |
+--------------+
| true         |
+--------------+
```

`!=`

```
nebula> YIELD 1 != '1';
+----------+
| (1!="1") |
+----------+
| true     |
+----------+
```

`IS [NOT] NULL`

```
nebula> RETURN null IS NULL AS value1, null == null AS value2, null != null AS value3;
+--------+----------+----------+
| value1 | value2   | value3   |
+--------+----------+----------+
| true   | __NULL__ | __NULL__ |
+--------+----------+----------+

nebula> RETURN length(NULL), size(NULL), count(NULL), NULL IS NULL, NULL IS NOT NULL, sin(NULL), NULL + NULL, [1, NULL] IS NULL;
+--------------+------------+-------------+--------------+------------------+-----------+-------------+------------------+
| length(NULL) | size(NULL) | count(NULL) | NULL IS NULL | NULL IS NOT NULL | sin(NULL) | (NULL+NULL) | [1,NULL] IS NULL |
+--------------+------------+-------------+--------------+------------------+-----------+-------------+------------------+
| __NULL__     | __NULL__   | 0           | true         | false            | __NULL__  | __NULL__    | false            |
+--------------+------------+-------------+--------------+------------------+-----------+-------------+------------------+

nebula> WITH {name: null} AS `map` \
        RETURN `map`.name IS NOT NULL;
+----------------------+
| map.name IS NOT NULL |
+----------------------+
| false                |
+----------------------+

nebula> WITH {name: 'Mats', name2: 'Pontus'} AS map1, \
        {name: null} AS map2, {notName: 0, notName2: null } AS map3 \
        RETURN map1.name IS NULL, map2.name IS NOT NULL, map3.name IS NULL;
+--------------------+----------------------+-------------------+
| map1.name IS NULL | map2.name IS NOT NULL | map3.name IS NULL |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+------------------+----------------------+------------------+
| false            | false                | true             |
+------------------+----------------------+------------------+

nebula> MATCH (n:player) \
        RETURN n.player.age IS NULL, n.player.name IS NOT NULL, n.player.empty IS NULL;
+---------------------+--------------------------+-----------------------+
| n.player.age IS NULL | n.player.name IS NOT NULL | n.player.empty IS NULL |
+---------------------+--------------------------+-----------------------+
| false               | true                     | true                  |
| false               | true                     | true                  |
...
```

`IS [NOT] EMPTY`

```
nebula> RETURN null IS EMPTY;
+--------------+
| NULL IS EMPTY |
+--------------+
| false        |
+--------------+

nebula> RETURN "a" IS NOT EMPTY;
+-----------------+
| "a" IS NOT EMPTY |
+-----------------+
| true            |
+-----------------+

nebula> GO FROM "player100" OVER * WHERE properties($$).name IS NOT EMPTY YIELD dst(edge);
+------------+
| dst(EDGE)  |
+------------+
| "team204"  |
| "player101" |
| "player125" |
+------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.2 布尔符

悦数图数据库支持的布尔符如下。

符号	说明
AND	逻辑与
OR	逻辑或
NOT	逻辑非
XOR	逻辑异或

对于以上运算的优先级，请参见运算优先级。

对于带有 `NULL` 的逻辑运算，请参见 NULL。

Note

非 0 数字不能转换为布尔值。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.3 管道符

nGQL 支持使用管道符（|）将多个查询组合起来。

openCypher 兼容性

管道符仅适用于原生 nGQL。

语法

nGQL 和 SQL 之间的一个主要区别是子查询的组成方式。

- 在 SQL 中，子查询是嵌套在查询语句中的。
- 在 nGQL 中，子查询是通过类似 shell 中的管道符（`|`）实现的。

示例

```
nebula> GO FROM "player100" OVER follow \
        YIELD dst(edge) AS dstid, properties($$).name AS Name | \
        GO FROM $-.dstid OVER follow YIELD dst(edge);

+-------------+
| dst(EDGE)   |
+-------------+
| "player100" |
| "player102" |
| "player125" |
| "player100" |
+-------------+
```

必须在`YIELD`子句中为需要的返回结果设置别名，才能在管道符右侧使用引用符`$-`，例如示例中的`$-.dstid`。

性能提示

悦数图数据库中的管道对性能有影响，以`A | B`为例，体现在以下几个方面：

1. 管道是同步操作。也即需要管道之前的子句`A`执行完毕后，数据才能整体进入管道子句。
2. 管道本身是需要序列化和反序列化的，这个是单线程执行的。
3. 如果`A`发大量数据给`|`，整个查询请求的总体时延可能会非常大。此时可以尝试拆分这个语句：

a. 应用程序发送`A`，
b. 将收到的返回结果在应用程序拆分，
c. 并发发送给多个 graphd，
d. 每个 graphd 执行部分 B。

这样通常比单个 graphd 执行完整地`A | B`要快很多。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.4 引用符

nGQL 提供引用符来表示 `WHERE` 和 `YIELD` 子句中的属性，或者复合查询中管道符之前的语句输出结果。

openCypher 兼容性

引用符仅适用于原生 nGQL。

引用符列表

引用符	说明
`$^`	引用起始点。更多信息请参见引用属性。
`$$`	引用目的点。更多信息请参见引用属性。
`$-`	引用复合查询中管道符之前的语句输出结果。更多信息请参见管道符。

示例

```
# 返回起始点和目的点的年龄。
nebula> GO FROM "player100" OVER follow YIELD properties($^).age AS SrcAge, properties($$).age AS DestAge;
+--------+---------+
| SrcAge | DestAge |
+--------+---------+
| 42     | 36      |
| 42     | 41      |
+--------+---------+

# 返回 player100 追随的 player 的名称和团队。
nebula> GO FROM "player100" OVER follow \
        YIELD dst(edge) AS id | \
        GO FROM $-.id OVER serve \
        YIELD properties($^).name AS Player, properties($$).name AS Team;
+-----------------+-----------+
| Player          | Team      |
+-----------------+-----------+
| "Tony Parker"   | "Spurs"   |
| "Tony Parker"   | "Hornets" |
| "Manu Ginobili" | "Spurs"   |
+-----------------+-----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.5 集合运算符

合并多个请求时，可以使用集合运算符，包括 `UNION` 、 `UNION ALL` 、 `INTERSECT` 和 `MINUS` 。

所有集合运算符的优先级相同，如果一个 nGQL 语句中有多个集合运算符，悦数图数据库会从左到右进行计算，除非用括号指定顺序。

Caution

集合运算符前后的查询语句中定义的变量名及顺序必需保持一致，例如 `RETURN a,b,c UNION RETURN a,b,c` 中的 `a,b,c` 的名称及顺序需要保持一致。

UNION、UNION DISTINCT、UNION ALL

```
<left> UNION [DISTINCT | ALL] <right> [ UNION [DISTINCT | ALL] <right> ...]
```

- 运算符 `UNION DISTINCT` （或使用缩写 `UNION` ）返回两个集合 A 和 B 的并集，不包含重复的元素。
- 运算符 `UNION ALL` 返回两个集合 A 和 B 的并集，包含重复的元素。
- `left` 和 `right` 必须有相同数量的列和数据类型。如果需要转换数据类型，请参见类型转换。

示例

```
# 返回两个查询结果的并集，不包含重复的元素。
nebula> GO FROM "player102" OVER follow YIELD dst(edge) \
        UNION \
        GO FROM "player100" OVER follow YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player100" |
| "player101" |
| "player125" |
+-------------+

nebula> MATCH (v:player) \
        WITH v.player.name AS n \
        RETURN n ORDER BY n LIMIT 3 \
        UNION \
        UNWIND ["Tony Parker", "Ben Simmons"] AS n \
        RETURN n;
+---------------------+
| n                   |
+---------------------+
| "Amar'e Stoudemire" |
| "Aron Baynes"       |
| "Ben Simmons"       |
| "Tony Parker"       |
+---------------------+

# 返回两个查询结果的并集，包含重复的元素。
nebula> GO FROM "player102" OVER follow YIELD dst(edge) \
        UNION ALL \
        GO FROM "player100" OVER follow YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player100" |
| "player101" |
| "player101" |
| "player125" |
+-------------+

nebula> MATCH (v:player) \
        WITH v.player.name AS n \
        RETURN n ORDER BY n LIMIT 3 \
        UNION ALL \
        UNWIND ["Tony Parker", "Ben Simmons"] AS n \
        RETURN n;
+---------------------+
| n                   |
+---------------------+
| "Amar'e Stoudemire" |
| "Aron Baynes"       |
| "Ben Simmons"       |
| "Tony Parker"       |
| "Ben Simmons"       |
+---------------------+

# UNION 也可以和 YIELD 语句一起使用，去重时会检查每一行的所有列，每列都相同时才会去重。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> GO FROM "player102" OVER follow \
        YIELD dst(edge) AS id, properties(edge).degree AS Degree, properties($$).age AS Age \
        UNION /* DISTINCT */ \
        GO FROM "player100" OVER follow \
        YIELD dst(edge) AS id, properties(edge).degree AS Degree, properties($$).age AS Age;
+-------------+--------+-----+
| id          | Degree | Age |
+-------------+--------+-----+
| "player100" | 75     | 42  |
| "player101" | 75     | 36  |
| "player101" | 95     | 36  |
| "player125" | 95     | 41  |
+-------------+--------+-----+
```

INTERSECT

```
<left> INTERSECT <right>
```

- 运算符 `INTERSECT` 返回两个集合 A 和 B 的交集。
- `left` 和 `right` 必须有相同数量的列和数据类型。如果需要转换数据类型，请参见类型转换。

示例

```
# 返回两个查询结果的交集。
nebula> GO FROM "player102" OVER follow \
        YIELD dst(edge) AS id, properties(edge).degree AS Degree, properties($$).age AS Age \
        INTERSECT \
        GO FROM "player100" OVER follow \
        YIELD dst(edge) AS id, properties(edge).degree AS Degree, properties($$).age AS Age;
+----+--------+-----+
| id | Degree | Age |
+----+--------+-----+
+----+--------+-----+

nebula> MATCH (v:player)-[e:follow]->(v2) \
        WHERE id(v) == "player102" \
        RETURN id(v2) As id, e.degree As Degree, v2.player.age AS Age \
        INTERSECT \
        MATCH (v:player)-[e:follow]->(v2) \
        WHERE id(v) == "player100" \
        RETURN id(v2) As id, e.degree As Degree, v2.player.age AS Age;
+----+--------+-----+
| id | Degree | Age |
+----+--------+-----+
+----+--------+-----+

nebula> UNWIND [1,2] AS a RETURN a \
        INTERSECT \
        UNWIND [1,2,3,4] AS a \
        RETURN a;
+---+
| a |
+---+
| 1 |
| 2 |
+---+
```

MINUS

```
<left> MINUS <right>
```

运算符 `MINUS` 返回两个集合 A 和 B 的差异，即 `A-B` 。请注意 `left` 和 `right` 的顺序， `A-B` 表示在集合 A 中，但是不在集合 B 中的元素。

示例

```
# 返回在第一个查询结果中，但是不在第二个查询结果中的元素。
nebula> GO FROM "player100" OVER follow YIELD dst(edge) \
        MINUS \
        GO FROM "player102" OVER follow YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player125" |
+-------------+

nebula> GO FROM "player102" OVER follow YIELD dst(edge) AS id \
        MINUS \
        GO FROM "player100" OVER follow YIELD dst(edge) AS id;
+-------------+
| id          |
+-------------+
| "player100" |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+-------------+

nebula> MATCH (v:player)-[e:follow]->(v2) \
        WHERE id(v) =="player102" \
        RETURN id(v2) AS id\
        MINUS \
        MATCH (v:player)-[e:follow]->(v2) \
        WHERE id(v) =="player100" \
        RETURN id(v2) AS id;
+-------------+
| id          |
+-------------+
| "player100" |
+-------------+

nebula> UNWIND [1,2,3] AS a RETURN a \
        MINUS \
        WITH 4 AS a \
        RETURN a;
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

集合运算符和管道符的优先级

当查询包含集合运算符和管道符（|）时，管道符的优先级高。例如 `GO FROM 1 UNION GO FROM 2 | GO FROM 3` 相当于 `GO FROM 1 UNION (GO FROM 2 | GO FROM 3)`。

示例

```
nebula> GO FROM "player102" OVER follow \
        YIELD dst(edge) AS play_dst  \
        UNION \
        GO FROM "team200" OVER serve REVERSELY \
        YIELD src(edge) AS play_src \
        | GO FROM $-.play_src OVER follow YIELD dst(edge) AS play_dst;

+-------------+
| play_dst    |
+-------------+
| "player100" |
| "player101" |
| "player117" |
| "player105" |
+-------------+
```

```
nebula> GO FROM "player102" OVER follow YIELD follow._dst AS play_dst  \
UNION \
GO FROM "team200" OVER serve REVERSELY YIELD serve._dst AS play_dst \
| GO FROM $-.play_dst OVER follow YIELD follow._dst AS play_dst;
```

该查询会先执行红框内的语句，然后执行绿框的 `UNION` 操作。

圆括号可以修改执行的优先级，例如：

```
nebula> (GO FROM "player102" OVER follow \
        YIELD dst(edge) AS play_dst  \
        UNION \
        GO FROM "team200" OVER serve REVERSELY \
        YIELD src(edge) AS play_dst) \
        | GO FROM $-.play_dst OVER follow YIELD dst(edge) AS play_dst;
```

该查询中，圆括号包裹的部分先执行，即先执行 `UNION` 操作，再将结果结合管道符进行下一步操作。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.6 字符串运算符

悦数图数据库支持使用字符串运算符进行连接、搜索、匹配运算。支持的运算符如下。

名称	说明
+	连接字符串。
CONTAINS	在字符串中执行搜索。
(NOT) IN	字符串是否匹配某个值。
(NOT) STARTS WITH	在字符串的开头执行匹配。
(NOT) ENDS WITH	在字符串的结尾执行匹配。
正则表达式	通过正则表达式匹配字符串。

Note

所有搜索或匹配都区分大小写。

示例

`+`

```
nebula> RETURN 'a' + 'b';
+-----------+
| ("a"+"b") |
+-----------+
| "ab"      |
+-----------+
nebula> UNWIND 'a' AS a UNWIND 'b' AS b RETURN a + b;
+-------+
| (a+b) |
+-------+
| "ab"  |
+-------+
```

`CONTAINS`

`CONTAINS` 要求待运算的左右两边都是字符串类型。

```
nebula> MATCH (s:player)-[e:serve]->(t:team) WHERE id(s) == "player101" \
        AND t.team.name CONTAINS "ets" RETURN s.player.name, e.start_year, e.end_year, t.team.name;
+---------------+--------------+------------+-------------+
| s.player.name | e.start_year | e.end_year | t.team.name |
+---------------+--------------+------------+-------------+
| "Tony Parker" | 2018         | 2019       | "Hornets"   |
+---------------+--------------+------------+-------------+

nebula> GO FROM "player101" OVER serve WHERE (STRING)properties(edge).start_year CONTAINS "19" AND \
        properties($^).name CONTAINS "ny" \
        YIELD properties($^).name, properties(edge).start_year, properties(edge).end_year, properties($$).name;
+---------------------+----------------------------+--------------------------+---------------------+
| properties($^).name | properties(EDGE).start_year | properties(EDGE).end_year | properties($$).name |
+---------------------+----------------------------+--------------------------+---------------------+
| "Tony Parker"       | 1999                       | 2018                     | "Spurs"             |
+---------------------+----------------------------+--------------------------+---------------------+

nebula> GO FROM "player101" OVER serve WHERE !(properties($$).name CONTAINS "ets") \
        YIELD properties($^).name, properties(edge).start_year, properties(edge).end_year, properties($$).name;
+---------------------+----------------------------+--------------------------+---------------------+
| properties($^).name | properties(EDGE).start_year | properties(EDGE).end_year | properties($$).name |
+---------------------+----------------------------+--------------------------+---------------------+
| "Tony Parker"       | 1999                       | 2018                     | "Spurs"             |
+---------------------+----------------------------+--------------------------+---------------------+
```

`(NOT) IN`

```
nebula> RETURN  1 IN [1,2,3], "Yao" NOT IN ["Yi", "Tim", "Kobe"], NULL IN ["Yi", "Tim", "Kobe"];
+---------------+----------------------------------+-------------------------------+
| (1 IN [1,2,3]) | ("Yao" NOT IN ["Yi","Tim","Kobe"]) | (NULL IN ["Yi","Tim","Kobe"]) |
+---------------+----------------------------------+-------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| true           | true                              | __NULL__                      |
+----------------+-----------------------------------+-------------------------------+
```

(NOT) STARTS WITH

```
nebula> RETURN 'apple' STARTS WITH 'app', 'apple' STARTS WITH 'a', 'apple' STARTS WITH toUpper('a');
+----------------------------+--------------------------+-----------------------------------+
| ("apple" STARTS WITH "app") | ("apple" STARTS WITH "a") | ("apple" STARTS WITH toUpper("a")) |
+----------------------------+--------------------------+-----------------------------------+
| true                       | true                     | false                             |
+----------------------------+--------------------------+-----------------------------------+

nebula> RETURN 'apple' STARTS WITH 'b','apple' NOT STARTS WITH 'app';
+--------------------------+--------------------------------+
| ("apple" STARTS WITH "b") | ("apple" NOT STARTS WITH "app") |
+--------------------------+--------------------------------+
| false                    | false                          |
+--------------------------+--------------------------------+
```

(NOT) ENDS WITH

```
nebula> RETURN 'apple' ENDS WITH 'app', 'apple' ENDS WITH 'e', 'apple' ENDS WITH 'E', 'apple' ENDS WITH 'b';
+--------------------------+-------------------------+-------------------------+-------------------------+
| ("apple" ENDS WITH "app") | ("apple" ENDS WITH "e") | ("apple" ENDS WITH "E") | ("apple" ENDS WITH "b") |
+--------------------------+-------------------------+-------------------------+-------------------------+
| false                    | true                    | false                   | false                   |
+--------------------------+-------------------------+-------------------------+-------------------------+
```

正则表达式

> **Note**
>
> 当前仅 opencypher 兼容语句（`MATCH` 、 `WITH` 等）支持正则表达式，原生 nGQL 语句（`FETCH` 、 `GO` 、 `LOOKUP` 等）不支持正则表达式。

悦数图数据库支持使用正则表达式进行过滤，正则表达式的语法是继承自 `std::regex` ，用户可以使用语法 `=~ '<regexp>'` 进行正则表达式匹配。例如：

```
nebula> RETURN "384748.39" =~ "\\d+(\\.\\d{2})?";
+--------------------------------+
| ("384748.39"=~"\d+(\.\d{2})?") |
+--------------------------------+
| true                           |
+--------------------------------+

nebula> MATCH (v:player) WHERE v.player.name =~ 'Tony.*' RETURN v.player.name;
+---------------+
| v.player.name |
+---------------+
| "Tony Parker" |
+---------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.7 列表运算符

悦数图数据库支持使用列表（List）运算符进行运算。支持的运算符如下。

名称	说明
+	连接列表。
IN	元素是否存在于列表中。
[]	使用下标操作符访问列表中的元素。

示例

```
nebula> YIELD [1,2,3,4,5]+[6,7] AS myList;
+----------------------+
| myList               |
+----------------------+
| [1, 2, 3, 4, 5, 6, 7] |
+----------------------+

nebula> RETURN size([NULL, 1, 2]);
+------------------+
| size([NULL,1,2]) |
+------------------+
| 3                |
+------------------+

nebula> RETURN NULL IN [NULL, 1];
+--------------------+
| (NULL IN [NULL,1]) |
+--------------------+
| __NULL__           |
+--------------------+

nebula> WITH [2, 3, 4, 5] AS numberlist \
    UNWIND numberlist AS number \
    WITH number \
    WHERE number IN [2, 3, 8] \
    RETURN number;
+--------+
| number |
+--------+
| 2      |
| 3      |
+--------+

nebula> WITH ['Anne', 'John', 'Bill', 'Diane', 'Eve'] AS names RETURN names[1] AS result;
+--------+
| result |
+--------+
| "John" |
+--------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.8 算术运算符

悦数图数据库支持的算术运算符如下。

符号	说明
`+`	加法
`-`	减法
`*`	乘法
`/`	除法
`%`	取模
`-`	负数符号

示例

```
nebula> RETURN 1+2 AS result;
+--------+
| result |
+--------+
| 3      |
+--------+

nebula> RETURN -10+5 AS result;
+--------+
| result |
+--------+
| -5     |
+--------+

nebula> RETURN (3*8)%5 AS result;
+--------+
| result |
+--------+
| 4      |
+--------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.4.9 运算符优先级

nGQL 运算符的优先级从高到低排列如下（同一行的运算符优先级相同）：

- `-`（负数）
- `!` 、 `NOT`
- `*` 、 `/` 、 `%`
- `-` 、 `+`
- `==` 、 `>=` 、 `>` 、 `<=` 、 `<` 、 `<>` 、 `!=`
- `AND`
- `OR` 、 `XOR`
- `=`（赋值）

如果表达式中有相同优先级的运算符，运算是从左到右进行，只有赋值操作是例外（从右到左运算）。

运算符的优先级决定运算的顺序，要显式修改运算顺序，可以使用圆括号。

示例

```
nebula> RETURN 2+3*5;
+-----------+
| (2+(3*5)) |
+-----------+
| 17        |
+-----------+

nebula> RETURN (2+3)*5;
+-----------+
| ((2+3)*5) |
+-----------+
| 25        |
+-----------+
```

openCypher 兼容性

在 openCypher 中，比较操作可以任意连接，例如 `x < y <= z` 等价于 `x < y AND y <= z` 。

在 nGQL 中， `x < y <= z` 等价于 `(x < y) <= z` ， `(x < y)` 的结果是一个布尔值，再将布尔值和 `z` 比较，最终结果是 `NULL` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5 函数和表达式

5.5.1 内置数学函数

本文介绍悦数图数据库支持的数学函数。

abs()

abs() 返回指定数字的绝对值。

语法： `abs(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN abs(-10);
+------------+
| abs(-(10)) |
+------------+
| 10         |
+------------+

nebula> RETURN abs(5-6);
+------------+
| abs((5-6)) |
+------------+
| 1          |
+------------+
```

floor()

floor() 返回小于或等于指定数字的最大整数。

语法： `floor(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN floor(9.9);
+------------+
| floor(9.9) |
+------------+
| 9.0        |
+------------+
```

ceil()

ceil() 返回大于或等于指定数字的最小整数。

语法： `ceil(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN ceil(9.1);
+-----------+
| ceil(9.1) |
+-----------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 10.0     |
+----------+
```

round()

round() 返回指定数字四舍五入后的值。极端情况下请注意浮点数的精度问题。

语法： `round(<expression>, <digit>)`

- `expression`： 结果的数据类型为 double 的表达式。
- `digit`： 小数位数。小于 0 时 ， 在小数点左侧做四舍五入。数据类型为 int。
- 返回类型：double。

示例：

```
nebula> RETURN round(314.15926, 2);
+--------------------+
| round(314.15926,2) |
+--------------------+
| 314.16             |
+--------------------+

nebula> RETURN round(314.15926, -1);
+-----------------------+
| round(314.15926,-(1)) |
+-----------------------+
| 310.0                 |
+-----------------------+
```

sqrt()

sqrt() 返回指定数字的平方根。

语法： `sqrt(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN sqrt(9);
+---------+
| sqrt(9) |
+---------+
| 3.0     |
+---------+
```

cbrt()

cbrt() 返回指定数字的立方根。

语法： `cbrt(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN cbrt(8);
+---------+
| cbrt(8) |
+---------+
| 2.0     |
+---------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

hypot()

hypot() 返回直角三角形的斜边长。

语法： `hypot(<expression_x>,<expression_y>)`

- `expression_x` 、 `expression_y` ： 结果的数据类型为 double 的表达式。表示直角三角形的边长 `x` 和 `y` 。
- 返回类型：double。

示例：

```
nebula> RETURN hypot(3,2*2);
+---------------+
| hypot(3,(2*2)) |
+---------------+
| 5.0           |
+---------------+
```

pow()

pow() 返回指定数字的幂（x^y）。

语法： `pow(<expression_x>,<expression_y>,)`

- `expression_x` ： 结果的数据类型为 double 的表达式。表示底数 `x` 。
- `expression_y` ： 结果的数据类型为 double 的表达式。表示指数 `y` 。
- 返回类型：double。

示例：

```
nebula> RETURN pow(3,3);
+---------+
| pow(3,3) |
+---------+
| 27      |
+---------+
```

exp()

exp() 返回自然常数 e 的幂（e^x）。

语法： `exp(<expression>)`

- `expression` ： 结果的数据类型为 double 的表达式。表示指数 `x` 。
- 返回类型：double。

示例：

```
nebula> RETURN exp(2);
+------------------+
| exp(2)           |
+------------------+
| 7.38905609893065 |
+------------------+
```

exp2()

exp2() 返回2的幂（2^x ）。

语法： `exp2(<expression>)`

- `expression` ： 结果的数据类型为 double 的表达式。表示指数 `x` 。
- 返回类型：double。

Copyright © Yueshu Graph Technology - All rights reserved.

示例：

```
nebula> RETURN exp2(3);
+---------+
| exp2(3) |
+---------+
| 8.0     |
+---------+
```

log()

log() 返回以自然数 e 为底的对数（\(log_{e}{N}\)）。

语法：`log(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。表示真数 `N`。
- 返回类型：double。

示例：

```
nebula> RETURN log(8);
+--------------------+
| log(8)             |
+--------------------+
| 2.0794415416798357 |
+--------------------+
```

log2()

log2() 返回以 2 为底的对数（\(log_{2}{N}\)）。

语法：`log2(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。表示真数 `N`。
- 返回类型：double。

示例：

```
nebula> RETURN log2(8);
+---------+
| log2(8) |
+---------+
| 3.0     |
+---------+
```

log10()

log10() 返回以 10 为底的对数（\(log_{10}{N}\)）。

语法：`log10(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。表示真数 `N`。
- 返回类型：double。

示例：

```
nebula> RETURN log10(100);
+------------+
| log10(100) |
+------------+
| 2.0        |
+------------+
```

sin()

sin() 返回指定数字的正弦值。可以使用函数 `radians()` 将角度转化为弧度。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `sin(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN sin(3);
+--------------------+
| sin(3)             |
+--------------------+
| 0.1411200080598672 |
+--------------------+
```

asin()

asin() 返回指定数字的反正弦值。可以使用函数 `radians()` 将角度转化为弧度。

语法： `asin(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN asin(0.5);
+--------------------+
| asin(0.5)          |
+--------------------+
| 0.5235987755982989 |
+--------------------+
```

cos()

cos() 返回指定数字的余弦值。可以使用函数 `radians()` 将角度转化为弧度。

语法： `cos(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN cos(0.5);
+--------------------+
| cos(0.5)           |
+--------------------+
| 0.8775825618903728 |
+--------------------+
```

acos()

acos() 返回指定数字的反余弦值。可以使用函数 `radians()` 将角度转化为弧度。

语法： `acos(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN acos(0.5);
+--------------------+
| acos(0.5)          |
+--------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 1.0471975511965979 |
+--------------------+
```

tan()

tan() 返回指定数字的正切值。可以使用函数 `radians()` 将角度转化为弧度。

语法： `tan(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN tan(0.5);
+--------------------+
| tan(0.5)           |
+--------------------+
| 0.5463024898437905 |
+--------------------+
```

atan()

atan() 返回指定数字的反正切值。可以使用函数 `radians()` 将角度转化为弧度。

语法： `atan(<expression>)`

- `expression`：结果的数据类型为 double 的表达式。
- 返回类型：double。

示例：

```
nebula> RETURN atan(0.5);
+--------------------+
| atan(0.5)          |
+--------------------+
| 0.4636476090008061 |
+--------------------+
```

rand()

rand() 返回 [0,1) 内的随机浮点数。

语法： `rand()`

- 返回类型：double。

示例：

```
nebula> RETURN rand();
+--------------------+
| rand()             |
+--------------------+
| 0.6545837172298736 |
+--------------------+
```

rand32()

rand32() 返回指定范围（`[min, max)`）内的随机 32 位整数。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `rand32(<expression_min>,<expression_max>)`

- `expression_min`： 结果的数据类型为 int 的表达式。表示最小值 `min` 。
- `expression_max`： 结果的数据类型为 int 的表达式。表示最大值 `max` 。
- 返回类型：int。
- 用户可以只传入一个参数，该参数会判定为 `max` ，此时 `min` 默认为 `0` 。如果不传入参数，此时会从带符号的 32 位 int 范围内随机返回。

示例：

```
nebula> RETURN rand32(1,100);
+--------------+
| rand32(1,100) |
+--------------+
| 63           |
+--------------+
```

rand64()

rand64() 返回指定范围（`[min, max)`）内的随机 64 位整数。

语法： `rand64(<expression_min>,<expression_max>)`

- `expression_min`： 结果的数据类型为 int 的表达式。表示最小值 `min` 。
- `expression_max`： 结果的数据类型为 int 的表达式。表示最大值 `max` 。
- 返回类型：int。
- 用户可以只传入一个参数，该参数会判定为 `max` ，此时 `min` 默认为 `0` 。如果不传入参数，此时会从带符号的 64 位 int 范围内随机返回。

示例：

```
nebula> RETURN rand64(1,100);
+--------------+
| rand64(1,100) |
+--------------+
| 34           |
+--------------+
```

bit_and()

bit_and() 返回按位进行 AND 运算后的结果。

语法： `bit_and(<expression_1>,<expression_2>)`

- `expression_1` 、 `expression_2`： 结果的数据类型为 int 的表达式。
- 返回类型：int。

示例：

```
nebula> RETURN bit_and(5,6);
+-------------+
| bit_and(5,6) |
+-------------+
| 4           |
+-------------+
```

bit_or()

bit_or() 返回按位进行 OR 运算后的结果。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `bit_or(<expression_1>,<expression_2>)`

- `expression_1` 、 `expression_2` ： 结果的数据类型为 int 的表达式。
- 返回类型：int。

示例：

```
nebula> RETURN bit_or(5,6);
+------------+
| bit_or(5,6) |
+------------+
| 7          |
+------------+
```

bit_xor()

bit_xor() 返回按位进行 XOR 运算后的结果。

语法： `bit_xor(<expression_1>,<expression_2>)`

- `expression_1` 、 `expression_2` ： 结果的数据类型为 int 的表达式。
- 返回类型：int。

示例：

```
nebula> RETURN bit_xor(5,6);
+--------------+
| bit_xor(5,6) |
+--------------+
| 3            |
+--------------+
```

size()

size() 返回列表或映射中元素的数量，或者返回字符串的长度。

语法： `size({<expression>|<string>})`

- `expression` ： 列表或映射的表达式。
- `string` ： 指定的字符串。
- 返回类型：int。

示例：

```
nebula> RETURN size([1,2,3,4]);
+-----------------+
| size([1,2,3,4]) |
+-----------------+
| 4               |
+-----------------+
```

```
nebula> RETURN size("basketballplayer") as size;
+------+
| size |
+------+
| 16   |
+------+
```

range()

range() 返回指定范围（ `[start,end]` ）中指定步长的值组成的列表。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `range(<expression_start>,<expression_end>[,<expression_step>])`

- `expression_start`： 结果的数据类型为 int 的表达式。表示起始值 `start`。
- `expression_end`： 结果的数据类型为 int 的表达式。表示结束值 `end`。
- `expression_step`： 结果的数据类型为 int 的表达式。表示步长 `step`，默认值为 `1`。
- 返回类型：list。

示例：

```
nebula> RETURN range(1,3*3,2);
+------------------+
| range(1,(3*3),2) |
+------------------+
| [1, 3, 5, 7, 9]  |
+------------------+
```

sign()

sign() 返回指定数字的正负号。如果数字为 `0`，则返回 `0`；如果数字为负数，则返回 `-1`；如果数字为正数，则返回 `1`。

语法： `sign(<expression>)`

- `expression`： 结果的数据类型为 double 的表达式。
- 返回类型：int。

示例：

```
nebula> RETURN sign(10);
+----------+
| sign(10) |
+----------+
| 1        |
+----------+
```

e()

e() 返回自然对数的底 e（2.718281828459045）。

语法： `e()`

- 返回类型：double。

示例：

```
nebula> RETURN e();
+-------------------+
| e()               |
+-------------------+
| 2.718281828459045 |
+-------------------+
```

pi()

pi() 返回数学常数π（3.141592653589793）。

语法： `pi()`

- 返回类型：double。

示例：

```
nebula> RETURN pi();
+-------------------+
| pi()              |
+-------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 3.141592653589793 |
+-------------------+
```

radians()

radians() 返回指定角度的弧度。

语法： `radians(<angle>)`

- 返回类型：double。

示例：

```
nebula> RETURN radians(180);
+-------------------+
| radians(180)      |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.2 聚合函数

本文介绍悦数图数据库支持的聚合函数。

avg()

avg() 返回参数的平均值。

语法：`avg(<expression>)`

- 返回类型：double。

示例：

```
nebula> MATCH (v:player) RETURN avg(v.player.age);
+--------------------+
| avg(v.player.age)  |
+--------------------+
| 33.294117647058826 |
+--------------------+
```

count()

count() 返回参数的数量。

- （原生 nGQL）用户可以同时使用 `count()` 和 `GROUP BY` 对传参进行分组和计数，再使用 `YIELD` 返回结果。
- （openCypher 方式）用户可以使用 `count()` 对指定的值进行计数，再使用 `RETURN` 返回结果。不需要使用 `GROUP BY` 。

语法：`count({<expression> | *})`

- count(*) 返回总行数（包括 NULL）。
- 返回类型：int。

示例：

```
nebula> WITH [NULL, 1, 1, 2, 2] As a UNWIND a AS b \
        RETURN count(b), count(*), count(DISTINCT b);
+----------+----------+------------------+
| count(b) | count(*) | count(distinct b) |
+----------+----------+------------------+
| 4        | 5        | 2                |
+----------+----------+------------------+
```

```
# 返回 player101 follow 的人，以及 follow player101 的人，即双向查询。
# 使用`count()`和`GROUP BY`进行分组和计数。
nebula> GO FROM "player101" OVER follow BIDIRECT \
        YIELD properties($$).name AS Name \
        | GROUP BY $-.Name YIELD $-.Name, count(*);
+---------------------+----------+
| $-.Name             | count(*) |
+---------------------+----------+
| "LaMarcus Aldridge" | 2        |
| "Tim Duncan"        | 2        |
| "Marco Belinelli"   | 1        |
| "Manu Ginobili"     | 1        |
| "Boris Diaw"        | 1        |
| "Dejounte Murray"   | 1        |
+---------------------+----------+

# 使用`count()`进行计数。
nebula> MATCH (v1:player)-[:follow]-(v2:player) \
        WHERE id(v1)== "player101" \
        RETURN v2.player.name AS Name, count(*) as cnt ORDER BY cnt DESC;
+---------------------+-----+
| Name                | cnt |
+---------------------+-----+
| "LaMarcus Aldridge" | 2   |
| "Tim Duncan"        | 2   |
| "Boris Diaw"        | 1   |
| "Manu Ginobili"     | 1   |
| "Dejounte Murray"   | 1   |
| "Marco Belinelli"   | 1   |
+---------------------+-----+
```


Copyright © Yueshu Graph Technology - All rights reserved.

上述示例的返回结果有两列：

- `$-.Name`： 查询结果包含的姓名。
- `count(*)`： 姓名出现的次数。

因为测试数据集 `basketballplayer` 中没有重复的姓名， `count(*)` 列中数字 2 表示该行的人和 `player101` 是互相 `follow` 的关系。

```
# 方法一：统计数据库中的年龄分布情况。
nebula> LOOKUP ON player \
        YIELD player.age As playerage \
        | GROUP BY $-.playerage \
        YIELD $-.playerage as age, count(*) AS number \
        | ORDER BY $-.number DESC, $-.age DESC;
+-----+--------+
| age | number |
+-----+--------+
| 34  | 4      |
| 33  | 4      |
| 30  | 4      |
| 29  | 4      |
| 38  | 3      |
+-----+--------+
...

# 方法二：统计数据库中的年龄分布情况。
nebula> MATCH (n:player) \
        RETURN n.player.age as age, count(*) as number \
        ORDER BY number DESC, age DESC;
+-----+--------+
| age | number |
+-----+--------+
| 34  | 4      |
| 33  | 4      |
| 30  | 4      |
| 29  | 4      |
| 38  | 3      |
+-----+--------+
...
```

```
# 统计 Tim Duncan 关联的边数。
nebula> MATCH (v:player{name:"Tim Duncan"}) -[e]- (v2) \
        RETURN count(e);
+----------+
| count(e) |
+----------+
| 13       |
+----------+


# 多跳查询，统计 Tim Duncan 关联的边数，返回两列（不去重和去重）。
nebula> MATCH (n:player {name : "Tim Duncan"})-[]->(friend:player)-[]->(fof:player) \
        RETURN count(fof), count(DISTINCT fof);
+------------+---------------------+
| count(fof) | count(distinct fof) |
+------------+---------------------+
| 4          | 3                   |
+------------+---------------------+
```

max()

max() 返回参数的最大值。

语法： `max(<expression>)`

- 返回类型：与原参数相同。

示例：

```
nebula> MATCH (v:player) RETURN max(v.player.age);
+------------------+
| max(v.player.age) |
+------------------+
| 47               |
+------------------+
```

min()

min() 返回参数的最小值。

Copyright © Yueshu Graph Technology - All rights reserved.

语法：`min(<expression>)`

- 返回类型：与原参数相同。

示例：

```
nebula> MATCH (v:player) RETURN min(v.player.age);
+------------------+
| min(v.player.age) |
+------------------+
| 20               |
+------------------+
```

collect()

collect() 返回一个符合表达式返回结果的列表。该函数可以将多条记录或值合并进一个列表，实现数据聚合。

语法：`collect(<expression>)`

- 返回类型：list。

示例：

```
nebula> UNWIND [1, 2, 1] AS a \
        RETURN a;
+---+
| a |
+---+
| 1 |
| 2 |
| 1 |
+---+

nebula> UNWIND [1, 2, 1] AS a \
        RETURN collect(a);
+------------+
| collect(a) |
+------------+
| [1, 2, 1]  |
+------------+

nebula> UNWIND [1, 2, 1] AS a \
        RETURN a, collect(a), size(collect(a));
+---+------------+------------------+
| a | collect(a) | size(collect(a)) |
+---+------------+------------------+
| 2 | [2]        | 1                |
| 1 | [1, 1]     | 2                |
+---+------------+------------------+

# 降序排列，限制输出行数为 3，然后将结果输出到列表中。
nebula> UNWIND ["c", "b", "a", "d" ] AS p \
        WITH p AS q \
        ORDER BY q DESC LIMIT 3 \
        RETURN collect(q);
+----------------+
| collect(q)     |
+----------------+
| ["d", "c", "b"] |
+----------------+

nebula> WITH [1, 1, 2, 2] AS coll \
        UNWIND coll AS x \
        WITH DISTINCT x \
        RETURN collect(x) AS ss;
+--------+
| ss     |
+--------+
| [1, 2] |
+--------+

nebula> MATCH (n:player) \
        RETURN collect(n.player.age);
+-------------------------------------------------------------+
| collect(n.player.age)                                       |
+-------------------------------------------------------------+
| [32, 32, 34, 29, 41, 40, 33, 25, 40, 37, ...
...

# 基于年龄聚合姓名。
nebula> MATCH (n:player) \
        RETURN n.player.age AS age, collect(n.player.name);
+-----+-------------------------------------------------------------------+
| age | collect(n.player.name)                                            |
+-----+-------------------------------------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 24  | ["Giannis Antetokounmpo"]                                          |
| 20  | ["Luka Doncic"]                                                    |
| 25  | ["Joel Embiid", "Kyle Anderson"]                                   |
+-----+--------------------------------------------------------------------+
...

nebula> GO FROM "player100" OVER serve \
        YIELD properties($$).name AS name \
        | GROUP BY $-.name \
        YIELD collect($-.name) AS name;
+-----------+
| name      |
+-----------+
| ["Spurs"] |
+-----------+

nebula> LOOKUP ON player \
        YIELD player.age As playerage \
        | GROUP BY $-.playerage \
        YIELD collect($-.playerage) AS playerage;
+-----------------+
| playerage       |
+-----------------+
| [22]            |
| [47]            |
| [43]            |
| [25, 25]        |
+-----------------+
...
```

std()

std() 返回参数的总体标准差。

语法：`std(<expression>)`

- 返回类型：double。

示例：

```
nebula> MATCH (v:player) RETURN std(v.player.age);
+------------------+
| std(v.player.age) |
+------------------+
| 6.423895701687502 |
+------------------+
```

sum()

sum() 返回参数的和。

语法：`sum(<expression>)`

- 返回类型：与原参数相同。

示例：

```
nebula> MATCH (v:player) RETURN sum(v.player.age);
+------------------+
| sum(v.player.age) |
+------------------+
| 1698             |
+------------------+
```

聚合示例

```
nebula>  GO FROM "player100" OVER follow YIELD dst(edge) AS dst, properties($$).age AS age \
         | GROUP BY $-.dst \
         YIELD \
         $-.dst AS dst, \
         toInteger((sum($-.age)/count($-.age)))+avg(distinct $-.age+1)+1 AS statistics;
+------------+------------+
| dst        | statistics |
+------------+------------+
| "player125" | 84.0       |
| "player101" | 74.0       |
+------------+------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.3 内置字符串函数

本文介绍悦数图数据库支持的字符串函数。

注意事项

- 字符串的表示方式为用双引号或单引号包裹。
- 和 SQL 一样，nGQL 的字符索引（位置）从 `1` 开始。但是 C 语言的字符索引是从 `0` 开始的。

strcasecmp()

strcasecmp() 比较两个字符串（不区分大小写）。

语法： `strcasecmp(<string_a>,<string_b>)`

- `string_a` 、 `string_b` ： 待比较的字符串。
- 返回类型：int。
- 当 `string_a = string_b` 时，返回 `0` ，当 `string_a > string_b` 是，返回大于 `0` 的数，当 `string_a < string_b` 时，返回小于 `0` 的数。

示例：

```
nebula> RETURN strcasecmp("a","aa");
+---------------------+
| strcasecmp("a","aa") |
+---------------------+
| -97                 |
+---------------------+
```

lower() 和 toLower()

lower() 和 toLower() 都可以返回指定字符串的小写形式。

语法： `lower(<string>)` 、 `toLower(<string>)`

- `string` ： 指定的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN lower("Basketball_Player");
+----------------------------+
| lower("Basketball_Player") |
+----------------------------+
| "basketball_player"        |
+----------------------------+
```

upper() 和 toUpper()

upper() 和 toUpper() 都可以返回指定字符串的大写形式。

语法： `upper(<string>)` 、 `toUpper(<string>)`

- `string` ： 指定的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN upper("Basketball_Player");
+----------------------------+
| upper("Basketball_Player") |
+----------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "BASKETBALL_PLAYER"       |
+---------------------------+
```

length()

length() 返回：

- 指定字符串的长度，单位：字节。
- 路径的长度，单位：跳。

语法：`length({<string>|<path>})`

- `string`：指定的字符串。
- `path`：指定的路径，使用变量表示。
- 返回类型：int。

示例：

```
nebula> RETURN length("basketball");
+----------------------+
| length("basketball") |
+----------------------+
| 10                   |
+----------------------+
```

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-->(v2) return length(p);
+-----------+
| length(p) |
+-----------+
| 1         |
| 1         |
| 1         |
+-----------+
```

trim()

trim() 删除指定字符串头部和尾部的空格。

语法：`trim(<string>)`

- `string`：指定的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN trim(" basketball player ");
+----------------------------+
| trim(" basketball player ") |
+----------------------------+
| "basketball player"        |
+----------------------------+
```

ltrim()

ltrim() 删除字符串头部的空格。

语法：`ltrim(<string>)`

- `string`：指定的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN ltrim(" basketball player ");
+-----------------------------+
| ltrim(" basketball player ") |
+-----------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "basketball player "        |
+-----------------------------+
```

rtrim()

rtrim() 删除字符串尾部的空格。

语法： `rtrim(<string>)`

- `string`： 指定的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN rtrim(" basketball player ");
+-----------------------------+
| rtrim(" basketball player ") |
+-----------------------------+
| " basketball player"        |
+-----------------------------+
```

left()

left() 返回指定字符串头部若干个字符组成的子字符串。

语法： `left(<string>,<count>)`

- `string`： 指定的字符串。
- `count`： 指定从头部开始的字符数量。如果 `count` 超过字符串的长度，则返回字符串本身。
- 返回类型：string。

示例：

```
nebula> RETURN left("basketball_player",6);
+----------------------------+
| left("basketball_player",6) |
+----------------------------+
| "basket"                   |
+----------------------------+
```

right()

right() 返回指定字符串尾部若干个字符组成的子字符串。

语法： `right(<string>,<count>)`

- `string`： 指定的字符串。
- `count`： 指定从尾部开始的字符数量。如果 `count` 超过字符串的长度，则返回字符串本身。
- 返回类型：string。

示例：

```
nebula> RETURN right("basketball_player",6);
+----------------------------+
| right("basketball_player",6) |
+----------------------------+
| "player"                   |
+----------------------------+
```

lpad()

lpad() 在指定字符串的头部填充字符串至指定长度，并返回结果字符串。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `lpad(<string>,<count>,<letters>)`

- `string`： 指定的字符串。
- `count`： 指定从尾部开始将要返回的字符数量。如果 `count` 少于 `string` 字符串的长度，则只返回 `string` 字符串从前到后的 `count` 个字符。
- `letters`： 从头部填充的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN lpad("abcd",10,"b");
+---------------------+
| lpad("abcd",10,"b") |
+---------------------+
| "bbbbbbabcd"        |
+---------------------+

nebula> RETURN lpad("abcd",3,"b");
+--------------------+
| lpad("abcd",3,"b") |
+--------------------+
| "abc"              |
+--------------------+
```

rpad()

rpad() 在指定字符串的尾部填充字符串至指定长度，并返回结果字符串。

语法： `rpad(<string>,<count>,<letters>)`

- `string`： 指定的字符串。
- `count`： 指定从头部开始将要返回的字符数量。如果 `count` 少于 `string` 字符串的长度，则只返回 `string` 字符串从前到后的 `count` 个字符。
- `letters`： 从尾部填充的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN rpad("abcd",10,"b");
+---------------------+
| rpad("abcd",10,"b") |
+---------------------+
| "abcdbbbbbb"        |
+---------------------+

nebula> RETURN rpad("abcd",3,"b");
+--------------------+
| rpad("abcd",3,"b") |
+--------------------+
| "abc"              |
+--------------------+
```

substr() 和 substring()

substr() 和 substring() 从指定字符串的指定位置开始（不包括开始位置的字符），提取后面的若干个字符，组成新的字符串并返回。

语法： `substr(<string>,<pos>,<count>)` 、 `substring(<string>,<pos>,<count>)`

- `string`： 指定的字符串。
- `pos`： 指定开始的位置，即字符索引，数据类型为 int。
- `count`： 指定从开始位置往后提取的字符数量。
- 返回类型：string。

Copyright © Yueshu Graph Technology - All rights reserved.

`SUBSTR()` 和 `SUBSTRING()` 的返回说明

- 如果 `pos` 为 0，表示从指定字符串头部开始提取（包括第一个字符）。
- 如果 `pos` 大于最大字符索引，则返回空字符串。
- 如果 `pos` 是负数，则返回 `BAD_DATA` 。
- 如果省略 `count` ，则返回从 `pos` 位置开始到字符串末尾的子字符串。
- 如果 `count` 为 0，则返回空字符串。
- 使用 `NULL` 作为任何参数会出现错误。

> **openCypher 兼容性**
>
> 在 openCypher 中，如果字符串 `a` 为 `null` ，会返回 `null` 。

示例：

```
nebula> RETURN substr("abcdefg",2,4);
+-----------------------+
| substr("abcdefg",2,4) |
+-----------------------+
| "cdef"                |
+-----------------------+

nebula> RETURN substr("abcdefg",0,4);
+-----------------------+
| substr("abcdefg",0,4) |
+-----------------------+
| "abcd"                |
+-----------------------+

nebula> RETURN substr("abcdefg",2);
+---------------------+
| substr("abcdefg",2) |
+---------------------+
| "cdefg"             |
+---------------------+
```

reverse()

reverse() 逆序返回指定的字符串。

语法： `reverse(<string>)`

- `string` ： 指定的字符串。
- 返回类型：string。

示例：

```
nebula> RETURN reverse("abcdefg");
+--------------------+
| reverse("abcdefg") |
+--------------------+
| "gfedcba"          |
+--------------------+
```

replace()

replace() 将指定字符串中的子字符串 a 替换为字符串 b。

Copyright © Yueshu Graph Technology - All rights reserved.

语法：`replace(<string>,<substr_a>,<string_b>)`

- `string`：指定的字符串。
- `substr_a`：子字符串 a。
- `string_b`：字符串 b。
- 返回类型：string。

示例：

```
nebula> RETURN replace("abcdefg","cd","AAAAA");
+---------------------------------+
| replace("abcdefg","cd","AAAAA") |
+---------------------------------+
| "abAAAAAefg"                    |
+---------------------------------+
```

split()

split() 将子字符串 b 识别为分隔符，分隔指定字符串，并返回分隔后的字符串列表。

语法：`split(<string>,<substr>)`

- `string`：指定的字符串。
- `substr`：子字符串 b。
- 返回类型：list。

示例：

```
nebula> RETURN split("basketballplayer","a");
+------------------------------+
| split("basketballplayer","a") |
+------------------------------+
| ["b", "sketb", "llpl", "yer"] |
+------------------------------+
```

concat()

concat() 返回所有参数连接成的字符串。

语法：`concat(<string1>,<string2>,...)`

- 函数至少需要两个或以上字符串参数。如果字符串参数只有一个，则返回该字符串参数本身。
- 如果任何一个的字符串参数为 `NULL`，则 `concat()` 函数返回值为 `NULL`。
- 返回类型：string。

示例：

```
//连接 1, 2, 3
nebula> RETURN concat("1","2","3") AS r;
+-------+
| r     |
+-------+
| "123" |
+-------+

//字符串参数有 NULL
nebula> RETURN concat("1","2",NULL) AS r;
+----------+
| r        |
+----------+
| __NULL__ |
+----------+

nebula> GO FROM "player100" over follow \
        YIELD concat(src(edge), properties($^).age, properties($$).name, properties(edge).degree) AS A;
+------------------------------+
| A                            |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+-----------------------------+
| "player10042Tony Parker95"   |
| "player10042Manu Ginobili95" |
+-----------------------------+
```

concat_ws()

concat_ws() 返回用分隔符（separator）连接的所有字符串。

语法： `concat_ws(<separator>,<string1>,<string2>,... )`

- 函数至少需要两个或以上字符串参数。
- 如果分隔符为 `NULL` 时， `concat_ws()` 函数才返回 `NULL` 。
- 如果分隔符不为 `NULL` ，字符串参数只有一个，则返回该字符串参数本身。
- 字符串参数存在 `NULL` 值时，忽略 `NULL` 值，继续连接下一个参数。

示例：

```
//分隔符为 +，连接 a，b，c。
nebula> RETURN concat_ws("+","a","b","c") AS r;
+---------+
| r       |
+---------+
| "a+b+c" |
+---------+

//分隔符为 NULL。
neubla> RETURN concat_ws(NULL,"a","b","c") AS r;
+----------+
| r        |
+----------+
| __NULL__ |
+----------+

//分隔符为 +，字符串参数有 NULL。
nebula> RETURN concat_ws("+","a",NULL,"b","c") AS r;
+---------+
| r       |
+---------+
| "a+b+c" |
+---------+

//分隔符为+。字符串参数只有一个
nebula> RETURN concat_ws("+","a") AS r;
+-----+
| r   |
+-----+
| "a" |
+-----+

nebula> GO FROM "player100" over follow \
        YIELD concat_ws(" ",src(edge), properties($^).age, properties($$).name, properties(edge).degree) AS A;
+--------------------------------+
| A                              |
+--------------------------------+
| "player100 42 Tony Parker 95"   |
| "player100 42 Manu Ginobili 95" |
+--------------------------------+
```

extract()

extract() 从指定字符串中提取符合正则表达式的子字符串。

语法： `extract(<string>,"<regular_expression>")`

- `string` ： 指定的字符串。
- `regular_expression` ： 正则表达式。
- 返回类型：list。

示例：

```
nebula> MATCH (a:player)-[b:serve]-(c:team{name: "Lakers"}) \
        WHERE a.player.age > 45 \
        RETURN extract(a.player.name, "\\w+") AS result;
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+----------------------------+
| result                     |
+----------------------------+
| ["Shaquille", "O", "Neal"] |
+----------------------------+

nebula> MATCH (a:player)-[b:serve]-(c:team{name: "Lakers"}) \
        WHERE a.player.age > 45 \
        RETURN extract(a.player.name, "hello") AS result;
+--------+
| result |
+--------+
| []     |
+--------+
```

json_extract() 函数

json_extract() 将指定 JSON 字符串转换为 map 类型。

语法：`extract(<string>)`

- `string`：指定字符串，为 JSON 格式。
- 返回类型：map。

Caution

- 目前仅支持 Bool、Double、Int、String 和 NULL 类型数据。
- 仅支持深度为 1 的 Map 嵌套，如果嵌套深度为 2 及以上，嵌套项保留为空。

示例：

```
nebula> YIELD json_extract('{"a": 1, "b": {}, "c": {"d": true}}') AS result;
+----------------------------+
| result                     |
+----------------------------+
| {a: 1, b: {}, c: {d: true}} |
+----------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.4 内置日期时间函数

悦数图数据库支持以下内置日期时间函数。

函数	说明
int now()	根据当前系统返回当前时间戳。
timestamp timestamp()	根据当前系统返回当前时间戳。
date date()	根据当前系统返回当前日期（UTC 时间）。
time time()	根据当前系统返回当前时间（UTC 时间）。
datetime datetime()	根据当前系统返回当前日期和时间（UTC 时间）。
map duration()	持续时间。可以用于对指定时间进行计算。

详细信息参见日期和时间类型。

示例

```
nebula>  RETURN now(), timestamp(), date(), time(), datetime();
+------------+-------------+------------+-----------------+----------------------------+
| now()      | timestamp() | date()     | time()          | datetime()                 |
+------------+-------------+------------+-----------------+----------------------------+
| 1640057560 | 1640057560  | 2021-12-21 | 03:32:40.351000 | 2021-12-21T03:32:40.351000 |
+------------+-------------+------------+-----------------+----------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.5 Schema 相关函数

本文介绍悦数图数据库支持的 Schema 相关的函数。

Schema 相关的函数分为两类：

- 适用于原生 nGQL 语句
- 适用于 openCypher 兼容语句

原生 nGQL 语句适用

原生 nGQL 语句的 `YIELD` 和 `WHERE` 子句中可以使用如下介绍的函数。

> **Note**
>
> 由于 vertex、edge、vertices、edges、path 属于关键字，使用时需要用 `AS <alias>` 设置别名才能正常使用。例如 `GO FROM "player100" OVER follow YIELD edge AS e;`。

ID(VERTEX)

id(vertex) 返回点 ID。

语法：`id(vertex)`

- 返回类型：和点 ID 的类型保持一致。

示例：

```
nebula> LOOKUP ON player WHERE player.age  > 45 YIELD id(vertex);
+------------+
| id(VERTEX) |
+------------+
| "player144" |
| "player140" |
+------------+
```

PROPERTIES(VERTEX)

properties(vertex) 返回点的所有属性。

语法：`properties(vertex)`

- 返回类型：map。

示例：

```
nebula> LOOKUP ON player WHERE player.age  > 45 \
        YIELD properties(vertex);
+-------------------------------------+
| properties(VERTEX)                  |
+-------------------------------------+
| {age: 47, name: "Shaquille O'Neal"} |
| {age: 46, name: "Grant Hill"}       |
+-------------------------------------+
```

用户也可以使用属性引用符（`$^` 和 `$$`）替代函数 `properties()` 中的 `vertex` 参数来获取点的所有属性。

- `$^` 表示探索开始时的点数据。例如 `GO FROM "player100" OVER follow reversely YIELD properties($^)` 中，`$^` 指 `player100` 这个点。
- `$$` 表示探索结束的终点数据。

`properties($^)` 和 `properties($$)` 一般用于 `GO` 语句中。更多信息，请参见属性引用符。

Copyright © Yueshu Graph Technology - All rights reserved.

Caution

用户可以通过 `properties().<property_name>` 来获取点的指定属性。但是不建议使用这种方式获取指定属性，因为 `properties()` 函数返回所有属性，这样会降低查询性能。

PROPERTIES(EDGE)

properties(edge) 返回边的所有属性。

语法： `properties(edge)`

- 返回类型：map。

示例：

```
nebula> GO FROM "player100" OVER follow \
        YIELD properties(edge);
+------------------+
| properties(EDGE) |
+------------------+
| {degree: 95}     |
| {degree: 95}     |
+------------------+
```

Warning

用户可以通过 `properties(edge).<property_name>` 来获取边的指定属性。但是不建议使用这种方式获取指定属性，因为 `properties(edge)` 函数返回边的所有属性，这样会降低查询性能。

TYPE(EDGE)

type(edge) 返回边的 Edge type。

语法： `type(edge)`

- 返回类型：string。

示例：

```
nebula> GO FROM "player100" OVER follow \
        YIELD src(edge), dst(edge), type(edge), rank(edge);
+-------------+-------------+------------+------------+
| src(EDGE)   | dst(EDGE)   | type(EDGE) | rank(EDGE) |
+-------------+-------------+------------+------------+
| "player100" | "player101" | "follow"   | 0          |
| "player100" | "player125" | "follow"   | 0          |
+-------------+-------------+------------+------------+
```

SRC(EDGE)

src(edge) 返回边的起始点 ID。

语法： `src(edge)`

- 返回类型：和点 ID 的类型保持一致。

示例：

```
nebula> GO FROM "player100" OVER follow \
        YIELD src(edge), dst(edge);
+-------------+-------------+
| src(EDGE)   | dst(EDGE)   |
+-------------+-------------+
| "player100" | "player101" |
| "player100" | "player125" |
+-------------+-------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

Note

src(edge) 和 properties(`$^`) 查找起始点的语义不同。src(edge) 始终表示图数据库中边的起始点 ID，而 properties(`$^`) 表示探索开始时的点数据，例如示例中 `GO FROM "player100"` 中 `player100` 这个点的数据。

DST(EDGE)

dst(edge) 返回边的目的点 ID。

语法： `dst(edge)`

- 返回类型：和点 ID 的类型保持一致。

示例：

```
nebula> GO FROM "player100" OVER follow \
        YIELD src(edge), dst(edge);
+-------------+-------------+
| src(EDGE)   | dst(EDGE)   |
+-------------+-------------+
| "player100" | "player101" |
| "player100" | "player125" |
+-------------+-------------+
```

Note

dst(edge) 始终表示图数据库中边的目的点 ID。

RANK(EDGE)

rank(edge) 返回边的 rank。

语法： `rank(edge)`

- 返回类型：int。

示例：

```
nebula> GO FROM "player100" OVER follow \
        YIELD src(edge), dst(edge), rank(edge);
+-------------+-------------+------------+
| src(EDGE)   | dst(EDGE)   | rank(EDGE) |
+-------------+-------------+------------+
| "player100" | "player101" | 0          |
| "player100" | "player125" | 0          |
+-------------+-------------+------------+
```

VERTEX

vertex 返回点的信息。包括点 ID、Tag、属性和值。需要用 `AS <alias>` 设置别名。

语法： `vertex`

示例：

```
nebula> LOOKUP ON player WHERE player.age  > 45 YIELD vertex AS v;
+-------------------------------------------------------+
| v                                                     |
+-------------------------------------------------------+
| ("player144" :player{age: 47, name: "Shaquille O'Neal"}) |
| ("player140" :player{age: 46, name: "Grant Hill"})       |
+-------------------------------------------------------+
```

EDGE

edge 返回边的信息。包括 Edge type、起始点 ID、目的点 ID、rank、属性和值。需要用 `AS <alias>` 设置别名。

语法： `edge`

Copyright © Yueshu Graph Technology - All rights reserved.

示例：

```
nebula> GO FROM "player100" OVER follow YIELD edge AS e;
+--------------------------------------------------+
| e                                                |
+--------------------------------------------------+
| [:follow "player100"->"player101" @0 {degree: 95}] |
| [:follow "player100"->"player125" @0 {degree: 95}] |
+--------------------------------------------------+
```

VERTICES

vertices 返回子图中的点的信息。详情参见 GET SUBGRAPH。

EDGES

edges 返回子图中的边的信息。详情参见 GET SUBGRAPH。

PATH

path 返回路径信息。详情参见 FIND PATH。

openCypher 兼容语句适用

openCypher 兼容语句的 `RETURN` 和 `WHERE` 子句中可以使用如下介绍的函数。

ID()

id() 返回点 ID。

语法： `id(<vertex>)`

- 返回类型：和点 ID 的类型保持一致。

示例：

```
nebula> MATCH (v:player) RETURN id(v);
+-------------+
| id(v)       |
+-------------+
| "player129" |
| "player115" |
| "player106" |
| "player102" |
...
```

TAGS() 和LABELS()

tags() 和labels() 返回点的 Tag。

语法： `tags(<vertex>)` 、 `labels(<vertex>)`

- 返回类型：list。

示例：

```
nebula> MATCH (v) WHERE id(v) == "player100" \
        RETURN tags(v);
+------------+
| tags(v)    |
+------------+
| ["player"] |
+------------+
```

PROPERTIES()

properties() 返回点或边的所有属性。

语法： `properties(<vertex_or_edge>)`

- 返回类型：map。

示例：

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> MATCH (v:player)-[e:follow]-() RETURN properties(v),properties(e);
+----------------------------------------+---------------+
| properties(v)                          | properties(e) |
+----------------------------------------+---------------+
| {age: 31, name: "Stephen Curry"}       | {degree: 90}  |
| {age: 47, name: "Shaquille O'Neal"}    | {degree: 100} |
| {age: 34, name: "LeBron James"}        | {degree: 13}  |
...
```

TYPE()

type() 返回边的 Edge type。

语法：`type(<edge>)`

- 返回类型：string。

示例：

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->() \
        RETURN type(e);
+----------+
| type(e)  |
+----------+
| "serve"  |
| "follow" |
| "follow" |
+----------+
```

TYPEID()

typeid() 返回边的 Edge type 的内部 ID 值，可以用正负来判断方向。

语法：`typeid(<edge>)`

- 返回类型：int。

示例：

```
nebula> MATCH (v:player)-[e:follow]-(v2) RETURN e,typeid(e), \
        CASE WHEN typeid(e) > 0 \
        THEN "正向" ELSE "反向" END AS direction \
        LIMIT 5;
+-----------------------------------------------+-----------+-----------+
| e                                             | typeid(e) | direction |
+-----------------------------------------------+-----------+-----------+
| [:follow "player127"->"player114" @0 {degree: 90}] | 5         | "正向"    |
| [:follow "player127"->"player148" @0 {degree: 70}] | 5         | "正向"    |
| [:follow "player148"->"player127" @0 {degree: 80}] | -5        | "反向"    |
| [:follow "player147"->"player136" @0 {degree: 90}] | 5         | "正向"    |
| [:follow "player136"->"player147" @0 {degree: 90}] | -5        | "反向"    |
+-----------------------------------------------+-----------+-----------+
```

SRC()

src() 返回边的起始点 ID。

语法：`src(<edge>)`

- 返回类型：和点 ID 的类型保持一致。

示例：

```
nebula> MATCH ()-[e]->(v:player{name:"Tim Duncan"}) \
        RETURN src(e);
+-------------+
| src(e)      |
+-------------+
| "player125" |
| "player113" |
| "player102" |
...
```

DST()

dst() 返回边的目的点 ID。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `dst(<edge>)`

- 返回类型：和点 ID 的类型保持一致。

示例：

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->() \
        RETURN dst(e);
+-------------+
| dst(e)      |
+-------------+
| "team204"   |
| "player101" |
| "player125" |
+-------------+
```

STARTNODE()

startNode() 获取一条路径并返回它的起始点信息，包括点 ID、Tag、属性和值。

语法： `startNode(<path>)`

示例：

```
nebula> MATCH p = (a :player {name : "Tim Duncan"})-[r:serve]-(t) \
        RETURN startNode(p);
+-------------------------------------------------+
| startNode(p)                                    |
+-------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) |
+-------------------------------------------------+
```

ENDNODE()

endNode() 获取一条路径并返回它的目的点信息，包括点 ID、Tag、属性和值。

语法： `endNode(<path>)`

示例：

```
nebula> MATCH p = (a :player {name : "Tim Duncan"})-[r:serve]-(t) \
        RETURN endNode(p);
+----------------------------------+
| endNode(p)                       |
+----------------------------------+
| ("team204" :team{name: "Spurs"}) |
+----------------------------------+
```

RANK()

rank() 返回边的 rank。

语法： `rank(<edge>)`

- 返回类型：int。

示例：

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->() \
        RETURN rank(e);
+---------+
| rank(e) |
+---------+
| 0       |
| 0       |
| 0       |
+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.6 列表函数

本文介绍悦数图数据库支持的列表（List）函数。部分列表函数在原生 nGQL 语句和 openCypher 兼容语句中的语法不同。

注意事项

- 和 SQL 一样，nGQL 的字符索引（位置）从 `1` 开始。但是 C 语言的字符索引是从 `0` 开始的。

通用

RANGE()

range() 返回指定整数范围 `[start,end]` 内固定步长的列表。

语法： `range(start, end [, step])`

- `step`： 可选参数。步长。默认为 `1`。
- 返回类型：list。

示例：

```
nebula> RETURN range(1,9,2);
+----------------+
| range(1,9,2)   |
+----------------+
| [1, 3, 5, 7, 9] |
+----------------+
```

REVERSE()

reverse() 返回将原列表逆序排列的新列表。

语法： `reverse(<list>)`

- 返回类型：list。

示例：

```
nebula> WITH [NULL, 4923, 'abc', 521, 487] AS ids \
        RETURN reverse(ids);
+----------------------------------+
| reverse(ids)                     |
+----------------------------------+
| [487, 521, "abc", 4923, __NULL__] |
+----------------------------------+
```

TAIL()

tail() 返回不包含原列表第一个元素的新列表。

语法： `tail(<list>)`

- 返回类型：list。

示例：

```
nebula> WITH [NULL, 4923, 'abc', 521, 487] AS ids \
        RETURN tail(ids);
+------------------------+
| tail(ids)              |
+------------------------+
| [4923, "abc", 521, 487] |
+------------------------+
```

HEAD()

head() 返回列表的第一个元素。

Copyright © Yueshu Graph Technology - All rights reserved.

语法： `head(<list>)`

- 返回类型：与原列表内的元素类型相同。

示例：

```
nebula> WITH [NULL, 4923, 'abc', 521, 487] AS ids \
        RETURN head(ids);
+-----------+
| head(ids) |
+-----------+
| __NULL__  |
+-----------+
```

LAST()

last() 返回列表的最后一个元素。

语法： `last(<list>)`

- 返回类型：与原列表内的元素类型相同。

示例：

```
nebula> WITH [NULL, 4923, 'abc', 521, 487] AS ids \
        RETURN last(ids);
+-----------+
| last(ids) |
+-----------+
| 487       |
+-----------+
```

REDUCE()

reduce() 将表达式逐个应用于列表中的元素，然后和累加器中的当前结果累加，最后返回完整结果。该函数将遍历给定列表中的每个元素 e，在 e 上运行表达式并和累加器的当前结果累加，将新的结果存储在累加器中。这个函数类似于函数式语言（如 Lisp 和 Scala）中的 fold 或 reduce 方法。

> **openCypher 兼容性**
>
> 在 openCypher 中，`reduce()` 函数没有定义。nGQL 使用了 Cypher 方式实现 `reduce()` 函数。

语法： `reduce(<accumulator> = <initial>, <variable> IN <list> | <expression>)`

- `accumulator`： 在遍历列表时保存累加结果。
- `initial`： 为 `accumulator` 提供初始值的表达式或值。
- `variable`： 为列表引入一个变量，决定使用列表中的哪个元素。
- `list`： 列表或列表表达式。
- `expression`： 该表达式将对列表中的每个元素运行一次，并将结果累加至 `accumulator` 。
- 返回类型：取决于提供的参数，以及表达式的语义。

示例：

```
nebula> RETURN reduce(totalNum = -4 * 5, n IN [1, 2] | totalNum + n * 2) AS r;
+-----+
| r   |
+-----+
| -14 |
+-----+

nebula> MATCH p = (n:player{name:"LeBron James"})<-[:follow]-(m) \
        RETURN  nodes(p)[0].player.age AS src1,  nodes(p)[1].player.age AS dst2,  \
        reduce(totalAge = 100, n IN nodes(p) | totalAge + n.player.age) AS sum;
+------+------+-----+
| src1 | dst2 | sum |
+------+------+-----+
| 34   | 31   | 165 |
| 34   | 29   | 163 |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 34   | 33   | 167 |
| 34   | 26   | 160 |
| 34   | 34   | 168 |
| 34   | 37   | 171 |
+------+------+-----+

nebula> LOOKUP ON player WHERE player.name == "Tony Parker" YIELD id(vertex) AS VertexID \
        |  GO FROM $-.VertexID over follow \
        WHERE properties(edge).degree != reduce(totalNum = 5, n IN range(1, 3) | properties($$).age + totalNum + n) \
        YIELD properties($$).name AS id, properties($$).age AS age, properties(edge).degree AS degree;
+--------------------+-----+--------+
| id                 | age | degree |
+--------------------+-----+--------+
| "Tim Duncan"       | 42  | 95     |
| "LaMarcus Aldridge" | 33  | 90     |
| "Manu Ginobili"    | 41  | 95     |
+--------------------+-----+--------+
```

原生 **nGQL** 语句适用

KEYS()

keys() 返回一个列表，包含字符串形式的点、边的所有属性。

语法： `keys({vertex | edge})`

- 返回类型：list。

示例：

```
nebula> LOOKUP ON player \
        WHERE player.age  > 45 \
        YIELD keys(vertex);
+----------------+
| keys(VERTEX)   |
+----------------+
| ["age", "name"] |
| ["age", "name"] |
+----------------+
```

LABELS()

labels() 返回点的 Tag 列表。

语法： `labels(verte)`

- 返回类型：list。

示例：

```
nebula> FETCH PROP ON * "player101", "player102", "team204" \
        YIELD labels(vertex);
+---------------+
| labels(VERTEX) |
+---------------+
| ["player"]    |
| ["player"]    |
| ["team"]      |
+---------------+
```

openCypher 兼容语句适用

KEYS()

keys() 返回一个列表，包含字符串形式的点、边或映射的所有属性。

语法： `keys(<vertex_or_edge>)`

- 返回类型：list。

示例：

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->() \
        RETURN keys(e);
+---------------------------+
| keys(e)                   |
+---------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| ["end_year", "start_year"] |
| ["degree"]                 |
| ["degree"]                 |
+----------------------------+
```

LABELS()

labels() 返回点的 Tag 列表。

语法： `labels(<vertex>)`

- 返回类型：list。

示例：

```
nebula> MATCH (v)-[e:serve]->() \
        WHERE id(v)=="player100" \
        RETURN labels(v);
+------------+
| labels(v)  |
+------------+
| ["player"] |
+------------+
```

NODES()

nodes() 返回路径中所有点的列表。包括点 ID、Tag、属性和值。

语法： `nodes(<path>)`

- 返回类型：list。

示例：

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-->(v2) \
        RETURN nodes(p);
+------------------------------------------------------------------------------------------------------------------------+
| nodes(p)                                                                                                               |
+------------------------------------------------------------------------------------------------------------------------+
| [("player100" :player{age: 42, name: "Tim Duncan"}), ("team204" :team{name: "Spurs"})]                                 |
| [("player100" :player{age: 42, name: "Tim Duncan"}), ("player101" :player{age: 36, name: "Tony Parker"})]              |
| [("player100" :player{age: 42, name: "Tim Duncan"}), ("player125" :player{age: 41, name: "Manu Ginobili"})]            |
+------------------------------------------------------------------------------------------------------------------------+
```

RELATIONSHIPS()

relationships() 返回路径中所有关系的列表。

语法： `relationships(<path>)`

- 返回类型：list。

示例：

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-->(v2) \
        RETURN relationships(p);
+----------------------------------------------------------------------+
| relationships(p)                                                     |
+----------------------------------------------------------------------+
| [[:serve "player100"->"team204" @0 {end_year: 2016, start_year: 1997}]] |
| [[:follow "player100"->"player101" @0 {degree: 95}]]                 |
| [[:follow "player100"->"player125" @0 {degree: 95}]]                 |
+----------------------------------------------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.7 类型转换函数

本文介绍悦数图数据库支持的类型转换函数。

toBoolean()

toBoolean() 将字符串转换为布尔。

语法：`toBoolean(<value>)`

- 返回类型：bool。

示例：

```
nebula> UNWIND [true, false, 'true', 'false', NULL] AS b \
        RETURN toBoolean(b) AS b;
+----------+
| b        |
+----------+
| true     |
| false    |
| true     |
| false    |
| __NULL__ |
+----------+
```

toFloat()

toFloat() 将整数或字符串转换为浮点数。

语法：`toFloat(<value>)`

- 返回类型：float。

示例：

```
nebula> RETURN toFloat(1), toFloat('1.3'), toFloat('1e3'), toFloat('not a number');
+------------+----------------+----------------+------------------------+
| toFloat(1) | toFloat("1.3") | toFloat("1e3") | toFloat("not a number") |
+------------+----------------+----------------+------------------------+
| 1.0        | 1.3            | 1000.0         | __NULL__               |
+------------+----------------+----------------+------------------------+
```

toString()

toString() 将任意非复合数据类型数据转换为字符串类型。

语法：`toString(<value>)`

- 返回类型：string。

示例：

```
nebula> RETURN toString(9669) AS int2str, toString(null) AS null2str;
+---------+----------+
| int2str | null2str |
+---------+----------+
| "9669"  | __NULL__ |
+---------+----------+
```

toInteger()

toInteger() 将浮点或字符串转换为整数。

语法：`toInteger(<value>)`

- 返回类型：int。

Copyright © Yueshu Graph Technology - All rights reserved.

示例：

```
nebula> RETURN toInteger(1), toInteger('1'), toInteger('1e3'), toInteger('not a number');
+--------------+----------------+------------------+---------------------------+
| toInteger(1) | toInteger("1") | toInteger("1e3") | toInteger("not a number") |
+--------------+----------------+------------------+---------------------------+
| 1            | 1              | 1000             | __NULL__                  |
+--------------+----------------+------------------+---------------------------+
```

toSet()

toSet() 将列表或集合转换为集合。

语法： `toSet(<value>)`

- 返回类型：set。

示例：

```
nebula> RETURN toSet(list[1,2,3,1,2]) AS list2set;
+-----------+
| list2set  |
+-----------+
| {3, 1, 2} |
+-----------+
```

hash()

hash() 返回参数的哈希值。其参数可以是数字、字符串、列表、布尔值、NULL 等类型的值，或者计算结果为这些类型的表达式。

`hash()` 函数采用 MurmurHash2 算法，种子（seed）为 `0xc70f6907UL`。用户可以在 `MurmurHash2.h` 中查看其源代码。

在 Java 中的调用方式如下：

```
MurmurHash2.hash64("to_be_hashed".getBytes(),"to_be_hashed".getBytes().length, 0xc70f6907)
```

语法： `hash(<string>)`

- 返回类型：int。

示例：

```
nebula> RETURN hash("abcde");
+--------------------+
| hash("abcde")      |
+--------------------+
| 811036730794841393 |
+--------------------+

nebula> YIELD hash([1,2,3]);
+----------------+
| hash([1,2,3])  |
+----------------+
| 11093822460243 |
+----------------+

nebula> YIELD hash(NULL);
+------------+
| hash(NULL) |
+------------+
| -1         |
+------------+

nebula> YIELD hash(toLower("HELLO NEBULA"));
+------------------------------+
| hash(toLower("HELLO NEBULA")) |
+------------------------------+
| -8481157362655072082         |
+------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.8 条件表达式函数

本文介绍悦数图数据库支持的条件表达式函数。

CASE

`CASE` 表达式使用条件来过滤传参。和 openCypher 一样，nGQL 提供两种形式的 `CASE` 表达式：简单形式和通用形式。

`CASE` 表达式会遍历所有条件，并在满足第一个条件时停止读取后续条件，然后返回结果。如果不满足任何条件，将通过 `ELSE` 子句返回结果。如果没有 `ELSE` 子句且不满足任何条件，则返回 `NULL` 。

简单形式

- 语法

```
CASE <comparer>
WHEN <value> THEN <result>
[WHEN ...]
[ELSE <default>]
END
```

Caution

`CASE` 表达式一定要用 `END` 结尾。

参数	说明
`comparer`	用于与 `value` 进行比较的值或者有效表达式。
`value`	和 `comparer` 进行比较，如果匹配，则满足此条件。
`result`	如果 `value` 匹配 `comparer` ，则返回该 `result` 。
`default`	如果没有条件匹配，则返回该 `default` 。

- 示例

```
nebula> RETURN \
        CASE 2+3 \
        WHEN 4 THEN 0 \
        WHEN 5 THEN 1 \
        ELSE -1 \
        END \
        AS result;
+--------+
| result |
+--------+
| 1      |
+--------+
```

```
nebula> GO FROM "player100" OVER follow \
        YIELD properties($$).name AS Name, \
        CASE properties($$).age > 35 \
        WHEN true THEN "Yes" \
        WHEN false THEN "No" \
        ELSE "Nah" \
        END \
        AS Age_above_35;
+----------------+--------------+
| Name           | Age_above_35 |
+----------------+--------------+
| "Tony Parker"  | "Yes"        |
| "Manu Ginobili" | "Yes"       |
+----------------+--------------+
```

通用形式

- 语法

Copyright © Yueshu Graph Technology - All rights reserved.

```
CASE
WHEN <condition> THEN <result>
[WHEN ...]
[ELSE <default>]
END
```

参数	说明
`condition`	如果条件 `condition` 为 true，表示满足此条件。
`result`	`condition` 为 true，则返回此 `result` 。
`default`	如果没有条件匹配，则返回该 `default` 。

- 示例

```
nebula> YIELD \
        CASE WHEN 4 > 5 THEN 0 \
        WHEN 3+4==7 THEN 1 \
        ELSE 2 \
        END \
        AS result;
+--------+
| result |
+--------+
| 1      |
+--------+
```

```
nebula> MATCH (v:player) WHERE v.player.age > 30 \
        RETURN v.player.name AS Name,  \
        CASE \
        WHEN v.player.name STARTS WITH "T" THEN "Yes" \
        ELSE "No" \
        END \
        AS Starts_with_T;
+--------------------+--------------+
| Name               | Starts_with_T |
+--------------------+--------------+
| "Tim Duncan"       | "Yes"        |
| "LaMarcus Aldridge" | "No"         |
| "Tony Parker"      | "Yes"        |
+--------------------+--------------+
```

简单形式和通用形式的区别

为了避免误用简单形式和通用形式，用户需要了解它们的差异。请参见如下示例：

```
nebula> GO FROM "player100" OVER follow \
        YIELD properties($$).name AS Name, properties($$).age AS Age, \
        CASE properties($$).age \
        WHEN properties($$).age > 35 THEN "Yes" \
        ELSE "No" \
        END \
        AS Age_above_35;
+----------------+-----+-------------+
| Name           | Age | Age_above_35 |
+----------------+-----+-------------+
| "Tony Parker"  | 36  | "No"        |
| "Manu Ginobili" | 41  | "No"        |
+----------------+-----+-------------+
```

示例本意为当玩家年龄大于 35 时输出 `Yes` 。但是查看输出结果，年龄为 36 时输出的却是 `No` 。

这是因为查询使用了简单形式的 `CASE` 表达式，比较对象是 `$$.player.age` 和 `$$.player.age > 35` 。当年龄为 36 时：

- `$$.player.age` 的值为 `36` ，数据类型为 int。
- `$$.player.age > 35` 的值为 `true` ，数据类型为 boolean。

这两种数据类型无法匹配，不满足条件，因此返回 `No` 。

coalesce()

coalesce() 返回所有表达式中第一个非空元素。

Copyright © Yueshu Graph Technology - All rights reserved.

语法：`coalesce(<expression_1>[,<expression_2>...])`

- 返回类型：与原元素类型相同。

示例：

```
nebula> RETURN coalesce(null,[1,2,3]) as result;
+-----------+
| result    |
+-----------+
| [1, 2, 3] |
+-----------+

nebula> RETURN coalesce(null) as result;
+----------+
| result   |
+----------+
| __NULL__ |
+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.9 谓词函数

谓词函数只返回 `true` 或 `false`，通常用于 `WHERE` 子句中。

悦数图数据库支持以下谓词函数。

函数	说明
exists()	如果指定的属性在点、边或映射中存在，则返回 `true`，否则返回 `false`。
any()	如果指定的谓词适用于列表中的至少一个元素，则返回 `true`，否则返回 `false`。
all()	如果指定的谓词适用于列表中的每个元素，则返回 `true`，否则返回 `false`。
none()	如果指定的谓词不适用于列表中的任何一个元素，则返回 `true`，否则返回 `false`。
single()	如果指定的谓词适用于列表中的唯一一个元素，则返回 `true`，否则返回 `false`。

Note

如果列表为空，或者列表中的所有元素都为空，则返回 NULL。

mpatibility

在 openCypher 中只定义了函数 `exists()`，其他几个函数依赖于具体实现。

语法

```
<predicate>(<variable> IN <list> WHERE <condition>)
```

示例

```
nebula> RETURN any(n IN [1, 2, 3, 4, 5, NULL] \
        WHERE n > 2) AS r;
+------+
| r    |
+------+
| true |
+------+

nebula> RETURN single(n IN range(1, 5) \
        WHERE n == 3) AS r;
+------+
| r    |
+------+
| true |
+------+

nebula> RETURN none(n IN range(1, 3) \
        WHERE n == 0) AS r;
+------+
| r    |
+------+
| true |
+------+

nebula> WITH [1, 2, 3, 4, 5, NULL] AS a \
        RETURN any(n IN a WHERE n > 2);
+-------------------------+
| any(n IN a WHERE (n>2)) |
+-------------------------+
| true                    |
+-------------------------+

nebula> MATCH p = (n:player{name:"LeBron James"})<-[:follow]-(m) \
        RETURN nodes(p)[0].player.name AS n1, nodes(p)[1].player.name AS n2, \
        all(n IN nodes(p) WHERE n.player.name NOT STARTS WITH "D") AS b;
+----------------+-------------------+-------+
| n1             | n2                | b     |
+----------------+-------------------+-------+
| "LeBron James" | "Danny Green"     | false |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "LeBron James" | "Dejounte Murray" | false |
| "LeBron James" | "Chris Paul"      | true  |
| "LeBron James" | "Kyrie Irving"    | true  |
| "LeBron James" | "Carmelo Anthony" | true  |
| "LeBron James" | "Dwyane Wade"     | false |
+---------------+------------------+-------+

nebula> MATCH p = (n:player{name:"LeBron James"})-[:follow]->(m) \
        RETURN single(n IN nodes(p) WHERE n.player.age > 40) AS b;
+------+
| b    |
+------+
| true |
+------+

nebula> MATCH (n:player) \
        RETURN exists(n.player.id), n IS NOT NULL;
+--------------------+--------------+
| exists(n.player.id) | n IS NOT NULL |
+--------------------+--------------+
| false              | true         |
...

nebula> MATCH (n:player) \
        WHERE exists(n['name']) \
        RETURN n;
+------------------------------------------------------------+
| n                                                          |
+------------------------------------------------------------+
| ("player105" :player{age: 31, name: "Danny Green"})        |
| ("player109" :player{age: 34, name: "Tiago Splitter"})     |
| ("player111" :player{age: 38, name: "David West"})         |
...
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.5.10 geo 函数

geo 函数用于生成地理空间（GEOGRAPHY）数据类型的值或对其执行操作。

关于地理空间数据类型说明请参见地理空间。

函数说明

函数	返回类型	说明
ST_Point(longitude, latitude)	`GEOGRAPHY`	创建包含一个点的地理空间。
ST_GeogFromText(wkt_string)	`GEOGRAPHY`	返回与传入的 WKT 字符串形式相对应的 GEOGRAPHY。
ST_ASText(geography)	`STRING`	返回传入的 GEOGRAPHY 的 WKT 字符串形式。
ST_Centroid(geography)	`GEOGRAPHY`	以单点 GEOGRAPHY 的形式返回传入的 GEOGRAPHY 的形心。
ST_ISValid(geography)	`BOOL`	返回传入的 GEOGRAPHY 是否有效。
ST_Intersects(geography_1, geography_2)	`BOOL`	返回传入的两个 GEOGRAPHY 是否有交集。
ST_Covers(geography_1, geography_2)	`BOOL`	返回 geography_1 是否完全包含 geography_2。如果 geography_2 中没有位于 geography_1 外部的点，返回 True。
ST_CoveredBy(geography_1, geography_2)	`BOOL`	返回 geography_2 是否完全包含 geography_1。如果 geography_1 中没有位于 geography_2 外部的点，返回 True。
ST_DWithin(geography_1, geography_2, distance)	`BOOL`	如果 geography_1 中至少有一个点与 geography_2 中的一个点的距离小于或等于 distance 参数（以米为单位）指定的距离，则返回 True。
ST_Distance(geography_1, geography_2)	`FLOAT`	返回两个非空 GEOGRAPHY 之间的最短距离（以米为单位）。
S2_CellIdFromPoint(point_geography)	`INT`	返回覆盖点 GEOGRAPHY 的 S2 单元ID。
S2_CoveringCellIds(geography)	`ARRAY<INT64>`	返回覆盖传入的 GEOGRAPHY 的 S2 单元 ID 的数组。

示例

```
nebula> RETURN ST_ASText(ST_Point(1,1));
+-------------------------+
| ST_ASText(ST_Point(1,1)) |
+-------------------------+
| "POINT(1 1)"            |
+-------------------------+

nebula> RETURN ST_ASText(ST_GeogFromText("POINT(3 8)"));
+----------------------------------------+
| ST_ASText(ST_GeogFromText("POINT(3 8)")) |
+----------------------------------------+
| "POINT(3 8)"                           |
+----------------------------------------+

nebula> RETURN ST_ASTEXT(ST_Centroid(ST_GeogFromText("LineString(0 1,1 0)")));
+----------------------------------------------------------------+
| ST_ASTEXT(ST_Centroid(ST_GeogFromText("LineString(0 1,1 0)"))) |
+----------------------------------------------------------------+
| "POINT(0.5000380800773782 0.5000190382261059)"                 |
+----------------------------------------------------------------+

nebula> RETURN ST_ISValid(ST_GeogFromText("POINT(3 8)"));
+----------------------------------------+
| ST_ISValid(ST_GeogFromText("POINT(3 8)")) |
+----------------------------------------+
| true                                   |
+----------------------------------------+

nebula> RETURN ST_Intersects(ST_GeogFromText("LineString(0 1,1 0)"),ST_GeogFromText("LineString(0 0,1 1)"));
+------------------------------------------------------------------------------------------------+
| ST_Intersects(ST_GeogFromText("LineString(0 1,1 0)"),ST_GeogFromText("LineString(0 0,1 1)")) |
+------------------------------------------------------------------------------------------------+
| true                                                                                           |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+-----------------------------------------------------------------------------------------+

nebula> RETURN ST_Covers(ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))"),ST_Point(1,2));
+-----------------------------------------------------------------------------+
| ST_Covers(ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))"),ST_Point(1,2)) |
+-----------------------------------------------------------------------------+
| true                                                                        |
+-----------------------------------------------------------------------------+

nebula> RETURN ST_CoveredBy(ST_Point(1,2),ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))"));
+-----------------------------------------------------------------------------+
| ST_CoveredBy(ST_Point(1,2),ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))")) |
+-----------------------------------------------------------------------------+
| true                                                                        |
+-----------------------------------------------------------------------------+

nebula> RETURN ST_dwithin(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)"),20000000000.0);
+-----------------------------------------------------------------------------+
| ST_dwithin(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)"),20000000000) |
+-----------------------------------------------------------------------------+
| true                                                                        |
+-----------------------------------------------------------------------------+

nebula> RETURN ST_Distance(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)"));
+-----------------------------------------------------------------------+
| ST_Distance(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)")) |
+-----------------------------------------------------------------------+
| 1.5685230187677438e+06                                                |
+-----------------------------------------------------------------------+

nebula> RETURN S2_CellIdFromPoint(ST_GeogFromText("Point(1 1)"));
+-----------------------------------------------+
| S2_CellIdFromPoint(ST_GeogFromText("Point(1 1)")) |
+-----------------------------------------------+
| 1153277837650709461                           |
+-----------------------------------------------+

nebula> RETURN S2_CoveringCellIds(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"));
+-----------------------------------------------------------------------------------------
```

5.6 通用查询语句

5.6.1 MATCH

`MATCH` 语句提供基于模式（Pattern）匹配的搜索功能，其通过定义一个或多个模式，允许在 NebulaGraph 中查找与模式匹配的数据。在检索到匹配的数据后，用户可以使用 RETURN 子句将其作为结果返回。

在本文中，我们将使用名为 basketballplayer 的测试数据集来演示 MATCH 语句的使用。

语法

`MATCH` 语句的语法相较于其他查询语句（如 `GO` 和 `LOOKUP`）更具灵活性。在进行查询时，`MATCH` 语句使用的路径类型是 `trail`，这意味着点可以重复出现，但边不能重复。

`MATCH` 语法的基本结构如下：

```
MATCH <pattern> [<clause_1>]  RETURN <output>  [<clause_2>];
```

- `pattern`： `MATCH` 语句支持匹配一个或多个模式，多个模式之间用英文逗号（`,`）分隔。例如 `(a)-[]->(b),(c)-[]->(d)`。Pattern 的详细说明请参见模式。
- `clause_1`： 支持 `WHERE`、`WITH`、`UNWIND`、`OPTIONAL MATCH` 子句，也可以使用 `MATCH` 作为子句。
- `output`： 定义需要返回输出结果的列表名称。可以使用 `AS` 设置列表的别名。
- `clause_2`： 支持 `ORDER BY`、`LIMIT` 子句。

> 版本兼容性

- 从 3.5.0 版本开始，`MATCH` 语句支持全表扫描，即在不使用任何索引或者过滤条件的情况下可遍历图中点或边。在此之前的版本中，MATCH 语句在某些情况下需要索引才能执行查询或者需要使用 `LIMIT` 限制输出结果数量。
- 从 3.0.0 版本开始，为了区别不同 Tag 的属性，返回属性时必须额外指定 Tag 名称。即从 `RETURN <变量名>.<属性名>` 改为 `RETURN <变量名>.<Tag名>.<属性名>`。

使用说明

- 尽量避免执行全表扫描，因为这可能导致查询性能下降；并且如果在进行全表扫描时内存不足，可能会导致查询失败，系统会提示报错。建议使用具有过滤条件或指定 Tag、边类型的查询，例如 `MATCH (v:player) RETURN v.player.name AS Name` 语句中的 `v:player` 和 `v.player.name`。
- 可为 Tag、Edge type 或 Tag、Edge type 的某个属性创建索引，以提高查询性能。例如，用户可以为 `player` Tag 创建索引，或者为 `player` Tag 的 `name` 属性创建索引。有关索引的使用及注意事项，请参见使用索引必读。
- 目前 MATCH 语句无法查询到悬挂边。

使用模式

匹配点

用户可以在一对括号中使用自定义变量来表示模式中的点。例如 `(v)`。

```
nebula> MATCH (v) \
        RETURN v \
        LIMIT 3;
+-------------------------------------------------------+
| v                                                     |
+-------------------------------------------------------+
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
| ("player106" :player{age: 25, name: "Kyle Anderson"})     |
| ("player115" :player{age: 40, name: "Kobe Bryant"})       |
+-------------------------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

匹配 **TAG**

> 历史版本兼容性

在悦数图数据库 3.0.0 之前，匹配 Tag 的前提是 Tag 本身有索引或者 Tag 的某个属性有索引，否则，用户无法基于该 Tag 执行 `MATCH` 语句。从悦数图数据库 3.0.0 开始，匹配 Tag 可以不创建索引，但需要使用 `LIMIT` 限制输出结果数量。从悦数图数据库 3.5.0 开始，`MATCH` 语句支持全表扫描，无需为 Tag 或 Tag 的某个属性创建索引，或者使用 `LIMIT` 限制输出结果数量，即可执行 `MATCH` 语句。

用户可以在点的右侧用 `:<tag_name>` 表示模式中的 Tag。

```
nebula> MATCH (v:player) \
        RETURN v;
+-------------------------------------------------------+
| v                                                     |
+-------------------------------------------------------+
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
| ("player106" :player{age: 25, name: "Kyle Anderson"})     |
| ("player115" :player{age: 40, name: "Kobe Bryant"})       |
...
```

需要匹配拥有多个 Tag 的点，可以用英文冒号（`:`）。

```
nebula> CREATE TAG actor (name string, age int);
nebula> INSERT VERTEX actor(name, age) VALUES "player100":("Tim Duncan", 42);
nebula> MATCH (v:player:actor) \
        RETURN v;
+-----------------------------------------------------------------------------------------+
| v                                                                                       |
+-----------------------------------------------------------------------------------------+
| ("player100" :actor{age: 42, name: "Tim Duncan"} :player{age: 42, name: "Tim Duncan"}) |
+-----------------------------------------------------------------------------------------+
```

匹配点的属性

用户可以在 Tag 的右侧用 `{<prop_name>: <prop_value>}` 表示模式中点的属性。

```
# 使用属性 name 搜索匹配的点。
nebula> MATCH (v:player{name:"Tim Duncan"}) \
        RETURN v;
+-------------------------------------------------+
| v                                               |
+-------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) |
+-------------------------------------------------+
```

使用 `WHERE` 子句也可以实现相同的操作：

```
nebula> MATCH (v:player) \
        WHERE v.player.name == "Tim Duncan" \
        RETURN v;
+-------------------------------------------------+
| v                                               |
+-------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) |
+-------------------------------------------------+
```

> **openCypher** 兼容性

在 openCypher 9 中，`=` 是相等运算符，在 nGQL 中，`==` 是相等运算符，`=` 是赋值运算符。

使用 `WHERE` 子句直接匹配点的属性。

```
nebula> MATCH (v) \
        WITH v, properties(v) as props, keys(properties(v)) as kk \
        WHERE [i in kk where props[i] == "Tim Duncan"] \
        RETURN v;
+-------------------------------------------------+
| v                                               |
+-------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) |
+-------------------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

匹配点 ID

用户可以使用点 ID 去匹配点。`id()` 函数可以检索点的 ID。

```
nebula> MATCH (v) \
        WHERE id(v) == 'player101' \
        RETURN v;
+--------------------------------------------------+
| v                                                |
+--------------------------------------------------+
| ("player101" :player{age: 36, name: "Tony Parker"}) |
+--------------------------------------------------+
```

要匹配多个点的 ID，可以用 `WHERE id(v) IN [vid_list]` 或者 `WHERE id(v) IN {vid_list}` 。

```
nebula> MATCH (v:player { name: 'Tim Duncan' })--(v2) \
        WHERE id(v2) IN ["player101", "player102"] \
        RETURN v2;
+----------------------------------------------------------+
| v2                                                       |
+----------------------------------------------------------+
| ("player101" :player{age: 36, name: "Tony Parker"})      |
| ("player101" :player{age: 36, name: "Tony Parker"})      |
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
+----------------------------------------------------------+

nebula> MATCH (v) WHERE id(v) IN {"player100", "player101"} \
        RETURN v.player.name AS name;
+--------------+
| name         |
+--------------+
| "Tony Parker" |
| "Tim Duncan"  |
+--------------+
```

匹配连接的点

用户可以使用 `--` 符号表示两个方向的边，并匹配这些边连接的点。

> 版本兼容性

在 nGQL 1.x 中，`--` 符号用于行内注释，从 nGQL 2.x 起，`--` 符号表示出边或入边，不再用于注释。

```
nebula> MATCH (v:player{name:"Tim Duncan"})--(v2:player) \
        RETURN v2.player.name AS Name;
+--------------------+
| Name               |
+--------------------+
| "Manu Ginobili"    |
| "Manu Ginobili"    |
| "Dejounte Murray"  |
...
```

用户可以在 `--` 符号上增加 `<` 或 `>` 符号指定边的方向。

```
# -->表示边从 v 开始，指向 v2。对于点 v 来说是出边，对于点 v2 来说是入边。
nebula> MATCH (v:player{name:"Tim Duncan"})-->(v2:player) \
        RETURN v2.player.name AS Name;
+----------------+
| Name           |
+----------------+
| "Tony Parker"  |
| "Manu Ginobili" |
+----------------+
```

如果需要判断目标点，可以使用 `CASE` 表达式。

```
nebula> MATCH (v:player{name:"Tim Duncan"})--(v2) \
        RETURN \
        CASE WHEN v2.team.name IS NOT NULL \
        THEN v2.team.name  \
        WHEN v2.player.name IS NOT NULL \
        THEN v2.player.name END AS Name;

+--------------------+
| Name               |
+--------------------+
| "Manu Ginobili"    |
| "Manu Ginobili"    |
| "Spurs"            |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "Dejounte Murray"   |
...
```

如果需要扩展模式，可以增加更多点和边。

```
nebula> MATCH (v:player{name:"Tim Duncan"})-->(v2)<--(v3) \
        RETURN v3.player.name AS Name;
+--------------------+
| Name               |
+--------------------+
| "Dejounte Murray"  |
| "LaMarcus Aldridge" |
| "Marco Belinelli"  |
...
```

如果不需要引用点，可以省略括号中表示点的变量。

```
nebula> MATCH (v:player{name:"Tim Duncan"})-->()<--(v3) \
        RETURN v3.player.name AS Name;
+--------------------+
| Name               |
+--------------------+
| "Dejounte Murray"  |
| "LaMarcus Aldridge" |
| "Marco Belinelli"  |
...
```

匹配路径

连接起来的点和边构成了路径。用户可以使用自定义变量命名路径。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-->(v2) \
        RETURN p;
+------------------------------------------------------------------------------------------------------------------------------------------------------------+
| p                                                                                                                                                          |
+------------------------------------------------------------------------------------------------------------------------------------------------------------+
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:serve@0 {end_year: 2016, start_year: 1997}]->("team204" :team{name: "Spurs"})> |
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})>   |
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41, name: "Manu Ginobili"})> |
+------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

openCypher 兼容性

在 nGQL 中，`@` 符号表示边的 rank，在 openCypher 中，没有 rank 概念。

匹配边

```
nebula> MATCH ()<-[e]-() \
        RETURN e \
        LIMIT 3;
+-------------------------------------------------+
| e                                               |
+-------------------------------------------------+
| [:follow "player101"->"player102" @0 {degree: 90}] |
| [:follow "player103"->"player102" @0 {degree: 70}] |
| [:follow "player135"->"player102" @0 {degree: 80}] |
+-------------------------------------------------+
```

匹配 **EDGE TYPE**

和点一样，用户可以用 `:<edge_type>` 表示模式中的 Edge type，例如 `-[e:follow]-`。

历史版本兼容性

在悦数图数据库 3.0.0 之前，匹配 Edge Type 的前提是 Edge type 本身有对应属性的索引，否则，用户无法基于 Edge Type 执行 `MATCH` 语句。从悦数图数据库 3.0.0 开始，匹配 Edge Type 可以不创建索引，但需要使用 `LIMIT` 限制输出结果数量，并且必须指定边的方向。从悦数图数据库 3.5.0 开始，无需为 Edge Type 创建索引或者使用 `LIMIT` 限制输出结果数量，即可使用 `MATCH` 语句匹配边。

```
nebula> MATCH ()-[e:follow]->() \
        RETURN e;
+-------------------------------------------------+
| e                                               |
+-------------------------------------------------+
| [:follow "player102"->"player100" @0 {degree: 75}] |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| [:follow "player102"->"player101" @0 {degree: 75}] |
| [:follow "player129"->"player116" @0 {degree: 90}] |
...
```

匹配边的属性

用户可以用 `{<prop_name>: <prop_value>}` 表示模式中 Edge type 的属性，例如 `[e:follow{likeness:95}]` 。

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e:follow{degree:95}]->(v2) \
        RETURN e;
+-------------------------------------------------------+
| e                                                     |
+-------------------------------------------------------+
| [:follow "player100"->"player101" @0 {degree: 95}]    |
| [:follow "player100"->"player125" @0 {degree: 95}]    |
+-------------------------------------------------------+
```

使用 `WHERE` 子句直接匹配边的属性。

```
nebula> MATCH ()-[e]->() \
        WITH e, properties(e) as props, keys(properties(e)) as kk \
        WHERE [i in kk where props[i] == 90] \
        RETURN e;
+-----------------------------------------------------+
| e                                                   |
+-----------------------------------------------------+
| [:follow "player125"->"player100" @0 {degree: 90}] |
| [:follow "player140"->"player114" @0 {degree: 90}] |
| [:follow "player133"->"player144" @0 {degree: 90}] |
| [:follow "player133"->"player114" @0 {degree: 90}] |
...
+-----------------------------------------------------+
```

匹配多个 **EDGE TYPE**

使用 `|` 可以匹配多个 Edge type，例如 `[e:follow|:serve]` 。第一个 Edge type 前的英文冒号（`:`）不可省略，后续 Edge type 前的英文冒号可以省略，例如 `[e:follow|serve]` 。

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e:follow|:serve]->(v2) \
        RETURN e;
+-----------------------------------------------------------------------+
| e                                                                     |
+-----------------------------------------------------------------------+
| [:follow "player100"->"player101" @0 {degree: 95}]                    |
| [:follow "player100"->"player125" @0 {degree: 95}]                    |
| [:serve "player100"->"team204" @0 {end_year: 2016, start_year: 1997}] |
+-----------------------------------------------------------------------+
```

匹配多条边

用户可以扩展模式，匹配路径中的多条边。

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[]->(v2)<-[e:serve]-(v3) \
        RETURN v2, v3;
+----------------------------------+-------------------------------------------------------+
| v2                               | v3                                                    |
+----------------------------------+-------------------------------------------------------+
| ("team204" :team{name: "Spurs"}) | ("player104" :player{age: 32, name: "Marco Belinelli"}) |
| ("team204" :team{name: "Spurs"}) | ("player101" :player{age: 36, name: "Tony Parker"})     |
| ("team204" :team{name: "Spurs"}) | ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
...
```

匹配定长路径

用户可以在模式中使用 `:<edge_type>*<hop>` 匹配定长路径。 `hop` 必须是一个非负整数。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
        RETURN DISTINCT v2 AS Friends;
+-------------------------------------------------------+
| Friends                                               |
+-------------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"})    |
| ("player125" :player{age: 41, name: "Manu Ginobili"}) |
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
+-------------------------------------------------------+
```

如果 `hop` 为 0，模式会匹配路径上的起始点。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) -[*0]-> (v2) \
        RETURN v2;
+-----------------------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| v2                                                |
+---------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) |
+---------------------------------------------------+
```

Note

在对匹配的多跳边进行过滤时，如对 `-[e:follow*2]->` 中的 `e` 进行过滤，此时的 `e` 不再是单条边的数据类型，而是一个包含多条边的列表，例如：

以下语句可以运行但是没有返回数据，因为 `e` 是一个列表，没有 `.degree` 的属性。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
        WHERE e.degree > 1 \
        RETURN DISTINCT v2 AS Friends;
```

这是正确的表达：

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
        WHERE ALL(e_ in e WHERE e_.degree > 0) \
        RETURN DISTINCT v2 AS Friends;
```

进一步，这是表达对多跳边的第一跳的边属性过滤的表达：

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
        WHERE e[0].degree > 98 \
        RETURN DISTINCT v2 AS Friends;
```

匹配变长路径

用户可以在模式中使用 `:<edge_type>*[minHop..maxHop]` 匹配变长路径。

Caution

如果未设置 `maxHop` 可能会导致 graph 服务 OOM，请谨慎执行该命令。

参数	说明
`minHop`	可选项。表示路径的最小长度。`minHop` 必须是一个非负整数，默认值为 1。
`maxHop`	可选项。表示路径的最大长度。`maxHop` 必须是一个非负整数，默认值为无穷大。

如果未指定 `minHop` 和 `maxHop`，仅设置了 `:<edge_type>*`，则二者都应用默认值，即 `minHop` 为 1，`maxHop` 为无穷大。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*]->(v2) \
        RETURN v2 AS Friends;
+-------------------------------------------------------+
| Friends                                               |
+-------------------------------------------------------+
| ("player125" :player{age: 41, name: "Manu Ginobili"})     |
| ("player101" :player{age: 36, name: "Tony Parker"})       |
...

nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*1..3]->(v2) \
        RETURN v2 AS Friends;
+-------------------------------------------------------+
| Friends                                               |
+-------------------------------------------------------+
| ("player101" :player{age: 36, name: "Tony Parker"})       |
| ("player125" :player{age: 41, name: "Manu Ginobili"})     |
| ("player100" :player{age: 42, name: "Tim Duncan"})        |
...

nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*1..]->(v2) \
        RETURN v2 AS Friends;
+-------------------------------------------------------+
| Friends                                               |
+-------------------------------------------------------+
| ("player125" :player{age: 41, name: "Manu Ginobili"})     |
| ("player101" :player{age: 36, name: "Tony Parker"})       |
| ("player100" :player{age: 42, name: "Tim Duncan"})        |
...
```

用户可以使用 `DISTINCT` 关键字聚合重复结果。

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*1..3]->(v2:player) \
        RETURN DISTINCT v2 AS Friends, count(v2);
+-------------------------------------------------------+-----------+
| Friends                                               | count(v2) |
+-------------------------------------------------------+-----------+
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) | 1         |
| ("player100" :player{age: 42, name: "Tim Duncan"})        | 4         |
| ("player101" :player{age: 36, name: "Tony Parker"})       | 3         |
| ("player125" :player{age: 41, name: "Manu Ginobili"})     | 3         |
+-------------------------------------------------------+-----------+
```

如果 `minHop` 为 `0` ，模式会匹配路径上的起始点。例如，与上个示例相比，下面的示例设置 `minHop` 为 `0` 。此时，因为表示 `"Tim Duncan"` 的点是路径的起始点，所以它在结果集中的计数为 5，比在上个示例的结果中多计一次。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*0..3]->(v2:player) \
        RETURN DISTINCT v2 AS Friends, count(v2);
+-------------------------------------------------------+-----------+
| Friends                                               | count(v2) |
+-------------------------------------------------------+-----------+
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) | 1         |
| ("player100" :player{age: 42, name: "Tim Duncan"})        | 5         |
| ("player125" :player{age: 41, name: "Manu Ginobili"})     | 3         |
| ("player101" :player{age: 36, name: "Tony Parker"})       | 3         |
+-------------------------------------------------------+-----------+
```

Note

当在模式中使用变量 `e` 匹配定长或者变长路径时，例如 `-[e:follow*0..3]->` ，不支持在其他模式中引用 `e` 。例如，不支持以下语句：

```
nebula> MATCH (v:player)-[e:like*1..3]->(n) \
        WHERE (n)-[e*1..4]->(:player) \
        RETURN v;
```

匹配多个 **EDGE TYPE** 的变长路径

用户可以在变长或定长模式中指定多个 Edge type。 `hop` 、 `minHop` 和 `maxHop` 对所有 Edge type 都生效。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow|serve*2]->(v2) \
        RETURN DISTINCT v2;
+-------------------------------------------------------+
| v2                                                    |
+-------------------------------------------------------+
| ("team204" :team{name: "Spurs"})                      |
| ("player100" :player{age: 42, name: "Tim Duncan"})    |
| ("team215" :team{name: "Hornets"})                    |
| ("player125" :player{age: 41, name: "Manu Ginobili"})  |
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
+-------------------------------------------------------+
```

匹配多个模式

用户可以用英文逗号（`,`）分隔多个模式。

```
nebula> CREATE TAG INDEX IF NOT EXISTS team_index ON team(name(20));
nebula> REBUILD TAG INDEX team_index;
nebula> MATCH (v1:player{name:"Tim Duncan"}), (v2:team{name:"Spurs"}) \
        RETURN v1,v2;
+-------------------------------------------------+--------------------------------+
| v1                                              | v2                             |
+-------------------------------------------------+--------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) | ("team204" :team{name: "Spurs"}) |
+-------------------------------------------------+--------------------------------+
```

匹配最短路径

用户可以使用 `allShortestPaths` 返回起始点到目标点的所有最短路径。

```
nebula> MATCH p = allShortestPaths((a:player{name:"Tim Duncan"})-[e*..5]-(b:player{name:"Tony Parker"})) \
        RETURN p;
+-------------------------------------------------------------------------------------------------------------------------+
| p                                                                                                                       |
+-------------------------------------------------------------------------------------------------------------------------+
| <("player100" :player{age: 42, name: "Tim Duncan"})<-[:follow@0 {degree: 95}]-("player101" :player{age: 36, name: "Tony Parker"})> |
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})> |
+-------------------------------------------------------------------------------------------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

用户可以使用 `shortestPath` 返回起始点到目标点的任意一条最短路径。

```
nebula> MATCH p = shortestPath((a:player{name:"Tim Duncan"})-[e*..5]-(b:player{name:"Tony Parker"})) \
        RETURN p;
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| p                                                                                                                                             |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| <("player100" :player{age: 42, name: "Tim Duncan"})<-[:follow@0 {degree: 95}]-("player101" :player{age: 36, name: "Tony Parker"})> |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
```

多MATCH检索

不同的模式有不同的筛选条件时，可以使用多 `MATCH`，会返回模式完全匹配的行。

```
nebula> MATCH (m)-[]->(n) WHERE id(m)=="player100" \
        MATCH (n)-[]->(l) WHERE id(n)=="player125" \
        RETURN id(m),id(n),id(l);
+-------------+-------------+-------------+
| id(m)       | id(n)       | id(l)       |
+-------------+-------------+-------------+
| "player100" | "player125" | "team204"   |
| "player100" | "player125" | "player100" |
+-------------+-------------+-------------+
```

OPTIONAL MATCH检索

参见OPTIONAL MATCH。

Performance

悦数图数据库 3.5.1 中 `MATCH` 语句的性能和资源占用得到了优化.但对性能要求较高时，仍建议使用 `GO`, `LOOKUP`, `|` 和 `FETCH` 等来替代 `MATCH`。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.6.2 OPTIONAL MATCH

Caution

目前 `OPTIONAL MATCH` 为 Beta 功能，后续可能会有一定优化调整。

`OPTIONAL MATCH` 通常与 `MATCH` 语句一起使用，作为 `MATCH` 语句的可选项去匹配命中的模式，如果没有命中对应的模式，对应的列返回 `NULL` 。

openCypher 兼容性

本文操作仅适用于 nGQL 中的 openCypher 方式。

使用限制

`OPTIONAL MATCH` 子句中暂不支持使用 `WHERE` 子句。

示例

`MATCH` 语句中使用 `OPTIONAL MATCH` 的示例如下：

```
nebula> MATCH (m)-[]->(n) WHERE id(m)=="player100" \
        OPTIONAL MATCH (n)-[]->(l) \
        RETURN id(m),id(n),id(l);
+-------------+-------------+-------------+
| id(m)       | id(n)       | id(l)       |
+-------------+-------------+-------------+
| "player100" | "team204"   | __NULL__    |
| "player100" | "player101" | "team204"   |
| "player100" | "player101" | "team215"   |
| "player100" | "player101" | "player100" |
| "player100" | "player101" | "player102" |
| "player100" | "player101" | "player125" |
| "player100" | "player125" | "team204"   |
| "player100" | "player125" | "player100" |
+-------------+-------------+-------------+
```

而使用多 `MATCH` ，不使用 `OPTIONAL MATCH` 时，会返回模式完全匹配的行。示例如下：

```
nebula> MATCH (m)-[]->(n) WHERE id(m)=="player100" \
        MATCH (n)-[]->(l) \
        RETURN id(m),id(n),id(l);
+-------------+-------------+-------------+
| id(m)       | id(n)       | id(l)       |
+-------------+-------------+-------------+
| "player100" | "player101" | "team204"   |
| "player100" | "player101" | "team215"   |
| "player100" | "player101" | "player100" |
| "player100" | "player101" | "player102" |
| "player100" | "player101" | "player125" |
| "player100" | "player125" | "team204"   |
| "player100" | "player125" | "player100" |
+-------------+-------------+-------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.6.3 LOOKUP

`LOOKUP` 根据索引遍历数据。用户可以使用 `LOOKUP` 实现如下功能：

- 根据 `WHERE` 子句搜索特定数据。
- 通过 Tag 列出点：检索指定 Tag 的所有点 ID。
- 通过 Edge type 列出边：检索指定 Edge type 的所有边的起始点、目的点和 rank。
- 统计包含指定 Tag 的点或属于指定 Edge type 的边的数量。

OpenCypher 兼容性

本文操作仅适用于原生 nGQL。

注意事项

- 索引会导致写性能大幅降低。请不要随意在生产环境中使用索引，除非很清楚使用索引对业务的影响。
- 通过 `Explain` 命令查看选择的索引。

> 版本兼容性
>
> 在 2.5.0 版本之前，如果用 `LOOKUP` 语句基于指定属性查询时该属性没有索引，系统将报错，而不会使用其它索引。

前提条件

请确保 `LOOKUP` 语句有至少一个索引可用。

如果已经存在相关的点、边或属性，必须在新创建索引后重建索引，才能使其生效。

语法

```
LOOKUP ON {<vertex_tag> | <edge_type>}
[WHERE <expression> [AND <expression> ...]]
YIELD [DISTINCT] <return_list> [AS <alias>]
[<clause>];

<return_list>
    <prop_name> [AS <col_alias>] [, <prop_name> [AS <prop_alias>] ...];
```

- `WHERE <expression>`： 指定遍历的过滤条件，还可以结合布尔运算符 AND 和 OR 一起使用。详情请参见 WHERE。
- `YIELD`： 定义需要返回的输出。详情请参见 `YIELD`。
- `DISTINCT`： 聚合输出结果，返回去重后的结果集。
- `AS`： 设置别名。
- `clause`： 支持 `ORDER BY` 、`LIMIT` 子句。

Copyright © Yueshu Graph Technology - All rights reserved.

WHERE 语句限制

在 `LOOKUP` 语句中使用 `WHERE` 子句，不支持如下操作：

- `$-` 和 `$^` 。
- 在关系表达式中，不支持运算符两边都有字段名，例如 `tagName.prop1 > tagName.prop2` 。
- 不支持运算表达式和函数表达式中嵌套 AliasProp 表达式。
- 不支持 XOR 运算符。
- 不支持除 `STARTS WITH` 之外的字符串操作。
- 不支持过滤 `rank()` 。
- 不支持图模式.

检索点

返回 Tag 为 `player` 且 `name` 为 `Tony Parker` 的点。

```
nebula> CREATE TAG INDEX IF NOT EXISTS index_player ON player(name(30), age);

nebula> REBUILD TAG INDEX index_player;
+------------+
| New Job Id |
+------------+
| 15         |
+------------+

nebula> LOOKUP ON player \
        WHERE player.name == "Tony Parker" \
        YIELD id(vertex);
+---------------+
| id(VERTEX)    |
+---------------+
| "player101"   |
+---------------+

nebula> LOOKUP ON player \
        WHERE player.name == "Tony Parker" \
        YIELD properties(vertex).name AS name, properties(vertex).age AS age;
+---------------+-----+
| name          | age |
+---------------+-----+
| "Tony Parker" | 36  |
+---------------+-----+

nebula> LOOKUP ON player \
        WHERE player.age  > 45 \
        YIELD id(vertex);
+-------------+
| id(VERTEX)  |
+-------------+
| "player144" |
| "player140" |
+-------------+

nebula> LOOKUP ON player \
        WHERE player.name STARTS WITH "B" \
        AND player.age IN [22,30] \
        YIELD properties(vertex).name, properties(vertex).age;
+-------------------------+------------------------+
| properties(VERTEX).name | properties(VERTEX).age |
+-------------------------+------------------------+
| "Ben Simmons"           | 22                     |
| "Blake Griffin"         | 30                     |
+-------------------------+------------------------+

nebula> LOOKUP ON player \
        WHERE player.name == "Kobe Bryant"\
        YIELD id(vertex) AS VertexID, properties(vertex).name AS name |\
        GO FROM $-.VertexID OVER serve \
        YIELD $-.name, properties(edge).start_year, properties(edge).end_year, properties($$).name;
+---------------+----------------------------+--------------------------+---------------------+
| $-.name       | properties(EDGE).start_year | properties(EDGE).end_year | properties($$).name |
+---------------+----------------------------+--------------------------+---------------------+
| "Kobe Bryant" | 1996                       | 2016                     | "Lakers"            |
+---------------+----------------------------+--------------------------+---------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

检索边

返回 Edge type 为 `follow` 且 `degree` 为 `90` 的边。

```
nebula> CREATE EDGE INDEX IF NOT EXISTS index_follow ON follow(degree);

nebula> REBUILD EDGE INDEX index_follow;
+------------+
| New Job Id |
+------------+
| 62         |
+------------+

nebula> LOOKUP ON follow \
        WHERE follow.degree == 90 YIELD edge AS e;
+--------------------------------------------------+
| e                                                |
+--------------------------------------------------+
| [:follow "player109"->"player125" @0 {degree: 90}] |
| [:follow "player118"->"player120" @0 {degree: 90}] |
| [:follow "player118"->"player131" @0 {degree: 90}] |
...

nebula> LOOKUP ON follow \
        WHERE follow.degree == 90 \
        YIELD properties(edge).degree;
+------------------------+
| properties(EDGE).degree |
+------------------------+
| 90                     |
| 90                     |
...

nebula> LOOKUP ON follow \
        YIELD properties(edge).degree as degree \
        | ORDER BY $-.degree \
        | LIMIT 10;
+--------+
| degree |
+--------+
| -1     |
| -1     |
| 9      |
| 10     |
| 13     |
| 50     |
| 55     |
| 60     |
| 70     |
| 70     |
+--------+

nebula> LOOKUP ON follow \
        WHERE follow.degree == 60 \
        YIELD dst(edge) AS DstVID, properties(edge).degree AS Degree |\
        GO FROM $-.DstVID OVER serve \
        YIELD $-.DstVID, properties(edge).start_year, properties(edge).end_year, properties($$).name;
+-------------+----------------------------+---------------------------+---------------------+
| $-.DstVID   | properties(EDGE).start_year | properties(EDGE).end_year | properties($$).name |
+-------------+----------------------------+---------------------------+---------------------+
| "player105" | 2010                       | 2018                      | "Spurs"             |
| "player105" | 2009                       | 2010                      | "Cavaliers"         |
| "player105" | 2018                       | 2019                      | "Raptors"           |
+-------------+----------------------------+---------------------------+---------------------+
```

通过 **Tag** 列出所有的对应的点/通过 **Edge type** 列出边

如果需要通过 Tag 列出所有的点，或通过 Edge type 列出边，则 Tag、Edge type 或属性上必须有至少一个索引。

Copyright © Yueshu Graph Technology - All rights reserved.

例如一个 Tag `player` 有属性 `name` 和 `age`，为了遍历所有包含 Tag `player` 的点 ID，Tag `player` 、属性 `name` 或属性 `age` 中必须有一个已经创建索引。

- 查找所有 Tag 为 `player` 的点 VID。

```
nebula> CREATE TAG IF NOT EXISTS player(name string,age int);

nebula> CREATE TAG INDEX IF NOT EXISTS player_index on player();

nebula> REBUILD TAG INDEX player_index;
+------------+
| New Job Id |
+------------+
| 66         |
+------------+

nebula> INSERT VERTEX player(name,age) \
        VALUES "player100":("Tim Duncan", 42), "player101":("Tony Parker", 36);

# 列出所有的 player。类似于 MATCH (n:player) RETURN id(n) /*, n */。

nebula> LOOKUP ON player YIELD id(vertex);
+-------------+
| id(VERTEX)  |
+-------------+
| "player100" |
| "player101" |
...

# 从结果中返回最前面的 4 行数据。
nebula> LOOKUP ON player YIELD id(vertex) | LIMIT 4;
+-------------+
| id(VERTEX)  |
+-------------+
| "player105" |
| "player109" |
| "player111" |
| "player118" |
+-------------+
```

- 查找 Edge type 为 `follow` 的所有边的信息。

```
nebula> CREATE EDGE IF NOT EXISTS follow(degree int);

nebula> CREATE EDGE INDEX IF NOT EXISTS follow_index on follow();

nebula> REBUILD EDGE INDEX follow_index;
+------------+
| New Job Id |
+------------+
| 88         |
+------------+

nebula> INSERT EDGE follow(degree) \
        VALUES "player100"->"player101":(95);

# 列出所有的 follow 边。类似于 MATCH (s)-[e:follow]->(d) RETURN id(s), rank(e), id(d) /*, type(e) */。

nebula)> LOOKUP ON follow YIELD edge AS e;
+-------------------------------------------------------+
| e                                                     |
+-------------------------------------------------------+
| [:follow "player105"->"player100" @0 {degree: 70}]  |
| [:follow "player105"->"player116" @0 {degree: 80}]  |
| [:follow "player109"->"player100" @0 {degree: 80}]  |
...
```

统计点或边

统计 Tag 为 `player` 的点和 Edge type 为 `follow` 的边。

```
nebula> LOOKUP ON player YIELD id(vertex)|\
        YIELD COUNT(*) AS Player_Number;
+---------------+
| Player_Number |
+---------------+
| 51            |
+---------------+

nebula> LOOKUP ON follow YIELD edge AS e| \
        YIELD COUNT(*) AS Follow_Number;
+---------------+
| Follow_Number |
+---------------+
| 81            |
+---------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

Note

使用 `SHOW STATS` 命令也可以统计点和边。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.6.4 GO

`GO` 语句是悦数图数据库图数据库中用于从给定起始点开始遍历图的语句。`GO` 语句采用的路径类型是 `walk`，即遍历时点和边都可以重复。

openCypher 兼容性

本文操作仅适用于原生 nGQL。

语法

```
GO [[<M> TO] <N> {STEP|STEPS} ] FROM <vertex_list>
OVER <edge_type_list> [{REVERSELY | BIDIRECT}]
[ WHERE <conditions> ]
YIELD [DISTINCT] <return_list>
[{ SAMPLE <sample_list> | <limit_by_list_clause> }]
[| GROUP BY {<col_name> | expression> | <position>} YIELD <col_name>]
[| ORDER BY <expression> [{ASC | DESC}]]
[| LIMIT [<offset>,] <number_rows>];

<vertex_list> ::=
    <vid> [, <vid> ...]

<edge_type_list> ::=
   <edge_type> [, <edge_type> ...]
   | *
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
<return_list> ::=
    <col_name> [AS <col_alias>] [, <col_name> [AS <col_alias>] ...]
```

- `<N> {STEP|STEPS}`： 指定跳数。如果没有指定跳数，默认值 `N` 为 `1` 。如果 `N` 为 `0` ，悦数图数据库不会检索任何边。
- `M TO N {STEP|STEPS}`： 遍历 `M~N` 跳的边。如果 `M` 为 `0` ，输出结果和 `M` 为 `1` 相同，即 `GO 0 TO 2` 和 `GO 1 TO 2` 是相同的。
- `<vertex_list>`： 用逗号分隔的点 ID 列表。
- `<edge_type_list>`： 遍历的 Edge type 列表。
- `REVERSELY | BIDIRECT`： 默认情况下检索的是 `<vertex_list>` 的出边（正向）， `REVERSELY` 表示反向，即检索入边； `BIDIRECT` 为双向，即检索正向和反向。可通过 `YIELD` 返回 `<edge_type>._type` 字段判断方向，其正数为正向，负数为反向。
- `WHERE <conditions>`： 指定遍历的过滤条件。用户可以在起始点、目的点和边使用 `WHERE` 子句，还可以结合 `AND` 、 `OR` 、 `NOT` 、 `XOR` 一起使用。详情参见 WHERE。

  Note

  - 遍历多个 Edge type 时， `WHERE` 子句有一些限制。例如不支持 `WHERE edge1.prop1 > edge2.prop2` 。
  - GO 语句执行时先遍历所有的点，然后再根据过滤器条件进行过滤。

- `YIELD [DISTINCT] <return_list>`： 定义需要返回的输出。 `<return_list>` 建议使用 Schema 相关函数指定返回信息，当前支持 `src(edge)` 、 `dst(edge)` 、 `type(edge)` 等，暂不支持嵌套函数。详情参见 YIELD。
- `SAMPLE <sample_list>`： 用于在结果集中取样。详情参见 SAMPLE。
- `<limit_by_list_clause>`： 用于在遍历过程中逐步限制输出数量。详情参见 LIMIT。
- `GROUP BY`： 根据指定属性的值将输出分组。详情参见 GROUP BY。分组后需要再次使用 `YIELD` 定义需要返回的输出。
- `ORDER BY`： 指定输出结果的排序规则。详情参见 ORDER BY。

  Note

  没有指定排序规则时，输出结果的顺序不是固定的。

- `LIMIT [<offset>,] <number_rows>]`： 限制输出结果的行数。详情参见 LIMIT。

使用说明

- GO 语句中的 `WHERE` 和 `YIELD` 子句通常结合属性引用符(`$^` 和 `$$`)或函数 `properties($^)` 和 `properties($$)` 指定点的属性；使用函数 `properties(edge)` 指定边的属性。用法参见属性引用符和 Schema 相关函数。
- GO 复合语句中如需引用子查询的结果，需要为该结果设置别名，并使用管道符 `|` 传递给下一个子查询，同时在下一个子查询中使用 `$-` 引用该结果的别名。详情参见管道符。
- 当查询属性没有值时，返回结果显示 `NULL` 。

场景及示例

查询起始点的直接邻居点

场景：查询某个点的直接相邻点，例如查询一个人所属队伍。

```
# 返回 player102 所属队伍。
nebula> GO FROM "player102" OVER serve YIELD dst(edge);
+-----------+
| dst(EDGE) |
+-----------+
| "team203" |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "team204" |
+-----------+
```

查询指定跳数内的点

场景：查询一个点在指定跳数内的所有点，例如查询一个人两跳内的朋友。

```
# 返回距离 player102 两跳的朋友。
nebula> GO 2 STEPS FROM "player102" OVER follow YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player101" |
| "player125" |
| "player100" |
| "player102" |
| "player125" |
+-------------+
```

```
# 查询 player100 1~2 跳内的朋友。
nebula> GO 1 TO 2 STEPS FROM "player100" OVER follow \
        YIELD dst(edge) AS destination;
+-------------+
| destination |
+-------------+
| "player101" |
| "player125" |
...

# 该 MATCH 查询与上一个 GO 查询具有相同的语义。
nebula> MATCH (v) -[e:follow*1..2]->(v2) \
        WHERE id(v) == "player100" \
        RETURN id(v2) AS destination;
+-------------+
| destination |
+-------------+
| "player100" |
| "player102" |
...
```

添加过滤条件

场景：查询满足特定条件的点和边，例如查询起始点和目的点之间具有特定属性的边。

```
# 使用 WHERE 添加过滤条件。
nebula> GO FROM "player100", "player102" OVER serve \
        WHERE properties(edge).start_year > 1995 \
        YIELD DISTINCT properties($$).name AS team_name, properties(edge).start_year AS start_year, properties($^).name AS player_name;

+-----------------+------------+--------------------+
| team_name       | start_year | player_name        |
+-----------------+------------+--------------------+
| "Spurs"         | 1997       | "Tim Duncan"       |
| "Trail Blazers" | 2006       | "LaMarcus Aldridge" |
| "Spurs"         | 2015       | "LaMarcus Aldridge" |
+-----------------+------------+--------------------+
```

查询多个 **EDGE TYPE**

场景：查询起始点关联的多个边类型可以通过设置多个Edge Type实现，也可以通过设置 `*` 关联所有的边类型。

```
# 遍历多个 Edge type。
nebula> GO FROM "player100" OVER follow, serve \
        YIELD properties(edge).degree, properties(edge).start_year;
+-------------------------+----------------------------+
| properties(EDGE).degree | properties(EDGE).start_year |
+-------------------------+----------------------------+
| 95                      | __NULL__                   |
| 95                      | __NULL__                   |
| __NULL__                | 1997                       |
+-------------------------+----------------------------+
```

查询入边方向的点

```
# 返回关注 player100 的邻居点。
nebula> GO FROM "player100" OVER follow REVERSELY \
        YIELD src(edge) AS destination;
+-------------+
| destination |
+-------------+
| "player101" |
| "player102" |
...

# 该 MATCH 查询与上一个 GO 查询具有相同的语义。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> MATCH (v)<-[e:follow]- (v2) WHERE id(v) == 'player100' \
        RETURN id(v2) AS destination;
+-------------+
| destination |
+-------------+
| "player101" |
| "player102" |
...
```

子查询作为起始点

场景：使用子查询的结果作为图遍历的起始点。

```
# 查询 player100 的朋友和朋友所属队伍。
nebula> GO FROM "player100" OVER follow REVERSELY \
        YIELD src(edge) AS id | \
        GO FROM $-.id OVER serve \
        WHERE properties($^).age > 20 \
        YIELD properties($^).name AS FriendOf, properties($$).name AS Team;
+--------------------+----------------+
| FriendOf           | Team           |
+--------------------+----------------+
| "Boris Diaw"       | "Spurs"        |
| "Boris Diaw"       | "Jazz"         |
| "Boris Diaw"       | "Suns"         |
...

# 该 MATCH 查询与上一个 GO 查询具有相同的语义。
nebula> MATCH (v)<-[e:follow]- (v2)-[e2:serve]->(v3)  \
        WHERE id(v) == 'player100' \
        RETURN v2.player.name AS FriendOf, v3.team.name AS Team;
+--------------------+----------------+
| FriendOf           | Team           |
+--------------------+----------------+
| "Boris Diaw"       | "Spurs"        |
| "Boris Diaw"       | "Jazz"         |
| "Boris Diaw"       | "Suns"         |
...
```

使用 **GROUP BY** 分组

场景：使用 `GROUP BY` 分组，然后使用 `YIELD` 返回分组后的结果。

```
# 根据年龄分组。
nebula> GO 2 STEPS FROM "player100" OVER follow \
        YIELD src(edge) AS src, dst(edge) AS dst, properties($$).age AS age \
        | GROUP BY $-.dst \
        YIELD $-.dst AS dst, collect_set($-.src) AS src, collect($-.age) AS age;
+-------------+---------------------------+----------+
| dst         | src                       | age      |
+-------------+---------------------------+----------+
| "player125" | {"player101"}             | [41]     |
| "player100" | {"player125", "player101"} | [42, 42] |
| "player102" | {"player101"}             | [33]     |
+-------------+---------------------------+----------+
```

使用 **ORDER BY** 和 **LIMIT** 排序和限制输出结果

```
# 分组并限制输出结果的行数。
nebula> $a = GO FROM "player100" OVER follow YIELD src(edge) AS src, dst(edge) AS dst; \
        GO 2 STEPS FROM $a.dst OVER follow \
        YIELD $a.src AS src, $a.dst, src(edge), dst(edge) \
        | ORDER BY $-.src | OFFSET 1 LIMIT 2;
+-------------+-------------+-------------+-------------+
| src         | $a.dst      | src(EDGE)   | dst(EDGE)   |
+-------------+-------------+-------------+-------------+
| "player100" | "player101" | "player100" | "player101" |
| "player100" | "player125" | "player100" | "player125" |
+-------------+-------------+-------------+-------------+
```

其他用法

```
# 在多个边上通过 IS NOT EMPTY 进行判断。
nebula> GO FROM "player100" OVER follow WHERE properties($$).name IS NOT EMPTY YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player125" |
| "player101" |
+-------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.6.5 FETCH

`FETCH` 可以获取指定点或边的属性值。

openCypher 兼容性

本文操作仅适用于原生 nGQL。

获取点的属性值

语法

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *}
<vid> [, vid ...]
YIELD [DISTINCT] <return_list> [AS <alias>];
```

参数	说明
`tag_name`	Tag 名称。
`*`	表示当前图空间中的所有 Tag。
`vid`	点 ID。
`YIELD`	定义需要返回的输出。详情请参见 `YIELD` 。
`AS`	设置别名。

基于 **TAG** 获取点的属性值

在 `FETCH` 语句中指定 Tag 获取对应点的属性值。

```
nebula> FETCH PROP ON player "player100" YIELD properties(vertex);
+------------------------------+
| properties(VERTEX)           |
+------------------------------+
| {age: 42, name: "Tim Duncan"} |
+------------------------------+
```

获取点的指定属性值

使用 `YIELD` 子句指定返回的属性。

```
nebula> FETCH PROP ON player "player100" \
        YIELD properties(vertex).name AS name;
+--------------+
| name         |
+--------------+
| "Tim Duncan" |
+--------------+
```

获取多个点的属性值

指定多个点 ID 获取多个点的属性值，点之间用英文逗号（,）分隔。

```
nebula> FETCH PROP ON player "player101", "player102", "player103" YIELD properties(vertex);
+--------------------------------------+
| properties(VERTEX)                   |
+--------------------------------------+
| {age: 33, name: "LaMarcus Aldridge"} |
| {age: 36, name: "Tony Parker"}       |
| {age: 32, name: "Rudy Gay"}          |
+--------------------------------------+
```

基于多个 **TAG** 获取点的属性值

在 `FETCH` 语句中指定多个 Tag 获取属性值。Tag 之间用英文逗号（,）分隔。

```
# 创建新 Tag t1。
nebula> CREATE TAG IF NOT EXISTS t1(a string, b int);
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# 为点 player100 添加 Tag t1。
nebula> INSERT VERTEX t1(a, b) VALUES "player100":("Hello", 100);

# 基于 Tag player 和 t1 获取点 player100 上的属性值。
nebula> FETCH PROP ON player, t1 "player100" YIELD vertex AS v;
+------------------------------------------------------------------------+
| v                                                                      |
+------------------------------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"} :t1{a: "Hello", b: 100}) |
+------------------------------------------------------------------------+
```

用户可以在 `FETCH` 语句中组合多个 Tag 和多个点。

```
nebula> FETCH PROP ON player, t1 "player100", "player103" YIELD vertex AS v;
+------------------------------------------------------------------------+
| v                                                                      |
+------------------------------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"} :t1{a: "Hello", b: 100}) |
| ("player103" :player{age: 32, name: "Rudy Gay"})                       |
+------------------------------------------------------------------------+
```

在所有标签中获取点的属性值

在 `FETCH` 语句中使用 `*` 获取当前图空间所有标签里，点的属性值。

```
nebula> FETCH PROP ON * "player100", "player106", "team200" YIELD vertex AS v;
+------------------------------------------------------------------------+
| v                                                                      |
+------------------------------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"} :t1{a: "Hello", b: 100}) |
| ("player106" :player{age: 25, name: "Kyle Anderson"})                  |
| ("team200" :team{name: "Warriors"})                                    |
+------------------------------------------------------------------------+
```

获取边的属性值

语法

```
FETCH PROP ON <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid> ...]
YIELD <output>;
```

参数	说明
`edge_type`	Edge type 名称。
`src_vid`	起始点 ID，表示边的起点。
`dst_vid`	目的点 ID，表示边的终点。
`rank`	边的 rank。可选参数，默认值为 `0`。起始点、目的点、Edge type 和 rank 可以唯一确定一条边。
`YIELD`	定义需要返回的输出。详情请参见 `YIELD`。

获取边的所有属性值

```
# 获取连接 player100 和 team204 的边 serve 的所有属性值。
nebula> FETCH PROP ON serve "player100" -> "team204" YIELD properties(edge);
+------------------------------------+
| properties(EDGE)                   |
+------------------------------------+
| {end_year: 2016, start_year: 1997} |
+------------------------------------+
```

获取边的指定属性值

使用 `YIELD` 子句指定返回的属性。

```
nebula> FETCH PROP ON serve "player100" -> "team204"    \
        YIELD properties(edge).start_year;
+----------------------------+
| properties(EDGE).start_year |
+----------------------------+
| 1997                       |
+----------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

获取多条边的属性值

指定多个边模式（`<src_vid> -> <dst_vid>[@<rank>]`）获取多个边的属性值。模式之间用英文逗号（,）分隔。

```
nebula> FETCH PROP ON serve "player100" -> "team204", "player133" -> "team202" YIELD edge AS e;
+----------------------------------------------------------------------+
| e                                                                    |
+----------------------------------------------------------------------+
| [:serve "player100"->"team204" @0 {end_year: 2016, start_year: 1997}] |
| [:serve "player133"->"team202" @0 {end_year: 2011, start_year: 2002}] |
+----------------------------------------------------------------------+
```

基于 **RANK** 获取属性值

如果有多条边，起始点、目的点和 Edge type 都相同，可以通过指定 rank 获取正确的边属性值。

```
# 插入不同属性值、不同 rank 的边。
nebula> insert edge serve(start_year,end_year) \
        values "player100"->"team204"@1:(1998, 2017);

nebula> insert edge serve(start_year,end_year) \
        values "player100"->"team204"@2:(1990, 2018);

# 默认返回 rank 为 0 的边。
nebula> FETCH PROP ON serve "player100" -> "team204" YIELD edge AS e;
+----------------------------------------------------------------------+
| e                                                                    |
+----------------------------------------------------------------------+
| [:serve "player100"->"team204" @0 {end_year: 2016, start_year: 1997}] |
+----------------------------------------------------------------------+

# 要获取 rank 不为 0 的边，请在 FETCH 语句中设置 rank。
nebula> FETCH PROP ON serve "player100" -> "team204"@1 YIELD edge AS e;
+----------------------------------------------------------------------+
| e                                                                    |
+----------------------------------------------------------------------+
| [:serve "player100"->"team204" @1 {end_year: 2017, start_year: 1998}] |
+----------------------------------------------------------------------+
```

复合语句中使用 FETCH

将 `FETCH` 与原生 nGQL 结合使用是一种常见的方式，例如和 `GO` 一起。

```
# 返回从点 player101 开始的 follow 边的 degree 值。
nebula> GO FROM "player101" OVER follow \
        YIELD src(edge) AS s, dst(edge) AS d \
        | FETCH PROP ON follow $-.s -> $-.d \
        YIELD properties(edge).degree;
+------------------------+
| properties(EDGE).degree |
+------------------------+
| 95                     |
| 90                     |
| 95                     |
+------------------------+
```

用户也可以通过自定义变量构建类似的查询。

```
nebula> $var = GO FROM "player101" OVER follow \
        YIELD src(edge) AS s, dst(edge) AS d; \
        FETCH PROP ON follow $var.s -> $var.d \
        YIELD properties(edge).degree;
+------------------------+
| properties(EDGE).degree |
+------------------------+
| 95                     |
| 90                     |
| 95                     |
+------------------------+
```

更多复合语句的详情，请参见复合查询（子句结构）。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.6.6 SHOW

SHOW CHARSET

`SHOW CHARSET` 语句显示当前的字符集。

目前可用的字符集为 `utf8` 和 `utf8mb4` 。默认字符集为 `utf8` 。悦数图数据库 扩展 `uft8` 支持四字节字符，因此 `utf8` 和 `utf8mb4` 是等价的。

语法

```
SHOW CHARSET;
```

示例

```
nebula> SHOW CHARSET;
+---------+-----------------+-------------------+--------+
| Charset | Description     | Default collation | Maxlen |
+---------+-----------------+-------------------+--------+
| "utf8"  | "UTF-8 Unicode" | "utf8_bin"        | 4      |
+---------+-----------------+-------------------+--------+
```

参数	说明
`Charset`	字符集名称。
`Description`	字符集说明。
`Default collation`	默认排序规则。
`Maxlen`	存储一个字符所需的最大字节数。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW COLLATION

`SHOW COLLATION` 语句显示当前的排序规则。

目前可用的排序规则为 `utf8_bin` 和 `utf8mb4_bin` 。

- 当字符集为 `utf8` ，默认排序规则为 `utf8_bin` 。
- 当字符集为 `utf8mb4` ，默认排序规则为 `utf8mb4_bin` 。

语法

```
SHOW COLLATION;
```

示例

```
nebula> SHOW COLLATION;
+------------+---------+
| Collation  | Charset |
+------------+---------+
| "utf8_bin" | "utf8"  |
+------------+---------+
```

参数	说明
`Collation`	排序规则名称。
`Charset`	与排序规则关联的字符集名称。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW CREATE SPACE

`SHOW CREATE SPACE` 语句显示指定图空间的创建语句。

图空间的更多详细信息，请参见 CREATE SPACE。

语法

```
SHOW CREATE SPACE <space_name>;
```

示例

```
nebula> SHOW CREATE SPACE basketballplayer;
+--------------------+-----------------------
```

SHOW CREATE TAG/EDGE

`SHOW CREATE TAG` 语句显示指定 Tag 的基本信息。Tag 的更多详细信息，请参见 CREATE TAG。

`SHOW CREATE EDGE` 语句显示指定 Edge type 的基本信息。Edge type 的更多详细信息，请参见 CREATE EDGE。

语法

```
SHOW CREATE {TAG <tag_name> | EDGE <edge_name>};
```

示例

```
nebula> SHOW CREATE TAG player;
+----------+----------------------------------+
| Tag      | Create Tag                       |
+----------+----------------------------------+
| "player" | "CREATE TAG `player` (           |
|          |  `name` string NULL,             |
|          |  `age` int64 NULL                |
|          | ) ttl_duration = 0, ttl_col = """ |
+----------+----------------------------------+

nebula> SHOW CREATE EDGE follow;
+----------+----------------------------------+
| Edge     | Create Edge                      |
+----------+----------------------------------+
| "follow" | "CREATE EDGE `follow` (          |
|          |  `degree` int64 NULL             |
|          | ) ttl_duration = 0, ttl_col = """ |
+----------+----------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW HOSTS

`SHOW HOSTS` 语句可以显示集群信息，包括端口、状态、leader、分片、版本等信息，或者指定显示 Graph、Storage、Meta 服务主机信息。

语法

```
SHOW HOSTS [GRAPH | STORAGE | META];
```

> **Note**
>
> 对于使用源码安装的悦数图数据库，执行添加了服务名的命令后，输出的信息中不显示版本信息。

示例

```
nebula> SHOW HOSTS;
+-------------+-------+----------+--------------+---------------------------------+------------------------------+---------+
| Host        | Port  | Status   | Leader count | Leader distribution             | Partition distribution       | Version |
+-------------+-------+----------+--------------+---------------------------------+------------------------------+---------+
| "storaged0" | 9779  | "ONLINE" | 8            | "docs:5, basketballplayer:3"    | "docs:5, basketballplayer:3" | "3.5.1" |
| "storaged1" | 9779  | "ONLINE" | 9            | "basketballplayer:4, docs:5"    | "docs:5, basketballplayer:4" | "3.5.1" |
| "storaged2" | 9779  | "ONLINE" | 8            | "basketballplayer:3, docs:5"    | "docs:5, basketballplayer:3" | "3.5.1" |
+-------------+-------+----------+--------------+---------------------------------+------------------------------+---------+

nebula> SHOW HOSTS GRAPH;
+-----------+------+----------+---------+--------------+---------+
| Host      | Port | Status   | Role    | Git Info Sha | Version |
+-----------+------+----------+---------+--------------+---------+
| "graphd"  | 9669 | "ONLINE" | "GRAPH" | "3ba41bd"    | "3.5.1" |
| "graphd1" | 9669 | "ONLINE" | "GRAPH" | "3ba41bd"    | "3.5.1" |
| "graphd2" | 9669 | "ONLINE" | "GRAPH" | "3ba41bd"    | "3.5.1" |
+-----------+------+----------+---------+--------------+---------+

nebula> SHOW HOSTS STORAGE;
+-------------+------+----------+-----------+--------------+---------+
| Host        | Port | Status   | Role      | Git Info Sha | Version |
+-------------+------+----------+-----------+--------------+---------+
| "storaged0" | 9779 | "ONLINE" | "STORAGE" | "3ba41bd"    | "3.5.1" |
| "storaged1" | 9779 | "ONLINE" | "STORAGE" | "3ba41bd"    | "3.5.1" |
| "storaged2" | 9779 | "ONLINE" | "STORAGE" | "3ba41bd"    | "3.5.1" |
+-------------+------+----------+-----------+--------------+---------+

nebula> SHOW HOSTS META;
+----------+------+----------+--------+--------------+---------+
| Host     | Port | Status   | Role   | Git Info Sha | Version |
+----------+------+----------+--------+--------------+---------+
| "metad2" | 9559 | "ONLINE" | "META" | "3ba41bd"    | "3.5.1" |
| "metad0" | 9559 | "ONLINE" | "META" | "3ba41bd"    | "3.5.1" |
| "metad1" | 9559 | "ONLINE" | "META" | "3ba41bd"    | "3.5.1" |
+----------+------+----------+--------+--------------+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW INDEX STATUS

`SHOW INDEX STATUS` 语句显示重建原生索引的作业状态，以便确定重建索引是否成功。

语法

```
SHOW {TAG | EDGE} INDEX STATUS;
```

示例

```
nebula> SHOW TAG INDEX STATUS;
+-----------------------------------+--------------+
| Name                              | Index Status |
+-----------------------------------+--------------+
| "date1_index"                     | "FINISHED"   |
| "basketballplayer_all_tag_indexes" | "FINISHED"   |
| "any_shape_geo_index"             | "FINISHED"   |
+-----------------------------------+--------------+

nebula> SHOW EDGE INDEX STATUS;
+---------------+--------------+
| Name          | Index Status |
+---------------+--------------+
| "follow_index" | "FINISHED"   |
+---------------+--------------+
```

相关文档

- 管理作业
- REBUILD NATIVE INDEX

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW INDEXES

`SHOW INDEXES` 语句可以列出当前图空间内的所有 Tag 和 Edge type（包括属性）的索引。

语法

```
SHOW {TAG | EDGE} INDEXES;
```

示例

```
nebula> SHOW TAG INDEXES;
+------------------+----------+----------+
| Index Name       | By Tag   | Columns  |
+------------------+----------+----------+
| "player_index_0" | "player" | []       |
| "player_index_1" | "player" | ["name"] |
+------------------+----------+----------+

nebula> SHOW EDGE INDEXES;
+----------------+----------+---------+
| Index Name     | By Edge  | Columns |
+----------------+----------+---------+
| "follow_index" | "follow" | []      |
+----------------+----------+---------+
```

版本兼容性

悦数图数据库 2.0.1 中， `SHOW TAG/EDGE INDEXES` 语句仅返回 `Names` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW PARTS

`SHOW PARTS` 语句显示图空间中指定分片或所有分片的信息。

语法

```
SHOW PARTS [<part_id>];
```

示例

```
nebula> SHOW PARTS;
+--------------+--------------------+--------------------+-------+
| Partition ID | Leader             | Peers              | Losts |
+--------------+--------------------+--------------------+-------+
| 1            | "192.168.2.1:9779" | "192.168.2.1:9779" | ""    |
| 2            | "192.168.2.2:9779" | "192.168.2.2:9779" | ""    |
| 3            | "192.168.2.3:9779" | "192.168.2.3:9779" | ""    |
| 4            | "192.168.2.1:9779" | "192.168.2.1:9779" | ""    |
| 5            | "192.168.2.2:9779" | "192.168.2.2:9779" | ""    |
| 6            | "192.168.2.3:9779" | "192.168.2.3:9779" | ""    |
| 7            | "192.168.2.1:9779" | "192.168.2.1:9779" | ""    |
| 8            | "192.168.2.2:9779" | "192.168.2.2:9779" | ""    |
| 9            | "192.168.2.3:9779" | "192.168.2.3:9779" | ""    |
| 10           | "192.168.2.1:9779" | "192.168.2.1:9779" | ""    |
+--------------+--------------------+--------------------+-------+

nebula> SHOW PARTS 1;
+--------------+--------------------+--------------------+-------+
| Partition ID | Leader             | Peers              | Losts |
+--------------+--------------------+--------------------+-------+
| 1            | "192.168.2.1:9779" | "192.168.2.1:9779" | ""    |
+--------------+--------------------+--------------------+-------+
```

返回结果的说明如下：

参数	说明
`Partition ID`	存储分片的 ID
`Leader`	分片对应的 Raft leader 副本的信息，包括 IP 地址与服务端口
`Peers`	分片对应的所有副本（leader 与 follower）的信息，包括 IP 地址与服务端口
`Losts`	分片对应的处于离线状态的副本信息，包括 IP 地址和服务端口

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW ROLES

`SHOW ROLES` 语句显示分配给用户的角色信息。

根据登录的用户角色，返回的结果也有所不同：

- 如果登录的用户角色是 `GOD`，或者有权访问该图空间的 `ADMIN`，则返回该图空间内除 `GOD` 之外的所有用户角色信息。
- 如果登录的用户角色是有权访问该图空间 `DBA` 、 `USER` 或 `GUEST`，则返回自身的角色信息。
- 如果登录的用户角色没有权限访问该图空间，则返回权限错误。

关于角色的详情请参见内置角色权限。

语法

```
SHOW ROLES IN <space_name>;
```

示例

```
nebula> SHOW ROLES in basketballplayer;
+---------+-----------+
| Account | Role Type |
+---------+-----------+
| "user1" | "ADMIN"   |
+---------+-----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW SNAPSHOTS

`SHOW SNAPSHOTS` 语句显示所有快照信息。

快照的使用方式请参见管理快照。

角色要求

只有 `GOD` 角色的用户（即 `root` ）才能执行 `SHOW SNAPSHOTS` 语句。

语法

```
SHOW SNAPSHOTS;
```

示例

```
nebula> SHOW SNAPSHOTS;
+--------------------------------+---------+-------------------------------------------------+
| Name                           | Status  | Hosts                                           |
+--------------------------------+---------+-------------------------------------------------+
| "SNAPSHOT_2020_12_16_11_13_55" | "VALID" | "storaged0:9779, storaged1:9779, storaged2:9779"    |
| "SNAPSHOT_2020_12_16_11_14_10" | "VALID" | "storaged0:9779, storaged1:9779, storaged2:9779"    |
+--------------------------------+---------+-------------------------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW SPACES

`SHOW SPACES` 语句显示现存的图空间。

如何创建图空间，请参见 CREATE SPACE。

语法

```
SHOW SPACES;
```

示例

```
nebula> SHOW SPACES;
+--------------------+
| Name               |
+--------------------+
| "docs"             |
| "basketballplayer" |
+--------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW STATS

`SHOW STATS` 语句显示最近一次 `SUBMIT JOB STATS` 作业收集的图空间统计信息。

图空间统计信息包含：

- 点的总数
- 边的总数
- 每个 Tag 关联的点的总数
- 每个 Edge type 关联的边的总数

Warning

`SHOW STATS` 返回的不是实时数据。因为返回的数据是最近一次 SUBMIT JOB STATS 作业收集的数据。返回的数据可能包含 TTL 过期数据，该过期数据会在下次执行 Compaction 操作时被删除并不纳入统计。

前提条件

在需要查看统计信息的图空间中执行 `SUBMIT JOB STATS` 。详情请参见 SUBMIT JOB STATS。

Caution

`SHOW STATS` 的结果取决于最后一次执行的 `SUBMIT JOB STATS` 。如果发生过新的写入或者更改，必须再次执行 `SUBMIT JOB STATS` ，否则统计数据有错误。

语法

```
SHOW STATS;
```

示例

```
# 选择图空间。
nebula> USE basketballplayer;

# 执行 SUBMIT JOB STATS。
nebula> SUBMIT JOB STATS;
+------------+
| New Job Id |
+------------+
| 98         |
+------------+

# 确认作业执行成功。
nebula> SHOW JOB 98;
+----------------+---------------+------------+----------------------------+----------------------------+-------------+
| Job Id(TaskId) | Command(Dest) | Status     | Start Time                 | Stop Time                  | Error Code  |
+----------------+---------------+------------+----------------------------+----------------------------+-------------+
| 98             | "STATS"       | "FINISHED" | 2021-11-01T09:33:21.000000 | 2021-11-01T09:33:21.000000 | "SUCCEEDED" |
| 0              | "storaged2"   | "FINISHED" | 2021-11-01T09:33:21.000000 | 2021-11-01T09:33:21.000000 | "SUCCEEDED" |
| 1              | "storaged0"   | "FINISHED" | 2021-11-01T09:33:21.000000 | 2021-11-01T09:33:21.000000 | "SUCCEEDED" |
| 2              | "storaged1"   | "FINISHED" | 2021-11-01T09:33:21.000000 | 2021-11-01T09:33:21.000000 | "SUCCEEDED" |
| "Total:3"      | "Succeeded:3" | "Failed:0" | "In Progress:0"            | ""                         | ""          |
+----------------+---------------+------------+----------------------------+----------------------------+-------------+

# 显示图空间统计信息。
nebula> SHOW STATS;
+---------+------------+-------+
| Type    | Name       | Count |
+---------+------------+-------+
| "Tag"   | "player"   | 51    |
| "Tag"   | "team"     | 30    |
| "Edge"  | "follow"   | 81    |
| "Edge"  | "serve"    | 152   |
| "Space" | "vertices" | 81    |
| "Space" | "edges"    | 233   |
+---------+------------+-------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW TAGS/EDGES

`SHOW TAGS` 语句显示当前图空间内的所有 Tag。

`SHOW EDGES` 语句显示当前图空间内的所有 Edge type。

语法

```
SHOW {TAGS | EDGES};
```

示例

```
nebula> SHOW TAGS;
+----------+
| Name     |
+----------+
| "player" |
| "star"   |
| "team"   |
+----------+

nebula> SHOW EDGES;
+----------+
| Name     |
+----------+
| "follow" |
| "serve"  |
+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW USERS

`SHOW USERS` 语句显示用户信息。

角色要求

只有 `GOD` 角色的用户（即 `root`）才能执行 `SHOW USERS` 语句。

语法

```
SHOW USERS;
```

示例

```
nebula> SHOW USERS;
+---------+-----------------+
| Account | IP Whitelist    |
+---------+-----------------+
| "root"  | ""              |
| "user1" | ""              |
| "user2" | "192.168.10.10" |
+---------+-----------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW SESSIONS

登录悦数图数据库数据库时，会创建对应会话，用户可以查询会话信息。

注意事项

- 执行 `exit` 退出登录时，客户端会调用 API `release`，释放会话并清除会话信息。如果没有正常退出，且没有在配置文件 nebula-graphd.conf 设置空闲会话超时时间（`session_idle_timeout_secs`），会话不会自动释放。对于未自动释放的会话，需要手动删除指定会话，详情参见终止会话。
- `SHOW SESSIONS` 查询所有 Graph 服务上的会话信息。
- `SHOW LOCAL SESSIONS` 从当前连接的 Graph 服务获取会话信息，不会查询其他 Graph 服务上的会话信息。
- `SHOW SESSION <Session_Id>` 查询指定 Session ID 的会话信息。

语法

```
SHOW [LOCAL] SESSIONS;
SHOW SESSION <Session_Id>;
```

示例

```
nebula> SHOW SESSIONS;
+-----------------+----------+-------------------+-------------------------+-------------------------+-----------------+----------+-------------------+
| SessionId       | UserName | SpaceName         | CreateTime              | UpdateTime              | GraphAddr       | Timezone | ClientIp          |
+-----------------+----------+-------------------+-------------------------+-------------------------+-----------------+----------+-------------------+
| 1651220858102296 | "root"   | "basketballplayer" | 2022-04-29T08:27:38.102296 | 2022-04-29T08:50:46.282921 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
| 1651199330300991 | "root"   | "basketballplayer" | 2022-04-29T02:28:50.300991 | 2022-04-29T08:16:28.339038 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
| 1651112899847744 | "root"   | "basketballplayer" | 2022-04-28T02:28:19.847744 | 2022-04-28T08:17:44.470210 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
| 1651041092662100 | "root"   | "basketballplayer" | 2022-04-27T06:31:32.662100 | 2022-04-27T07:01:25.200978 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
| 1650959429593975 | "root"   | "basketballplayer" | 2022-04-26T07:50:29.593975 | 2022-04-26T07:51:47.184810 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
| 1650958897679595 | "root"   | ""                 | 2022-04-26T07:41:37.679595 | 2022-04-26T07:41:37.683802 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
+-----------------+----------+-------------------+-------------------------+-------------------------+-----------------+----------+-------------------+

nebula> SHOW SESSION 1635254859271703;
+-----------------+----------+-------------------+-------------------------+-------------------------+-----------------+----------+-------------------+
| SessionId       | UserName | SpaceName         | CreateTime              | UpdateTime              | GraphAddr       | Timezone | ClientIp          |
+-----------------+----------+-------------------+-------------------------+-------------------------+-----------------+----------+-------------------+
| 1651220858102296 | "root"   | "basketballplayer" | 2022-04-29T08:27:38.102296 | 2022-04-29T08:50:54.254384 | "127.0.0.1:9669" | 0        | "127.0.0.1" |
+-----------------+----------+-------------------+-------------------------+-------------------------+-----------------+----------+-------------------+
```

参数	说明
`SessionId`	会话 ID，唯一标识一个会话。
`UserName`	会话的登录用户名称。
`SpaceName`	用户当前所使用的图空间。刚登录时为空（`""`）。
`CreateTime`	会话的创建时间，即用户认证登录的时间。时区为配置文件中 `timezone_name` 指定的时区。
`UpdateTime`	用户有执行操作时，会更新此时间。时区为配置文件中 `timezone_name` 指定的时区。
`GraphAddr`	会话的 Graph 服务地址和端口。
`Timezone`	保留参数，暂无意义。
`ClientIp`	会话的客户端 IP 地址。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW QUERIES

`SHOW QUERIES` 语句可以查看当前 Session 中正在执行的查询请求信息。

> **Note**

如果需要终止查询，请参见终止查询。

注意事项

- `SHOW LOCAL QUERIES` 从本地缓存获取当前 Session 中查询的状态，几乎没有延迟。
- `SHOW QUERIES` 从 Meta 服务获取所有 Session 中的查询信息。这些信息会根据参数 `session_reclaim_interval_secs` 定义的周期同步到 Meta 服务，因此在客户端获取到的信息可能属于上个同步周期。

语法

```
SHOW [LOCAL] QUERIES;
```

示例

```
nebula> SHOW LOCAL QUERIES;
+------------------+----------------+--------+----------------------+----------------------------+----------------+-----------+------------------------+
| SessionID        | ExecutionPlanID | User   | Host                 | StartTime                  | DurationInUSec | Status    | Query                  |
+------------------+----------------+--------+----------------------+----------------------------+----------------+-----------+------------------------+
| 1625463842921750 | 46             | "root" | ""192.168.x.x":9669" | 2021-07-05T05:44:19.502903 | 0              | "RUNNING" | "SHOW LOCAL QUERIES;" |
+------------------+----------------+--------+----------------------+----------------------------+----------------+-----------+------------------------+

nebula> SHOW QUERIES;
+------------------+----------------+---------+----------------------+----------------------------+----------------+-----------+-----------------------------------------------------------+
| SessionID        | ExecutionPlanID | User    | Host                 | StartTime                  | DurationInUSec | Status    | Query                                                     |
+------------------+----------------+---------+----------------------+----------------------------+----------------+-----------+-----------------------------------------------------------+
| 1625456037718757 | 54             | "user1" | ""192.168.x.x":9669" | 2021-07-05T05:51:08.691318 | 1504502        | "RUNNING" | "MATCH p=(v:player)-[*1..4]-(v2) RETURN v2 AS Friends;" |
+------------------+----------------+---------+----------------------+----------------------------+----------------+-----------+-----------------------------------------------------------+

# 返回耗时 TOP 10 的查询。
nebula> SHOW QUERIES | ORDER BY $-.DurationInUSec DESC | LIMIT 10;
+------------------+----------------+---------+----------------------+----------------------------+----------------+-----------+-----------------------------------------------------------+
| SessionID        | ExecutionPlanID | User    | Host                 | StartTime                  | DurationInUSec | Status    | Query                                                     |
+------------------+----------------+---------+----------------------+----------------------------+----------------+-----------+-----------------------------------------------------------+
| 1625471375320831 | 98             | "user2" | ""192.168.x.x":9669" | 2021-07-05T07:50:24.461779 | 2608176        | "RUNNING" | "MATCH (v:player)-[*1..4]-(v2) RETURN v2 AS Friends;" |
| 1625456037718757 | 99             | "user1" | ""192.168.x.x":9669" | 2021-07-05T07:50:24.910616 | 2159333        | "RUNNING" | "MATCH (v:player)-[*1..4]-(v2) RETURN v2 AS Friends;" |
+------------------+----------------+---------+----------------------+----------------------------+----------------+-----------+-----------------------------------------------------------+
```

参数说明如下。

参数	说明
`SessionID`	会话 ID。
`ExecutionPlanID`	执行计划 ID。
`User`	执行查询的用户名。
`Host`	用户连接的服务器地址和端口。
`StartTime`	执行查询的开始时间。
`DurationInUSec`	执行查询的持续时长。单位：微秒。
`Status`	查询的当前状态。
`Query`	查询语句。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

SHOW META LEADER

`SHOW META LEADER` 语句显示当前 Meta 集群的 leader 信息。

关于 Meta 服务的详细说明请参见 Meta 服务。

语法

```
SHOW META LEADER;
```

示例

```
nebula> SHOW META LEADER;
+-----------------+--------------------------+
| Meta Leader     | secs from last heart beat |
+-----------------+--------------------------+
| "127.0.0.1:9559" | 3                        |
+-----------------+--------------------------+
```

参数	说明
`Meta Leader`	Meta 集群的 leader 信息，包括 leader 所在服务器的 IP 地址和端口。
`secs from last heart beat`	距离上次心跳的时间间隔。单位：秒。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7 子句和选项

5.7.1 GROUP BY

`GROUP BY` 子句可以用于聚合数据。

openCypher 兼容性

本文操作仅适用于原生 nGQL。

用户也可以使用 openCypher 方式的 count() 函数聚合数据。

```
nebula>  MATCH (v:player)<-[:follow]-(:player) RETURN v.player.name AS Name, count(*) as cnt ORDER BY cnt DESC;
+---------------------+-----+
| Name                | cnt |
+---------------------+-----+
| "Tim Duncan"        | 10  |
| "LeBron James"      | 6   |
| "Tony Parker"       | 5   |
| "Chris Paul"        | 4   |
| "Manu Ginobili"     | 4   |
+---------------------+-----+
...
```

语法

`GROUP BY` 子句可以聚合相同值的行，然后进行计数、排序和计算等操作。

`GROUP BY` 子句可以在管道符（|）之后和 `YIELD` 子句之前使用。

```
| GROUP BY <var> YIELD <var>, <aggregation_function(var)>
```

`aggregation_function()` 函数支持 `avg()` 、 `sum()` 、 `max()` 、 `min()` 、 `count()` 、 `collect()` 、 `std()` 。

示例

```
# 查找所有连接到 player100 的点，并根据他们的姓名进行分组，返回姓名的出现次数。
nebula> GO FROM "player100" OVER follow BIDIRECT \
        YIELD properties($$).name as Name \
        | GROUP BY $-.Name \
        YIELD $-.Name as Player, count(*) AS Name_Count;
+--------------------+-----------+
| Player             | Name_Count |
+--------------------+-----------+
| "Shaquille O'Neal"  | 1         |
| "Tiago Splitter"    | 1         |
| "Manu Ginobili"     | 2         |
| "Boris Diaw"        | 1         |
| "LaMarcus Aldridge" | 1         |
| "Tony Parker"       | 2         |
| "Marco Belinelli"   | 1         |
| "Dejounte Murray"   | 1         |
| "Danny Green"       | 1         |
| "Aron Baynes"       | 1         |
+--------------------+-----------+
```

```
# 查找所有连接到 player100 的点，并根据起始点进行分组，返回 degree 的总和。
nebula> GO FROM "player100" OVER follow \
        YIELD src(edge) AS player, properties(edge).degree AS degree \
        | GROUP BY $-.player \
        YIELD sum($-.degree);
+---------------+
| sum($-.degree) |
+---------------+
| 190           |
+---------------+
```

`sum()` 函数详情请参见内置数学函数。

Copyright © Yueshu Graph Technology - All rights reserved.

隐式分组

在上述 nGQL 语句中明确写出 `GROUP BY` 并起到分组字段作用的用法称为 `GROUP BY` 显示用法；而在 openCypher 语句中 `GROUP BY` 的用法是隐式的，即在语句中不用写出 `GROUP BY` 也可起到分组字段的作用。nGQL 语句中显示地 `GROUP BY` 用法与 openCypher 语句中的隐式地 `GROUP BY` 用法相同，并且 nGQL 语句兼容 openCypher 的用法，即也支持隐式地使用 `GROUP BY` 。有关 `GROUP BY` 的隐式用法，请参见how-to-make-group-by-in-a-cypher-query。

例如：查询 34 岁以上的球员中完全重叠服役的区间。

```
nebula> LOOKUP ON player WHERE player.age > 34 YIELD id(vertex) AS v | \
        GO FROM $-.v OVER serve YIELD serve.start_year AS start_year, serve.end_year AS end_year | \
        YIELD $-.start_year, $-.end_year, count(*) AS count | \
        ORDER BY $-.count DESC | LIMIT 5;
+---------------+-------------+-------+
| $-.start_year | $-.end_year | count |
+---------------+-------------+-------+
| 2018          | 2019        | 3     |
| 2007          | 2012        | 2     |
| 1998          | 2004        | 2     |
| 2017          | 2018        | 2     |
| 2010          | 2011        | 2     |
+---------------+-------------+-------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.2 LIMIT

`LIMIT` 子句限制输出结果的行数。`LIMIT` 在原生 nGQL 语句和 openCypher 兼容语句中的用法有所不同。

- 在原生 nGQL 语句中，一般需要在 `LIMIT` 子句前使用管道符，可以直接在 LIMIT 语句后设置或者省略偏移量参数。
- 在 openCypher 兼容语句中，不允许在 `LIMIT` 子句前使用管道符，可以使用 `SKIP` 指明偏移量。

> **Note**
>
> 在原生 nGQL 或 openCypher 方式中使用 `LIMIT` 时，使用 `ORDER BY` 子句限制输出顺序非常重要，否则会输出一个不可预知的子集。

原生 **nGQL** 语句中的 **LIMIT**

在原生 nGQL 中，`LIMIT` 有通用语法和 `GO` 语句中的专属语法。

原生 **NGQL** 中的通用 **LIMIT** 语法

原生 nGQL 中的通用 `LIMIT` 语法与 `SQL` 中的 `LIMIT` 原理相同。`LIMIT` 子句接收一个或两个参数，参数的值必须是非负整数，且必须用在管道符之后。语法和说明如下：

```
... | LIMIT [<offset>,] <number_rows>;
```

参数	说明
`offset`	偏移量，即定义从哪一行开始返回。索引从 `0` 开始。默认值为 `0`，表示从第一行开始返回。
`number_rows`	返回的总行数。

示例：

```
# 从结果中返回最前面的 3 行数据。
nebula> LOOKUP ON player YIELD id(vertex)|\
        LIMIT 3;
+-------------+
| id(VERTEX)  |
+-------------+
| "player100" |
| "player101" |
| "player102" |
+-------------+

# 从排序后结果中返回第 2 行开始的 3 行数据。
nebula> GO FROM "player100" OVER follow REVERSELY \
        YIELD properties($$).name AS Friend, properties($$).age AS Age \
        | ORDER BY $-.Age, $-.Friend \
        | LIMIT 1, 3;
+-------------------+-----+
| Friend            | Age |
+-------------------+-----+
| "Danny Green"     | 31  |
| "Aron Baynes"     | 32  |
| "Marco Belinelli" | 32  |
+-------------------+-----+
```

GO 语句中的 **LIMIT**

`GO` 语句中的 `LIMIT` 除了支持原生 nGQL 中的通用语法外，还支持根据边限制输出结果数量。

语法：

```
<go_statement> LIMIT <limit_list>;
```


Copyright © Yueshu Graph Technology - All rights reserved.

`limit_list` 是一个列表，列表中的元素必须为自然数，且元素数量必须与 `GO` 语句中的 `STEPS` 的最大数相同。下文以 `GO 1 TO 3 STEPS FROM "A" OVER * LIMIT <limit_list>` 为例详细介绍 `LIMIT` 的这种用法。

- 列表 `limit_list` 必须包含 3 个自然数元素，例如 `GO 1 TO 3 STEPS FROM "A" OVER * LIMIT [1,2,4]` 。
- `LIMIT [1,2,4]` 中的 `1` 表示系统在第一步时自动选择 1 条边继续遍历， `2` 表示在第二步时选择 2 条边继续遍历， `4` 表示在第三步时选择 4 条边继续遍历。
- 因为 `GO 1 TO 3 STEPS` 表示返回第一到第三步的所有遍历结果，因此下图中所有红色边和它们的原点与目的点都会被这条 `GO` 语句匹配上，而黄色边表示 `GO` 语句遍历时没有选择的路径。如果不是 `GO 1 TO 3 STEPS` 而是 `GO 3 STEPS` ，则只会匹配上第三步的红色边和它们两端的点。

在 basketballplayer 数据集中的执行示例如下：

```
nebula> GO 3 STEPS FROM "player100" \
        OVER * \
        YIELD properties($$).name AS NAME, properties($$).age AS Age \
        LIMIT [3,3,3];
+----------------+----------+
| NAME           | Age      |
+----------------+----------+
| "Tony Parker"  | 36       |
| "Manu Ginobili" | 41       |
| "Spurs"        | __NULL__ |
+----------------+----------+

nebula> GO 3 STEPS FROM "player102" OVER * BIDIRECT\
        YIELD dst(edge) \
        LIMIT [rand32(5),rand32(5),rand32(5)];
+------------+
| dst(EDGE)  |
+------------+
| "player100" |
| "player100" |
+------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

openCypher 兼容语句中的 LIMIT

在 `MATCH` 等 openCypher 兼容语句中使用 LIMIT 不需要加管道符。语法和说明如下：

```
... [SKIP <offset>] [LIMIT <number_rows>];
```

参数	说明
`offset`	偏移量，即定义从哪一行开始返回。索引从 `0` 开始。默认值为 `0`，表示从第一行开始返回。
`number_rows`	返回的总行数量。

`offset` 和 `number_rows` 可以使用表达式，但是表达式的结果必须是非负整数。

Note

两个整数组成的分数表达式会自动向下取整。例如 `8/6` 向下取整为 1。

单独使用 **LIMIT**

`LIMIT` 可以单独使用，返回指定数量的结果。

```
nebula> MATCH (v:player) RETURN v.player.name AS Name, v.player.age AS Age \
        ORDER BY Age LIMIT 5;
+-------------------------+-----+
| Name                    | Age |
+-------------------------+-----+
| "Luka Doncic"           | 20  |
| "Ben Simmons"           | 22  |
| "Kristaps Porzingis"    | 23  |
| "Giannis Antetokounmpo" | 24  |
| "Kyle Anderson"         | 25  |
+-------------------------+-----+
```

单独使用 **SKIP**

`SKIP` 可以单独使用，用于设置偏移量，返回指定位置之后的数据。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) --> (v2) \
        RETURN v2.player.name AS Name, v2.player.age AS Age \
        ORDER BY Age DESC SKIP 1;
+-----------------+-----+
| Name            | Age |
+-----------------+-----+
| "Manu Ginobili" | 41  |
| "Tony Parker"   | 36  |
+-----------------+-----+

nebula> MATCH (v:player{name:"Tim Duncan"}) --> (v2) \
        RETURN v2.player.name AS Name, v2.player.age AS Age \
        ORDER BY Age DESC SKIP 1+1;
+---------------+-----+
| Name          | Age |
+---------------+-----+
| "Tony Parker" | 36  |
+---------------+-----+
```

同时使用 **SKIP** 与 **LIMIT**

同时使用 `SKIP` 与 `LIMIT` 可以返回从指定位置开始的指定数量的数据。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) --> (v2) \
        RETURN v2.player.name AS Name, v2.player.age AS Age \
        ORDER BY Age DESC SKIP 1 LIMIT 1;
+-----------------+-----+
| Name            | Age |
+-----------------+-----+
| "Manu Ginobili" | 41  |
+-----------------+-----+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.3 SAMPLE

`SAMPLE` 子句用于在结果集中均匀取样并返回指定数量的数据。

`SAMPLE` 仅能在 `GO` 语句中使用，语法如下：

```
<go_statement> SAMPLE <sample_list>;
```

`sample_list` 是一个列表，列表中的元素必须为自然数，且元素数量必须与 `GO` 语句中的 `STEPS` 的最大数相同。下文以 `GO 1 TO 3 STEPS FROM "A" OVER * SAMPLE <sample_list>` 为例详细介绍 `SAMPLE` 的用法。

- 列表 `sample_list` 必须包含 3 个自然数元素，例如 `GO 1 TO 3 STEPS FROM "A" OVER * SAMPLE [1,2,4]` 。
- `SAMPLE [1,2,4]` 中的 `1` 表示系统在第一步时自动选择 1 条边继续遍历， `2` 表示在第二步时选择 2 条边继续遍历， `4` 表示在第三步时选择 4 条边继续遍历。如果某一步没有匹配的边或者匹配到的边数量小于指定数量，则按实际数量返回。
- 因为 `GO 1 TO 3 STEPS` 表示返回第一到第三步的所有遍历结果，因此下图中所有红色边和它们的原点与目的点都会被这条 `GO` 语句匹配上，而黄色边表示 `GO` 语句遍历时没有选择的路径。如果不是 `GO 1 TO 3 STEPS` 而是 `GO 3 STEPS` ，则只会匹配上第三步的红色边和它们两端的点。

在 basketballplayer 数据集中的执行示例如下：

```
nebula> GO 3 STEPS FROM "player100" \
        OVER * \
        YIELD properties($$).name AS NAME, properties($$).age AS Age \
        SAMPLE [1,2,3];
+----------------+----------+
| NAME           | Age      |
+----------------+----------+
| "Tony Parker"  | 36       |
| "Manu Ginobili" | 41       |
| "Spurs"        | __NULL__ |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+----------------+----------+

nebula> GO 1 TO 3 STEPS FROM "player100" \
        OVER * \
        YIELD properties($$).name AS NAME, properties($$).age AS Age \
        SAMPLE [2,2,2];
+----------------+----------+
| NAME           | Age      |
+----------------+----------+
| "Manu Ginobili" | 41       |
| "Spurs"        | __NULL__ |
| "Tim Duncan"   | 42       |
| "Spurs"        | __NULL__ |
| "Manu Ginobili" | 41       |
| "Spurs"        | __NULL__ |
+----------------+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.4 ORDER BY

`ORDER BY` 子句指定输出结果的排序规则。

- 在原生 nGQL 中，必须在 `YIELD` 子句之后使用管道符（|）和 `ORDER BY` 子句。
- 在 openCypher 方式中，不允许使用管道符。在 `RETURN` 子句之后使用 `ORDER BY` 子句。

排序规则分为如下两种：

- `ASC` （默认）：升序。
- `DESC`：降序。

原生 nGQL 语法

```
<YIELD clause>
| ORDER BY <expression> [ASC | DESC] [, <expression> [ASC | DESC] ...];
```

Compatibility

原生 nGQL 语法中，`ORDER BY` 命令后必须使用引用符 `$-.`。但在 2.5.0 之前的版本中不需要。

示例

```
nebula> FETCH PROP ON player "player100", "player101", "player102", "player103" \
        YIELD properties(vertex).age AS age, properties(vertex).name AS name \
        | ORDER BY $-.age ASC, $-.name DESC;
+-----+--------------------+
| age | name               |
+-----+--------------------+
| 32  | "Rudy Gay"         |
| 33  | "LaMarcus Aldridge" |
| 36  | "Tony Parker"      |
| 42  | "Tim Duncan"       |
+-----+--------------------+

nebula> $var = GO FROM "player100" OVER follow \
        YIELD dst(edge) AS dst; \
        ORDER BY $var.dst DESC;
+-------------+
| dst         |
+-------------+
| "player125" |
| "player101" |
+-------------+
```

OpenCypher 方式语法

```
<RETURN clause>
ORDER BY <expression> [ASC | DESC] [, <expression> [ASC | DESC] ...];
```

示例

```
nebula> MATCH (v:player) RETURN v.player.name AS Name, v.player.age AS Age  \
        ORDER BY Name DESC;
+-----------------+-----+
| Name            | Age |
+-----------------+-----+
| "Yao Ming"      | 38  |
| "Vince Carter"  | 42  |
| "Tracy McGrady" | 39  |
| "Tony Parker"   | 36  |
| "Tim Duncan"    | 42  |
+-----------------+-----+
...

# 首先以年龄排序，如果年龄相同，再以姓名排序。
nebula> MATCH (v:player) RETURN v.player.age AS Age, v.player.name AS Name  \
        ORDER BY Age DESC, Name ASC;
+-----+-------------------+
| Age | Name              |
+-----+-------------------+
| 47  | "Shaquille O'Neal" |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 46  | "Grant Hill"     |
| 45  | "Jason Kidd"     |
| 45  | "Steve Nash"     |
+-----+------------------+
...
```

NULL 值的排序

升序排列时，会在输出的最后列出 NULL 值，降序排列时，会在输出的开头列出 NULL 值。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) --> (v2) \
        RETURN v2.player.name AS Name, v2.player.age AS Age  \
        ORDER BY Age;
+----------------+----------+
| Name           | Age      |
+----------------+----------+
| "Tony Parker"  | 36       |
| "Manu Ginobili" | 41       |
| __NULL__       | __NULL__ |
+----------------+----------+

nebula> MATCH (v:player{name:"Tim Duncan"}) --> (v2) \
        RETURN v2.player.name AS Name, v2.player.age AS Age  \
        ORDER BY Age DESC;
+----------------+----------+
| Name           | Age      |
+----------------+----------+
| __NULL__       | __NULL__ |
| "Manu Ginobili" | 41       |
| "Tony Parker"  | 36       |
+----------------+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.5 RETURN

`RETURN` 子句定义了 nGQL 查询的输出结果。如果需要返回多个字段，用英文逗号（,）分隔。

`RETURN` 可以引导子句或语句：

- `RETURN` 子句可以用于 nGQL 中的 openCypher 方式语句中，例如 `MATCH` 或 `UNWIND` 。
- `RETURN` 可以单独使用，输出表达式的结果。

openCypher 兼容性

本文操作仅适用于 nGQL 中的 openCypher 方式。关于原生 nGQL 如何定义输出结果，请参见 `YIELD` 。

`RETURN` 不支持如下 openCypher 功能：

- 使用不在英文字母表中的字符作为变量名。例如：

```
MATCH (`点 1`:player) \
RETURN `点 1`;
```

- 设置一个模式，并返回该模式匹配的所有元素。例如：

```
MATCH (v:player) \
RETURN (v)-[e]->(v2);
```

历史版本兼容性

- 在 nGQL 1.x 中， `RETURN` 适用于原生 nGQL，语法为 `RETURN <var_ref> IF <var_ref> IS NOT NULL` 。
- 从 nGQL 2.0 开始， `RETURN` 不适用于原生 nGQL。

Map 顺序说明

`RETURN` 返回 Map 时，Key 的顺序是未定义的。

```
nebula> RETURN {age: 32, name: "Marco Belinelli"};
+------------------------------------+
| {age:32,name:"Marco Belinelli"}    |
+------------------------------------+
| {age: 32, name: "Marco Belinelli"} |
+------------------------------------+

nebula> RETURN {zage: 32, name: "Marco Belinelli"};
+------------------------------------+
| {zage:32,name:"Marco Belinelli"}   |
+------------------------------------+
| {name: "Marco Belinelli", zage: 32} |
+------------------------------------+
```

返回点或边

使用 `RETURN {<vertex_name> | <edge_name>}` 返回点或边的所有信息。

```
// 返回点
nebula> MATCH (v:player) \
        RETURN v;
+------------------------------------------------------------+
| v                                                          |
+------------------------------------------------------------+
| ("player104" :player{age: 32, name: "Marco Belinelli"})    |
| ("player107" :player{age: 32, name: "Aron Baynes"})        |
| ("player116" :player{age: 34, name: "LeBron James"})       |
| ("player120" :player{age: 29, name: "James Harden"})       |
| ("player125" :player{age: 41, name: "Manu Ginobili"})      |
+------------------------------------------------------------+
...
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
// 返回边
nebula> MATCH (v:player)-[e]->() \
        RETURN e;
+------------------------------------------------------------------------------+
| e                                                                            |
+------------------------------------------------------------------------------+
| [:follow "player104"->"player100" @0 {degree: 55}]                           |
| [:follow "player104"->"player101" @0 {degree: 50}]                           |
| [:follow "player104"->"player105" @0 {degree: 60}]                           |
| [:serve "player104"->"team200" @0 {end_year: 2009, start_year: 2007}]        |
| [:serve "player104"->"team208" @0 {end_year: 2016, start_year: 2015}]        |
+------------------------------------------------------------------------------+
...
```

返回点 ID

使用 `id()` 函数返回点 ID。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) \
        RETURN id(v);
+------------+
| id(v)      |
+------------+
| "player100" |
+------------+
```

返回 Tag

使用 `labels()` 函数返回点上的 Tag 列表。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) \
        RETURN labels(v);
+------------+
| labels(v)  |
+------------+
| ["player"] |
+------------+
```

返回列表 `labels(v)` 中的第 N 个元素，可以使用 `labels(v)[n-1]` 。例如下面示例使用 `labels(v)[0]` 检索第一个元素。

```
nebula> MATCH (v:player{name:"Tim Duncan"}) \
        RETURN labels(v)[0];
+--------------+
| labels(v)[0] |
+--------------+
| "player"     |
+--------------+
```

返回属性

使用语法 `<vertex_name>.<tag_name>.<property_name>` 返回点的属性；使用语法 `<edge_name>.<property_name>` 返回边的属性。

```
// 返回点的属性
nebula> MATCH (v:player) \
        RETURN v.player.name, v.player.age \
        LIMIT 3;
+------------------+-------------+
| v.player.name    | v.player.age |
+------------------+-------------+
| "Danny Green"    | 31          |
| "Tiago Splitter" | 34          |
| "David West"     | 38          |
+------------------+-------------+
```

```
// 返回边的属性
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->() \
        RETURN e.start_year, e.degree \
+--------------+----------+
| e.start_year | e.degree |
+--------------+----------+
| __NULL__     | 95       |
| __NULL__     | 95       |
| 1997         | __NULL__ |
+--------------+----------+
```

使用 `properties()` 函数返回点或边的所有属性。

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[]->(v2) \
        RETURN properties(v2);
+---------------------------------+
| properties(v2)                  |
+---------------------------------+
| {name: "Spurs"}                 |
| {age: 36, name: "Tony Parker"}  |
| {age: 41, name: "Manu Ginobili"} |
+---------------------------------+
```

返回 **Edge type**

使用 `type()` 函数返回匹配的 Edge type。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e]->() \
        RETURN DISTINCT type(e);
+----------+
| type(e)  |
+----------+
| "serve"  |
| "follow" |
+----------+
```

返回路径

使用 `RETURN <path_name>` 返回匹配路径的所有信息。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[*3]->() \
        RETURN p;
+-------------------------------
```

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[*..2]->(v2) \
        RETURN p AS Paths, length(p) AS Length;
+-----------------------
```

```
| "Manu Ginobili" | "follow" | 41           |
| __NULL__        | "serve"  | __NULL__     |
| "Tony Parker"   | "follow" | 36           |
+-----------------+----------+--------------+
```

返回表达式结果

`RETURN` 语句可以返回字面量、函数或谓词等表达式的结果。

```
nebula> MATCH (v:player{name:"Tony Parker"})-->(v2:player) \
        RETURN DISTINCT v2.player.name, "Hello"+" graphs!", v2.player.age > 35;
+---------------------+---------------------+--------------------+
| v2.player.name      | ("Hello"+" graphs!") | (v2.player.age>35) |
+---------------------+---------------------+--------------------+
| "LaMarcus Aldridge" | "Hello graphs!"     | false              |
| "Tim Duncan"        | "Hello graphs!"     | true               |
| "Manu Ginobili"     | "Hello graphs!"     | true               |
+---------------------+---------------------+--------------------+

nebula> RETURN 1+1;
+-------+
| (1+1) |
+-------+
| 2     |
+-------+

nebula> RETURN 1- -1;
+----------+
| (1--(1)) |
+----------+
| 2        |
+----------+

nebula> RETURN 3 > 1;
+-------+
| (3>1) |
+-------+
| true  |
+-------+

nebula> RETURN 1+1, rand32(1, 5);
+-------+-------------+
| (1+1) | rand32(1,5) |
+-------+-------------+
| 2     | 1           |
+-------+-------------+
```

返回唯一字段

使用 `DISTINCT` 可以删除结果集中的重复字段。

```
# 未使用 DISTINCT。
nebula> MATCH (v:player{name:"Tony Parker"})--(v2:player) \
        RETURN v2.player.name, v2.player.age;
+---------------------+--------+
| v2.name             | v2.age |
+---------------------+--------+
| "Tim Duncan"        | 42     |
| "LaMarcus Aldridge" | 33     |
| "Marco Belinelli"   | 32     |
| "Boris Diaw"        | 36     |
| "Dejounte Murray"   | 29     |
| "Tim Duncan"        | 42     |
| "LaMarcus Aldridge" | 33     |
| "Manu Ginobili"     | 41     |
+---------------------+--------+

# 使用 DISTINCT。
nebula> MATCH (v:player{name:"Tony Parker"})--(v2:player) \
        RETURN DISTINCT v2.player.name, v2.player.age;
+---------------------+--------+
| v2.name             | v2.age |
+---------------------+--------+
| "Tim Duncan"        | 42     |
| "LaMarcus Aldridge" | 33     |
| "Marco Belinelli"   | 32     |
| "Boris Diaw"        | 36     |
| "Dejounte Murray"   | 29     |
| "Manu Ginobili"     | 41     |
+---------------------+--------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.6 TTL

TTL（Time To Live）指定属性的存活时间，超时后，该属性就会过期。

openCypher 兼容性

本文操作仅适用于原生 nGQL。

注意事项

- 不能修改带有 TTL 选项的属性的 Schema。
- TTL 和 INDEX 共存问题：
- 如果一个 Tag/Edge type 的其中一属性已有 INDEX，则不能为其设置 TTL，也不能为该 Tag 的其他属性设置 TTL。
- 如果已有 TTL，可以再添加 INDEX。

TTL 选项

nGQL 支持的 TTL 选项如下。

选项	说明
`ttl_col`	指定要设置存活时间的属性。属性的数据类型必须是 `int` 或者 `timestamp` 。
`ttl_duration`	指定时间戳差值，默认单位：秒。时间戳差值必须为 64 位非负整数。属性值和时间戳差值之和如果小于当前时间戳，属性就会过期。如果 `ttl_duration` 为 `0` ，属性永不过期。 可在配置文件 `nebula-storaged.conf` （默认路径 `/usr/local/nightly/etc/` ）中设置 `ttl_use_ms` 为 `true` 将默认单位设为毫秒。

Caution

- 在设置 `ttl_use_ms` 为 `true` 前，请确保没有为属性设置 TTL，否则会因为过期时间缩短，导致数据被错误地删除。
- 在设置 `ttl_use_ms` 为 `true` 后，即设置 `ttl_duration` 的默认单位为毫秒后， `ttl_col` 的默认单位仍然为秒，它的数据类型必须是 `int` ，并且需要手动转换属性值为毫秒。例如设置 `ttl_col` 为 `a` ，则需要将 `a` 的值转换为毫秒，如当 `a` 的值为 `now()` ，则需要将 `a` 的值设置为 `now() * 1000` 。

属性过期

Caution

- 当为一个 Tag 或 Edge type 的属性设置 TTL 并该属性的值为 `NULL` 时，TTL 功能不会生效，即该属性永不过期。
- 如果为一个 Tag 或 Edge type 新增默认值为 `now()` 的属性并且该属性设置了 TTL，该 Tag 或 Edge type 相关的历史数据不会过期，因为历史数据的该属性值为当前时间戳。

点属性过期

点属性过期有如下影响：

- 如果一个点仅有一个 Tag，点上的一个属性过期，点也会过期。
- 如果一个点有多个 Tag，点上的一个属性过期，和该属性相同 Tag 的其他属性也会过期，但是点不会过期，点上其他 Tag 的属性保持不变。

边属性过期

因为一条边仅有一个 Edge type，边上的一个属性过期，边也会过期。

Copyright © Yueshu Graph Technology - All rights reserved.

过期处理

属性过期后，对应的过期数据仍然存储在硬盘上，但是查询时会过滤过期数据。

悦数图数据库自动删除过期数据后，会在下一次 Compaction 过程中回收硬盘空间。

> **Note**
>
> 如果关闭 TTL 选项，上一次 Compaction 之后的过期数据将可以被查询到。

使用 **TTL** 选项

TAG 或 **EDGE TYPE** 已存在

如果 Tag 和 Edge type 已经创建，请使用 `ALTER` 语句更新 Tag 或 Edge type。

```
# 创建 Tag。
nebula> CREATE TAG IF NOT EXISTS t1 (a timestamp);

# ALTER 修改 Tag，添加 TTL 选项。
nebula> ALTER TAG t1 TTL_COL = "a", TTL_DURATION = 5;

# 插入点，插入后 5 秒过期。
nebula> INSERT VERTEX t1(a) VALUES "101":(now());
```

TAG 或 **EDGE TYPE** 不存在

创建 Tag 或 Edge type 时可以同时设置 TTL 选项。详情请参见 CREATE TAG 和 CREATE EDGE。

```
# 创建 Tag 并设置 TTL 选项。
nebula> CREATE TAG IF NOT EXISTS t2(a int, b int, c string) TTL_DURATION= 100, TTL_COL = "a";

# 插入点。过期时间戳为 1648197238（1648197138 + 100）。
nebula> INSERT VERTEX t2(a, b, c) VALUES "102":(1648197138, 30, "Hello");
```

删除存活时间

删除存活时间可以使用如下几种方法：

- 删除设置存活时间的属性。

  ```
  nebula> ALTER TAG t1 DROP (a);
  ```

- 设置 `ttl_col` 为空字符串。

  ```
  nebula> ALTER TAG t1 TTL_COL = "";
  ```

- 设置 `ttl_duration` 为 `0`。本操作可以保留 TTL 选项，属性永不过期，且属性的 Schema 无法修改。

  ```
  nebula> ALTER TAG t1 TTL_DURATION = 0;
  ```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.7 WHERE

`WHERE` 子句可以根据条件过滤输出结果。

`WHERE` 子句通常用于如下查询：

- 原生 nGQL，例如 `GO` 和 `LOOKUP` 语句。
- openCypher 方式，例如 `MATCH` 和 `WITH` 语句。

openCypher 兼容性

过滤 Rank 是原生 nGQL 功能。如需在 openCypher 兼容语句中直接获取 Rank 值，可以使用 rank() 函数，例如 `MATCH (:player)-[e:follow]->() RETURN rank(e);` 。

基础用法

Note

下文示例中的 `$$` 、`$^` 等是引用符号，详情请参见引用符。

用布尔运算符定义条件

在 `WHERE` 子句中使用布尔运算符 `NOT` 、`AND` 、`OR` 和 `XOR` 定义条件。关于运算符的优先级，请参见运算符优先级。

```
nebula> MATCH (v:player) \
        WHERE v.player.name == "Tim Duncan" \
        XOR (v.player.age < 30 AND v.player.name == "Yao Ming") \
        OR NOT (v.player.name == "Yao Ming" OR v.player.name == "Tim Duncan") \
        RETURN v.player.name, v.player.age;
+------------------------+--------------+
| v.player.name          | v.player.age |
+------------------------+--------------+
| "Danny Green"          | 31           |
| "Tiago Splitter"       | 34           |
| "David West"           | 38           |
...
```

```
nebula> GO FROM "player100" \
        OVER follow \
        WHERE properties(edge).degree > 90 \
        OR properties($$).age != 33 \
        AND properties($$).name != "Tony Parker" \
        YIELD properties($$);
+---------------------------------+
| properties($$)                  |
+---------------------------------+
| {age: 41, name: "Manu Ginobili"} |
+---------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

过滤属性

在 `WHERE` 子句中使用点或边的属性定义条件。

- 过滤点属性：

```
nebula> MATCH (v:player)-[e]->(v2) \
        WHERE v2.player.age < 25 \
        RETURN v2.player.name, v2.player.age;
+----------------------+--------------+
| v2.player.name       | v2.player.age |
+----------------------+--------------+
| "Ben Simmons"        | 22           |
| "Luka Doncic"        | 20           |
| "Kristaps Porzingis" | 23           |
+----------------------+--------------+
```

```
nebula> GO FROM "player100" OVER follow \
        WHERE $^.player.age >= 42 \
        YIELD dst(edge);
+------------+
| dst(EDGE)  |
+------------+
| "player101" |
| "player125" |
+------------+
```

- 过滤边属性：

```
nebula> MATCH (v:player)-[e]->() \
        WHERE e.start_year < 2000 \
        RETURN DISTINCT v.player.name, v.player.age;
+--------------------+-------------+
| v.player.name      | v.player.age |
+--------------------+-------------+
| "Tony Parker"      | 36          |
| "Tim Duncan"       | 42          |
| "Grant Hill"       | 46          |
...
```

```
nebula> GO FROM "player100" OVER follow \
        WHERE follow.degree > 90 \
        YIELD dst(edge);
+------------+
| dst(EDGE)  |
+------------+
| "player101" |
| "player125" |
+------------+
```

过滤动态计算属性

```
nebula> MATCH (v:player) \
        WHERE v[toLower("AGE")] < 21 \
        RETURN v.player.name, v.player.age;
+---------------+-------+
| v.name        | v.age |
+---------------+-------+
| "Luka Doncic" | 20    |
+---------------+-------+
```

过滤现存属性

```
nebula> MATCH (v:player) \
        WHERE exists(v.player.age) \
        RETURN v.player.name, v.player.age;
+-------------------------+-------------+
| v.player.name           | v.player.age |
+-------------------------+-------------+
| "Danny Green"           | 31          |
| "Tiago Splitter"        | 34          |
| "David West"            | 38          |
...
```

过滤 **RANK**

在 nGQL 中，如果多个边拥有相同的起始点、目的点和属性，则它们的唯一区别是 rank 值。在 `WHERE` 子句中可以使用 rank 过滤边。

```
# 创建测试数据。
nebula> CREATE SPACE IF NOT EXISTS test (vid_type=FIXED_STRING(30));
nebula> USE test;
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> CREATE EDGE IF NOT EXISTS e1(p1 int);
nebula> CREATE TAG IF NOT EXISTS person(p1 int);
nebula> INSERT VERTEX person(p1) VALUES "1":(1);
nebula> INSERT VERTEX person(p1) VALUES "2":(2);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@0:(10);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@1:(11);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@2:(12);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@3:(13);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@4:(14);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@5:(15);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@6:(16);

# 通过 rank 过滤边，查找 rank 大于 2 的边。
nebula> GO FROM "1" \
        OVER e1 \
        WHERE rank(edge) > 2 \
        YIELD src(edge), dst(edge), rank(edge) AS Rank, properties(edge).p1 | \
        ORDER BY $-.Rank DESC;
+-----------+-----------+------+--------------------+
| src(EDGE) | dst(EDGE) | Rank | properties(EDGE).p1 |
+-----------+-----------+------+--------------------+
| "1"       | "2"       | 6    | 16                 |
| "1"       | "2"       | 5    | 15                 |
| "1"       | "2"       | 4    | 14                 |
| "1"       | "2"       | 3    | 13                 |
+-----------+-----------+------+--------------------+

# 通过 rank 过滤边，查找 rank 值等于 0 的 follow 边。
nebula> MATCH (v)-[e:follow]->() \
        WHERE rank(e)==0 \
        RETURN *;
+-------------------------------------------------------+-----------------------------------------------+
| v                                                     | e                                             |
+-------------------------------------------------------+-----------------------------------------------+
| ("player142" :player{age: 29, name: "Klay Thompson"})      | [:follow "player142"->"player117" @0 {degree: 90}]  |
| ("player139" :player{age: 34, name: "Marc Gasol"})         | [:follow "player139"->"player138" @0 {degree: 99}]  |
| ("player108" :player{age: 36, name: "Boris Diaw"})         | [:follow "player108"->"player100" @0 {degree: 80}]  |
| ("player108" :player{age: 36, name: "Boris Diaw"})         | [:follow "player108"->"player101" @0 {degree: 80}]  |
...
```

过滤 **PATTERN**

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->(t) \
        WHERE (v)-[e]->(t:team) \
        RETURN (v)-->();

+-----------------------------------------------------------
```

```
        WHERE v.player.name STARTS WITH "T" \
        RETURN v.player.name, v.player.age;
+------------------+--------------+
| v.player.name    | v.player.age |
+------------------+--------------+
| "Tony Parker"    | 36           |
| "Tiago Splitter" | 34           |
| "Tim Duncan"     | 42           |
| "Tracy McGrady"  | 39           |
+------------------+--------------+
```

如果使用小写 `t` （`STARTS WITH "t"`），会返回空集，因为数据库中没有以小写 `t` 开头的姓名。

```
nebula> MATCH (v:player) \
        WHERE v.player.name STARTS WITH "t" \
        RETURN v.player.name, v.player.age;
+---------------+--------------+
| v.player.name | v.player.age |
+---------------+--------------+
+---------------+--------------+
Empty set (time spent 5080/6474 us)
```

`ENDS WITH`

`ENDS WITH` 会从字符串的结束位置开始匹配。

```
nebula> MATCH (v:player) \
        WHERE v.player.name ENDS WITH "r" \
        RETURN v.player.name, v.player.age;
+------------------+--------------+
| v.player.name    | v.player.age |
+------------------+--------------+
| "Tony Parker"    | 36           |
| "Tiago Splitter" | 34           |
| "Vince Carter"   | 42           |
+------------------+--------------+
```

`CONTAINS`

`CONTAINS` 会检查关键字是否匹配字符串的某一部分。

```
nebula> MATCH (v:player) \
        WHERE v.player.name CONTAINS "Pa" \
        RETURN v.player.name, v.player.age;
+---------------+--------------+
| v.player.name | v.player.age |
+---------------+--------------+
| "Paul George" | 28           |
| "Tony Parker" | 36           |
| "Paul Gasol"  | 38           |
| "Chris Paul"  | 33           |
+---------------+--------------+
```

结合 `NOT` 使用

用户可以结合布尔运算符 `NOT` 一起使用，否定字符串匹配条件。

```
nebula> MATCH (v:player) \
        WHERE NOT v.player.name ENDS WITH "R" \
        RETURN v.player.name, v.player.age;
+-------------------------+--------------+
| v.player.name           | v.player.age |
+-------------------------+--------------+
| "Danny Green"           | 31           |
| "Tiago Splitter"        | 34           |
| "David West"            | 38           |
| "Russell Westbrook"     | 30           |
...
```

过滤列表

匹配列表中的值

使用 `IN` 运算符检查某个值是否在指定列表中。

```
nebula> MATCH (v:player) \
        WHERE v.player.age IN range(20,25) \
        RETURN v.player.name, v.player.age;
+-------------------------+--------------+
| v.player.name           | v.player.age |
+-------------------------+--------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "Ben Simmons"           | 22           |
| "Giannis Antetokounmpo" | 24           |
| "Kyle Anderson"         | 25           |
| "Joel Embiid"           | 25           |
| "Kristaps Porzingis"    | 23           |
| "Luka Doncic"           | 20           |
+-------------------------+--------------+

nebula> LOOKUP ON player \
        WHERE player.age IN [25,28]  \
        YIELD properties(vertex).name, properties(vertex).age;
+-------------------------+-------------------------+
| properties(VERTEX).name | properties(VERTEX).age |
+-------------------------+-------------------------+
| "Kyle Anderson"         | 25                      |
| "Damian Lillard"        | 28                      |
| "Joel Embiid"           | 25                      |
| "Paul George"           | 28                      |
| "Ricky Rubio"           | 28                      |
+-------------------------+-------------------------+
```

结合 **NOT** 使用

```
nebula> MATCH (v:player) \
        WHERE v.player.age NOT IN range(20,25) \
        RETURN v.player.name AS Name, v.player.age AS Age \
        ORDER BY Age;
+---------------------+-----+
| Name                | Age |
+---------------------+-----+
| "Kyrie Irving"      | 26  |
| "Cory Joseph"       | 27  |
| "Damian Lillard"    | 28  |
| "Paul George"       | 28  |
| "Ricky Rubio"       | 28  |
...
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.8 YIELD

`YIELD` 定义 nGQL 查询的输出结果。

`YIELD` 可以引导子句或语句：

- `YIELD` 子句用于原生 nGQL 语句中，例如 `GO` 、 `FETCH` 或 `LOOKUP` ，必须通过 `YIELD` 子句定义返回结果。
- `YIELD` 语句可以在独立查询或复合查询中使用。

openCypher 兼容性

本文操作仅适用于原生 nGQL。关于 openCypher 方式如何定义输出结果，请参见 `RETURN` 。

`YIELD` 在 nGQL 和 openCypher 中有不同的函数：

- 在 openCypher 中， `YIELD` 用于在 `CALL[...YIELD]` 子句中指定过程调用的输出。

  Note

  nGQL 不支持 `CALL[...YIELD]` 。

- 在 nGQL 中， `YIELD` 和 openCypher 中的 `RETURN` 类似。

  Note

  下文示例中的 `$$` 、 `$-` 等是引用符号，详情请参见引用符。

YIELD 子句

语法

```
YIELD [DISTINCT] <col> [AS <alias>] [, <col> [AS <alias>] ...];
```

参数	说明
`DISTINCT`	聚合输出结果，返回去重后的结果集。
`col`	要返回的字段。如果没有为字段设置别名，返回结果中的列名为 `col` 。
`alias`	`col` 的别名。使用关键字 `AS` 进行设置，设置后返回结果中的列名为该别名。

使用 **YIELD** 子句

- `GO` 语句中使用 `YIELD` ：

```
nebula> GO FROM "player100" OVER follow \
        YIELD properties($$).name AS Friend, properties($$).age AS Age;
+-----------------+-----+
| Friend          | Age |
+-----------------+-----+
| "Tony Parker"   | 36  |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "Manu Ginobili" | 41  |
+-----------------+-----+
```

- `FETCH` 语句中使用 `YIELD`：

```
nebula> FETCH PROP ON player "player100" \
        YIELD properties(vertex).name;
+-------------------------+
| properties(VERTEX).name |
+-------------------------+
| "Tim Duncan"            |
+-------------------------+
```

- `LOOKUP` 语句中使用 `YIELD`：

```
nebula> LOOKUP ON player WHERE player.name == "Tony Parker" \
        YIELD properties(vertex).name, properties(vertex).age;
+-------------------------+------------------------+
| properties(VERTEX).name | properties(VERTEX).age |
+-------------------------+------------------------+
| "Tony Parker"           | 36                     |
+-------------------------+------------------------+
```

YIELD 语句

语法

```
YIELD [DISTINCT] <col> [AS <alias>] [, <col> [AS <alias>] ...]
[WHERE <conditions>];
```

参数	说明
`DISTINCT`	聚合输出结果，返回去重后的结果集。
`col`	要按返回的字段。如果没有为字段设置别名，返回结果中的列名为 `col` 。
`alias`	`col` 的别名。使用关键字 `AS` 进行设置，设置后返回结果中的列名为该别名。
`conditions`	在 `WHERE` 子句中设置的过滤条件。详情请参见 `WHERE` 。

复合查询中使用 **YIELD** 语句

在复合查询中， `YIELD` 语句可以接收、过滤、修改之前语句的结果集，然后输出。

```
# 查找 player100 关注的 player，并计算他们的平均年龄。
nebula> GO FROM "player100" OVER follow \
        YIELD dst(edge) AS ID \
        | FETCH PROP ON player $-.ID \
        YIELD properties(vertex).age AS Age \
        | YIELD AVG($-.Age) as Avg_age, count(*)as Num_friends;
+---------+-------------+
| Avg_age | Num_friends |
+---------+-------------+
| 38.5    | 2           |
+---------+-------------+
```

```
# 查找 player101 关注的 player，返回 degree 大于 90 的 player。
nebula> $var1 = GO FROM "player101" OVER follow \
        YIELD properties(edge).degree AS Degree, dst(edge) as ID; \
        YIELD $var1.ID AS ID WHERE $var1.Degree > 90;
+-------------+
| ID          |
+-------------+
| "player100" |
| "player125" |
+-------------+

# 查找 player 中年龄大于 30 且小于 32 的点，返回去掉重复属性的值。
nebula> LOOKUP ON player  \
        WHERE player.age < 32 and player.age >30  \
        YIELD DISTINCT properties(vertex).age as v;
+--------+
| v      |
+--------+
| 31     |
+--------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

独立使用 **YIELD** 语句

`YIELD` 可以计算表达式并返回结果。

```
nebula> YIELD rand32(1, 6);
+-------------+
| rand32(1,6) |
+-------------+
| 3           |
+-------------+

nebula> YIELD "Hel" + "\tlo" AS string1, ", World!" AS string2;
+-------------+------------+
| string1     | string2    |
+-------------+------------+
| "Hel    lo" | ", World!" |
+-------------+------------+

nebula> YIELD hash("Tim") % 100;
+-------------------+
| (hash("Tim")%100) |
+-------------------+
| 42                |
+-------------------+

nebula> YIELD \
      CASE 2+3 \
      WHEN 4 THEN 0 \
      WHEN 5 THEN 1 \
      ELSE -1 \
      END \
      AS result;
+--------+
| result |
+--------+
| 1      |
+--------+

nebula> YIELD 1- -1;
+----------+
| (1--(1)) |
+----------+
| 2        |
+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.9 WITH

`WITH` 子句可以获取并处理查询前半部分的结果，并将处理结果作为输入传递给查询的后半部分。

openCypher 兼容性

本文操作仅适用于 openCypher 方式。

> **Note**
>
> 在原生 nGQL 中，有与 `WITH` 类似的管道符，但它们的工作方式不同。不要在 openCypher 方式中使用管道符，也不要在原生 nGQL 中使用 `WITH` 子句。

组成复合查询

使用 `WITH` 子句可以组合语句，将一条语句的输出转换为另一条语句的输入。

示例 **1**

1. 匹配一个路径。
2. 通过 `nodes()` 函数将路径上的所有点输出到一个列表。
3. 将列表拆分为行。
4. 去重后返回点的信息。

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})--() \
        WITH nodes(p) AS n \
        UNWIND n AS n1 \
        RETURN DISTINCT n1;
+-------------------------------------------------------+
| n1                                                    |
+-------------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"})    |
| ("player101" :player{age: 36, name: "Tony Parker"})   |
| ("team204" :team{name: "Spurs"})                      |
| ("player102" :player{age: 33, name: "LaMarcus Aldridge"}) |
| ("player125" :player{age: 41, name: "Manu Ginobili"}) |
| ("player104" :player{age: 32, name: "Marco Belinelli"}) |
| ("player144" :player{age: 47, name: "Shaquille O'Neal"}) |
| ("player105" :player{age: 31, name: "Danny Green"})   |
| ("player113" :player{age: 29, name: "Dejounte Murray"}) |
| ("player107" :player{age: 32, name: "Aron Baynes"})   |
| ("player109" :player{age: 34, name: "Tiago Splitter"}) |
| ("player108" :player{age: 36, name: "Boris Diaw"})    |
+-------------------------------------------------------+
```

示例 **2**

1. 匹配点 ID 为 `player100` 的点。
2. 通过 `labels()` 函数将点的所有 Tag 输出到一个列表。
3. 将列表拆分为行。
4. 返回结果。

```
nebula> MATCH (v) \
        WHERE id(v)=="player100" \
        WITH labels(v) AS tags_unf \
        UNWIND tags_unf AS tags_f \
        RETURN tags_f;
+----------+
| tags_f   |
+----------+
| "player" |
+----------+
```

过滤聚合查询

`WITH` 可以在聚合查询中作为过滤器使用。

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> MATCH (v:player)-->(v2:player) \
        WITH DISTINCT v2 AS v2, v2.player.age AS Age \
        ORDER BY Age \
        WHERE Age<25 \
        RETURN v2.player.name AS Name, Age;
+---------------------+-----+
| Name                | Age |
+---------------------+-----+
| "Luka Doncic"       | 20  |
| "Ben Simmons"       | 22  |
| "Kristaps Porzingis" | 23  |
+---------------------+-----+
```

collect() 之前处理输出

在 `collect()` 函数将输出结果转换为列表之前，可以使用 `WITH` 子句排序和限制输出结果。

```
nebula> MATCH (v:player) \
        WITH v.player.name AS Name \
        ORDER BY Name DESC \
        LIMIT 3 \
        RETURN collect(Name);
+---------------------------------------------+
| collect(Name)                               |
+---------------------------------------------+
| ["Yao Ming", "Vince Carter", "Tracy McGrady"] |
+---------------------------------------------+
```

结合 RETURN 语句使用

在 `WITH` 子句中设置别名，并通过 `RETURN` 子句输出结果。

```
nebula> WITH [1, 2, 3] AS `list` RETURN 3 IN `list` AS r;
+------+
| r    |
+------+
| true |
+------+

nebula> WITH 4 AS one, 3 AS two RETURN one > two AS result;
+--------+
| result |
+--------+
| true   |
+--------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.7.10 UNWIND

`UNWIND` 语句可以将列表拆分为单独的行，列表中的每个元素为一行。

`UNWIND` 可以作为单独语句或语句中的子句使用。

UNWIND 语句

语法

```
UNWIND <list> AS <alias> <RETURN clause>;
```

示例

- 拆分列表。

```
nebula> UNWIND [1,2,3] AS n RETURN n;
+---+
| n |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

UNWIND 子句

语法

- 原生 nGQL 语句中使用 `UNWIND` 子句。

  Note

  在原生 nGQL 语句中使用 `UNWIND` 子句时，需要用在管道符 `|` 之后，并使用 `$-` 引用管道符之前的变量。如果 `UNWIND` 后使用语句或子句，需要使用管道符 `|` 并且使用 `$-` 引用管道符之前的变量。

```
<statement> | UNWIND $-.<var> AS <alias> <|> <clause>;
```

- openCypher 语句中使用 `UNWIND` 子句。

```
<statement> UNWIND <list> AS <alias> <RETURN clause> ;
```

示例

- 在 `UNWIND` 子句中使用 `WITH DISTINCT` 可以将列表中的重复项忽略，返回去重后的结果。

  Note

  原生 nGQL 语句不支持 `WITH DISTINCT` 。

```
// 拆分列表`[1,1,2,2,3,3]`，删除重复行，排序行，将行转换为列表。
nebula> WITH [1,1,2,2,3,3] AS n \
        UNWIND n AS r \
        WITH DISTINCT r AS r \
        ORDER BY r \
        RETURN collect(r);
+------------+
| collect(r) |
+------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| [1, 2, 3]  |
+------------+
```

- `MATCH` 语句中使用 `UNWIND` 。

```
// 将匹配路径上的顶点输出到列表中，拆分列表，删除重复行，将行转换为列表。
nebula> MATCH p=(v:player{name:"Tim Duncan"})--(v2) \
        WITH nodes(p) AS n \
        UNWIND n AS r \
        WITH DISTINCT r AS r \
        RETURN collect(r);
+------------------------------------------------------------------------------------------------------------------------------+
| collect(r)                                                                                                                   |
+------------------------------------------------------------------------------------------------------------------------------+
| [("player100" :player{age: 42, name: "Tim Duncan"}), ("player101" :player{age: 36, name: "Tony Parker"}),                    |
|("team204" :team{name: "Spurs"}), ("player102" :player{age: 33, name: "LaMarcus Aldridge"}),                                  |
|("player125" :player{age: 41, name: "Manu Ginobili"}), ("player104" :player{age: 32, name: "Marco Belinelli"}),               |
|("player144" :player{age: 47, name: "Shaquile O'Neal"}), ("player105" :player{age: 31, name: "Danny Green"}),                |
|("player113" :player{age: 29, name: "Dejounte Murray"}), ("player107" :player{age: 32, name: "Aron Baynes"}),                |
|("player109" :player{age: 34, name: "Tiago Splitter"}), ("player108" :player{age: 36, name: "Boris Diaw"})]                  |
+------------------------------------------------------------------------------------------------------------------------------+
```

- `GO` 语句中使用 `UNWIND` 。

```
// 在点列表中查询点关联的边。
nebula> YIELD ['player101', 'player100'] AS a | UNWIND $-.a AS  b | GO FROM $-.b OVER follow YIELD edge AS e;
+--------------------------------------------------+
| e                                                |
+--------------------------------------------------+
| [:follow "player101"->"player100" @0 {degree: 95}] |
| [:follow "player101"->"player102" @0 {degree: 90}] |
| [:follow "player101"->"player125" @0 {degree: 95}] |
| [:follow "player100"->"player101" @0 {degree: 95}] |
| [:follow "player100"->"player125" @0 {degree: 95}] |
+--------------------------------------------------+
```

- `LOOKUP` 语句中使用 `UNWIND` 。

```
// 查询年龄大于 46 岁球员的所有属性，去掉重复属性，并将结果转换为行。
nebula> LOOKUP ON player \
        WHERE player.age > 46 \
        YIELD DISTINCT keys(vertex) as p | UNWIND $-.p as a | YIELD $-.a AS a;
+--------+
| a      |
+--------+
| "age"  |
| "name" |
+--------+
```

- `FETCH` 语句中使用 `UNWIND` 。

```
// 查询 player101 点的所有 Tag，并将结果转换为行。
nebula> CREATE TAG hero(like string, height int);
        INSERT VERTEX hero(like, height) VALUES "player101":("deep", 182);
        FETCH PROP ON * "player101" \
        YIELD tags(vertex) as t | UNWIND $-.t as a | YIELD $-.a AS a;
+----------+
| a        |
+----------+
| "hero"   |
| "player" |
+----------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

- `GET SUBGRAPH` 语句中使用 `UNWIND` 。

```
// 查询从点 player100 开始、0~2 跳、serve 类型的出边和入边的子图，并将结果转换为行。
nebula> GET SUBGRAPH 2 STEPS FROM "player100" BOTH serve \
        YIELD edges as e | UNWIND $-.e as a | YIELD $-.a AS a;
+---------------------------------------------+
| a                                           |
+---------------------------------------------+
| [:serve "player100"->"team204" @0 {}]       |
| [:serve "player101"->"team204" @0 {}]       |
| [:serve "player102"->"team204" @0 {}]       |
| [:serve "player103"->"team204" @0 {}]       |
| [:serve "player105"->"team204" @0 {}]       |
| [:serve "player106"->"team204" @0 {}]       |
| [:serve "player107"->"team204" @0 {}]       |
| [:serve "player108"->"team204" @0 {}]       |
| [:serve "player109"->"team204" @0 {}]       |
| [:serve "player110"->"team204" @0 {}]       |
| [:serve "player111"->"team204" @0 {}]       |
| [:serve "player112"->"team204" @0 {}]       |
| [:serve "player113"->"team204" @0 {}]       |
| [:serve "player114"->"team204" @0 {}]       |
| [:serve "player125"->"team204" @0 {}]       |
| [:serve "player138"->"team204" @0 {}]       |
| [:serve "player104"->"team204" @20132015 {}] |
| [:serve "player104"->"team204" @20182019 {}] |
+---------------------------------------------+
```

- `FIND PATH` 语句中使用 `UNWIND` 。

```
// 找出 player101 到 team204 延 serve 类型边的最短路径上的所有点，并将结果转换为行。
nebula> FIND SHORTEST PATH FROM "player101" TO "team204" OVER serve \
        YIELD path as p | YIELD nodes($-.p) AS nodes | UNWIND $-.nodes AS a | YIELD $-.a AS a;
+---------------+
| a             |
+---------------+
| ("player101") |
| ("team204")   |
+---------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.8 图空间语句

5.8.1 CREATE SPACE

图空间是悦数图数据库中彼此隔离的图数据集合，与 MySQL 中的 database 概念类似。`CREATE SPACE` 语句可以创建一个新的图空间，或者克隆现有图空间的 Schema。

前提条件

只有 God 角色的用户可以执行 `CREATE SPACE` 语句。详情请参见身份验证。

语法

创建图空间

```
CREATE SPACE [IF NOT EXISTS] <graph_space_name> (
    [partition_num = <partition_number>,]
    [replica_factor = <replica_number>,]
    vid_type = {FIXED_STRING(<N>) | INT[64]}
    )
    [COMMENT = '<comment>'];
```

参数	说明
`IF NOT EXISTS`	检测待创建的图空间是否存在，只有不存在时，才会创建图空间。仅检测图空间的名称，不会检测具体属性。
`<graph_space_name>`	1、在悦数图数据库实例中唯一标识一个图空间。 2、图空间名称设置后无法被修改。 3、不能以数字开头；支持 1~4 字节的 UTF-8 编码字符，包括英文字母（区分大小写）、数字、中文等，但是不包括除下划线外的特殊字符；使用特殊字符、保留关键字或数字开头时，需要用反引号（`）包围且不能使用英文句号（`.`）。详情参见关键字和保留字。 注意：如果以中文为图空间命名，报 `SyntaxError` 错误时，需使用反引号（`）包围中文字符。
`partition_num`	指定图空间的分片数量。建议设置为集群中硬盘数量的 20 倍（HDD 硬盘建议为 2 倍）。例如集群中有 3 个硬盘，建议设置 60 个分片。默认值为 100。
`replica_factor`	指定每个分片的副本数量。建议在生产环境中设置为 3，在测试环境中设置为 1。由于需要基于多数表决，副本数量必须是奇数。默认值为 1。
`vid_type`	必选参数。指定点 ID 的数据类型。可选值为 `FIXED_STRING(<N>)` 和 `INT64`。`INT` 等同于 `INT64`。 `FIXED_STRING(<N>)` 表示数据类型为定长字符串，长度为 `N` 字节，超出长度会报错。例如，UTF-8中，一个中文字符的长度为三个字节，如果设置 `N` 为 12，那么 `vid_type` 为最多 4 个中文字符。 `INT64` 表示数据类型为整数。
`COMMENT`	图空间的描述。最大为 256 字节。默认无描述。

Copyright © Yueshu Graph Technology - All rights reserved.

Caution

- 如果将副本数设置为 1，用户将无法使用 SUBMIT JOB BALANCE 命令为悦数图数据库的存储服务平衡负载或扩容。

- VID 类型变更与长度限制：
- 在悦数图数据库 1.x 中，VID 的类型只能为 `INT64`，不支持字符型；在悦数图数据库 2.x 中，VID 的类型支持 `INT64` 和 `FIXED_STRING(<N>)`。请在创建图空间时指定 VID 类型，使用 `INSERT` 语句时也需要保持一致，否则会报错 VID 类型不匹配 `Wrong vertex id type: 1001`。
- VID 最大长度必须为 `N`，不可任意长度；超过该长度也会报错 `The VID must be a 64-bit integer or a string fitting space vertex id length limit.`。

- 如果出现 `Host not enough!` 的报错，直接原因是线上 Storage Host 的数量少于即将创建的图空间的 `replica_factor` 的数量。此时，可用 `SHOW HOSTS` 指令查看判断，出现的情况有：
- 在集群是单 Storage Host 的情况下试图创建多副本（replica_factor）图空间，这时候只能选择创建单副本数的图空间，或者扩容 Storage Host 之后再创建图空间。
- 新创建的集群里 Storage Host 已经被服务发现，但是尚未执行 `ADD HOSTS` 将其激活，这时候需要通过 Console 连接，执行 `SHOW HOSTS` 获取被发现了的 Storage Host，然后执行相应的 `ADD HOSTS` 激活，待有足够的 Online Storage Host 之后再尝试创建图空间。
- 有部分 Storage Host 处在非 Online 状态，需要进行进一步排查。

版本兼容性

2.5.0 之前的 2.x 版本中，`vid_type` 不是必选参数，默认为 `FIXED_STRING(8)`。

Note

`graph_space_name`，`partition_num`，`replica_factor`，`vid_type`，`comment` 设置后就无法改变。除非 `DROP SPACE`，并重新 `CREATE SPACE`。

克隆图空间

```
CREATE SPACE [IF NOT EXISTS] <new_graph_space_name> AS <old_graph_space_name>;
```

参数	说明
`IF NOT EXISTS`	检测待创建的图空间是否存在，只有不存在时，才会克隆图空间。仅检测图空间的名称，不会检测具体属性。
`<new_graph_space_name>`	目标图空间名称。该图空间必须未创建。图空间名称以英文字母开头，支持 1~4 字节的 UTF-8 编码字符，包括英文字母（区分大小写）、数字、中文等，但是不包括除下划线外的特殊字符。使用保留关键字时，需要用反引号（`）包围，详情参见关键字和保留字。创建时会克隆 `<old_graph_space_name>` 图空间的 Schema，包括图空间本身参数（分片数量、副本数量等）、Tag、Edge type 和 原生索引。
`<old_graph_space_name>`	原始图空间名称。该图空间必须已存在。

示例

```
# 仅指定 VID 类型，其他选项使用默认值。
nebula> CREATE SPACE IF NOT EXISTS my_space_1 (vid_type=FIXED_STRING(30));

# 指定分片数量、副本数量和 VID 类型。
nebula> CREATE SPACE IF NOT EXISTS my_space_2 (partition_num=15, replica_factor=1, vid_type=FIXED_STRING(30));

# 指定分片数量、副本数量和 VID 类型，并添加描述。
nebula> CREATE SPACE IF NOT EXISTS my_space_3 (partition_num=15, replica_factor=1, vid_type=FIXED_STRING(30)) comment="测试图空间";

# 克隆图空间。
nebula> CREATE SPACE IF NOT EXISTS my_space_4 as my_space_3;
nebula> SHOW CREATE SPACE my_space_4;
+--------------+------------------
```

Caution

立刻尝试使用刚创建的图空间可能会失败。因为创建是异步实现的。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。如果需要修改心跳间隔，请为所有配置文件修改参数 `heartbeat_interval_secs` 。但过短的心跳周期（<5 秒）可能会导致分布式系统中的机器误判对端失联。

检查分片分布情况

在大型集群中，由于启动时间不同，分片的分布可能不均衡。用户可以执行如下命令检查分片的分布情况：

```
nebula> SHOW HOSTS;
+-------------+------+----------+--------------+-------------------------------+-------------------------------+---------+
| Host        | Port | Status   | Leader count | Leader distribution           | Partition distribution        | Version |
+-------------+------+----------+--------------+-------------------------------+-------------------------------+---------+
| "storaged0" | 9779 | "ONLINE" | 8            | "basketballplayer:3, test:5"  | "basketballplayer:10, test:10" | "3.1.0" |
| "storaged1" | 9779 | "ONLINE" | 9            | "basketballplayer:4, test:5"  | "basketballplayer:10, test:10" | "3.1.0" |
| "storaged2" | 9779 | "ONLINE" | 3            | "basketballplayer:3"          | "basketballplayer:10, test:10" | "3.1.0" |
+-------------+------+----------+--------------+-------------------------------+-------------------------------+---------+
```

如果需要均衡负载，请执行如下命令：

```
nebula> BALANCE LEADER;
nebula> SHOW HOSTS;
+-------------+------+----------+--------------+-------------------------------+-------------------------------+---------+
| Host        | Port | Status   | Leader count | Leader distribution           | Partition distribution        | Version |
+-------------+------+----------+--------------+-------------------------------+-------------------------------+---------+
| "storaged0" | 9779 | "ONLINE" | 7            | "basketballplayer:3, test:4"  | "basketballplayer:10, test:10" | "3.5.1" |
| "storaged1" | 9779 | "ONLINE" | 7            | "basketballplayer:4, test:3"  | "basketballplayer:10, test:10" | "3.5.1" |
| "storaged2" | 9779 | "ONLINE" | 6            | "basketballplayer:3, test:3"  | "basketballplayer:10, test:10" | "3.5.1" |
+-------------+------+----------+--------------+-------------------------------+-------------------------------+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.8.2 USE

`USE` 语句可以指定一个图空间，或切换到另一个图空间，将其作为后续查询的工作空间。

前提条件

执行 `USE` 语句指定图空间时，需要当前登录的用户拥有指定图空间的权限，否则会报错。

语法

```
USE <graph_space_name>;
```

示例

```
# 创建示例空间。
nebula> CREATE SPACE IF NOT EXISTS space1 (vid_type=FIXED_STRING(30));
nebula> CREATE SPACE IF NOT EXISTS space2 (vid_type=FIXED_STRING(30));

# 指定图空间 space1 作为工作空间。
nebula> USE space1;

# 切换到图空间 space2。检索 space2 时，无法从 space1 读取任何数据，检索的点和边与 space1 无关。
nebula> USE space2;
```

Caution

不能在一条语句中同时操作两个图空间。

与 Fabric Cypher 不同，悦数图数据库的图空间彼此之间是完全隔离的，将一个图空间作为工作空间后，用户无法访问其他空间。使用新图空间的唯一方法是通过 `USE` 语句切换。而在 Fabric Cypher 中可以在一条语句中（`USE + CALL` 语法）使用两个图空间。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.8.3 SHOW SPACES

`SHOW SPACES` 语句可以列出悦数图数据库示例中的所有图空间。

语法

```
SHOW SPACES;
```

示例

```
nebula> SHOW SPACES;
+-------------------+
| Name              |
+-------------------+
| "cba"             |
| "basketballplayer" |
+-------------------+
```

创建图空间请参见 CREATE SPACE。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.8.4 DESCRIBE SPACE

`DESCRIBE SPACE` 语句可以显示指定图空间的信息。

语法

你可以用 `DESC` 作为 `DESCRIBE` 的缩写。

```
DESC[RIBE] SPACE <graph_space_name>;
```

示例

```
nebula> DESCRIBE SPACE basketballplayer;
+----+--------------------+------------------+----------------+---------+------------+--------------------+---------+
| ID | Name               | Partition Number | Replica Factor | Charset | Collate    | Vid Type           | Comment |
+----+--------------------+------------------+----------------+---------+------------+--------------------+---------+
| 1  | "basketballplayer" | 10               | 1              | "utf8"  | "utf8_bin" | "FIXED_STRING(32)" |         |
+----+--------------------+------------------+----------------+---------+------------+--------------------+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.8.5 CLEAR SPACE

`CLEAR SPACE` 语句用于清空图空间中的点和边，但不会删除图空间本身以及其中的 Schema 信息。

> **Note**
>
> 建议在执行 `CLEAR SPACE` 操作之后，立即执行 `SUBMIT JOB COMPACT` 操作以提升查询性能。需要注意的是，COMPACT 操作可能会影响查询性能，建议在业务低峰期（例如凌晨）执行该操作。

权限要求

只有 God 角色的用户可以执行 `CLEAR SPACE` 语句。

注意事项

- 数据清除后，如无备份，无法恢复。使用该功能务必谨慎。
- `CLEAR SPACE` 不是原子性操作。如果执行出错，请重新执行，避免残留数据。
- 图空间中的数据量越大，`CLEAR SPACE` 消耗的时间越长。如果 `CLEAR SPACE` 的执行因客户端连接超时而失败，可以增大 Graph 服务配置中 `storage_client_timeout_ms` 参数的值。
- 在 `CLEAR SPACE` 的执行过程中，向该图空间写入数据的行为不会被自动禁止。这样的写入行为可能导致 `CLEAR SPACE` 清除数据不完全，残留的数据也可能受到损坏。

> **Enterpriseonly**
>
> 企业版悦数图数据库支持先运行 `SET VARIABLE read_only=true` 阻止向悦数图数据库写入数据，再运行 `CLEAR SPACE` 。数据清除成功后运行 `SET VARIABLE read_only=false` 即可重新允许写入。

语法

```
CLEAR SPACE [IF EXISTS] <space_name>;
```

参数/选项	说明
`IF EXISTS`	检查待清空的图空间是否存在，如果图空间存在，则继续执行清空操作；如果图空间不存在，则完成执行，并且提示执行成功，不会提示图空间不存在。若不设置该选项，当图空间不存在时，`CLEAR SPACE` 语句会执行失败，系统会报错。
`space_name`	要清除数据的图空间名称。

示例：

```
CLEAR SPACE basketballplayer;
```

保留的数据

图空间中，`CLEAR SPACE` 不会删除的数据包括：

- Tag 信息。
- Edge type 信息。
- 原生索引和全文索引的元数据。

下面的执行示例明确展示了 `CLEAR SPACE` 会删除与保留的数据。

```
# 进入图空间 basketballplayer。
nebula[(none)]> use basketballplayer;
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
Execution succeeded

# 查看 Tag 和 Edge type。
nebula[basketballplayer]> SHOW TAGS;
+----------+
| Name     |
+----------+
| "player" |
| "team"   |
+----------+
Got 2 rows

nebula[basketballplayer]> SHOW EDGES;
+----------+
| Name     |
+----------+
| "follow" |
| "serve"  |
+----------+
Got 2 rows

# 统计图空间中的数据。
nebula[basketballplayer]> SUBMIT JOB STATS;
+------------+
| New Job Id |
+------------+
| 4          |
+------------+
Got 1 rows

# 查看统计结果。
nebula[basketballplayer]> SHOW STATS;
+---------+------------+-------+
| Type    | Name       | Count |
+---------+------------+-------+
| "Tag"   | "player"   | 51    |
| "Tag"   | "team"     | 30    |
| "Edge"  | "follow"   | 81    |
| "Edge"  | "serve"    | 152   |
| "Space" | "vertices" | 81    |
| "Space" | "edges"    | 233   |
+---------+------------+-------+
Got 6 rows

# 查看 Tag 索引。
nebula[basketballplayer]> SHOW TAG INDEXES;
+------------------+----------+----------+
| Index Name       | By Tag   | Columns  |
+------------------+----------+----------+
| "player_index_0" | "player" | []       |
| "player_index_1" | "player" | ["name"] |
+------------------+----------+----------+
Got 2 rows

# ---------------------- CLEAR SPACE 分割线 ----------------------
# 执行 CLEAR SPACE 清空图空间 basketballplayer。
nebula[basketballplayer]> CLEAR SPACE basketballplayer;
Execution succeeded

# 更新统计信息。
nebula[basketballplayer]> SUBMIT JOB STATS;
+------------+
| New Job Id |
+------------+
| 5          |
+------------+
Got 1 rows

# 查看统计信息。Tag 和 Edge type 还存在，但所有的点和边都没了。
nebula[basketballplayer]> SHOW STATS;
+---------+------------+-------+
| Type    | Name       | Count |
+---------+------------+-------+
| "Tag"   | "player"   | 0     |
| "Tag"   | "team"     | 0     |
| "Edge"  | "follow"   | 0     |
| "Edge"  | "serve"    | 0     |
| "Space" | "vertices" | 0     |
| "Space" | "edges"    | 0     |
+---------+------------+-------+
Got 6 rows

# 查看 Tag 索引，它们依然存在。
nebula[basketballplayer]> SHOW TAG INDEXES;
+------------------+----------+----------+
| Index Name       | By Tag   | Columns  |
+------------------+----------+----------+
| "player_index_0" | "player" | []       |
| "player_index_1" | "player" | ["name"] |
+------------------+----------+----------+
Got 2 rows (time spent 523/978 us)
```


Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.8.6 DROP SPACE

`DROP SPACE` 语句用于删除指定图空间以及其中的所有信息。

> **Note**
>
> `DROP SPACE` 是否删除图空间对应的硬盘数据由 Storage 配置参数 `auto_remove_invalid_space` 决定。 `auto_remove_invalid_space` 的默认值为 `true` ，表示会删除数据。如需在删除逻辑图空间时保留硬盘数据，将 `auto_remove_invalid_space` 的值修改为 `false` 。详情参见 Storage 服务配置。

> **Warning**
>
> 执行 `DROP SPACE` 后，即使快照中存在该图空间的数据，该图空间的数据也无法恢复。但是如果 `auto_remove_invalid_space` 的值为 `false` 时，可联系商务人员恢复数据。

前提条件

只有 God 角色的用户可以执行 `DROP SPACE` 语句。详情请参见身份验证。

语法

```
DROP SPACE [IF EXISTS] <graph_space_name>;
```

`IF EXISTS` 关键字可以检测待删除的图空间是否存在，只有存在时，才会删除图空间。

> 版本兼容性
>
> 在悦数图数据库 3.1.0 版本前， `DROP SPACE` 语句不会删除硬盘上对应图空间的目录和文件。

> **Caution**
>
> 请谨慎执行删除图空间操作。

FAQ

问：执行 `DROP SPACE` 语句删除图空间后，为什么磁盘的大小没变化？

答：如果使用 3.1.0 之前版本的悦数图数据库，`DROP SPACE` 语句仅删除指定的逻辑图空间，不会删除硬盘上对应图空间的目录和文件。如需删除硬盘上的数据，需手动删除相应文件的路径，文件路径为 `<nebula_graph_install_path>/data/storage/nebula/<space_id>` 。其中 `<space_id>` 可以通过 `DESCRIBE SPACE {space_name}` 查看。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9 Tag 语句

5.9.1 CREATE TAG

`CREATE TAG` 语句可以通过指定名称创建一个 Tag。

OpenCypher 兼容性

nGQL 中的 Tag 和 openCypher 中的 Label 相似，但又有所不同，例如它们的创建方式。

- openCypher 中的 Label 需要在 `CREATE` 语句中与点一起创建。
- nGQL 中的 Tag 需要使用 `CREATE TAG` 语句独立创建。Tag 更像是 MySQL 中的表。

前提条件

执行 `CREATE TAG` 语句需要当前登录的用户拥有指定图空间的创建 Tag 权限，否则会报错。

语法

创建 Tag 前，需要先用 `USE` 语句指定工作空间。

```
CREATE TAG [IF NOT EXISTS] <tag_name>
    (
      <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']
      [{, <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']} ...]
    )
    [TTL_DURATION = <ttl_duration>]
    [TTL_COL = <prop_name>]
    [COMMENT = '<comment>'];
```

参数	说明
`IF NOT EXISTS`	检测待创建的 Tag 是否存在，只有不存在时，才会创建 Tag。仅检测 Tag 的名称，不会检测具体属性。
`<tag_name>`	1、每个图空间内的 Tag 必须是唯一的。 2、Tag 名称设置后无法修改。 3、不能以数字开头；支持 1~4 字节的 UTF-8 编码字符，包括英文字母（区分大小写）、数字、中文等，但是不包括除下划线外的特殊字符；使用特殊字符、保留关键字或数字开头时，需要用反引号（`）包围且不能使用英文句号（`.`）。详情参见关键字和保留字。 注意：如果以中文命名 Tag，报 `SyntaxError` 错误时，需使用反引号（`）包围中文字符。
`<prop_name>`	属性名称。每个 Tag 中的属性名称必须唯一。属性的命名规则与 Tag 相同。
`<data_type>`	属性的数据类型，目前支持数值、布尔、字符串以及日期与时间。
`NULL | NOT NULL`	指定属性值是否支持为 `NULL`。默认值为 `NULL`。指定 `NOT NULL` 且没有指定 `DEFAULT` 值时，插入数据必须指定值；同时指定 `NOT NULL` 和 `DEFAULT` 值时，插入数据如果没有指定值，则默认插入 `DEFAULT` 值。
`DEFAULT`	指定属性的默认值。默认值可以是一个文字值或悦数图数据库支持的表达式。如果插入点时没有指定某个属性的值，则使用默认值。
`COMMENT`	对单个属性或 Tag 的描述。最大为 256 字节。默认无描述。
`TTL_DURATION`	指定时间戳差值，单位：秒。时间戳差值必须为 64 位非负整数。属性值和时间戳差值之和如果小于当前时间戳，属性就会过期。默认值为 `0`，表示属性永不过期。
`TTL_COL`	指定要设置存活时间的属性。属性的数据类型必须是 `int` 或者 `timestamp`。一个 Tag 只能指定一个字段为 `TTL_COL`。更多 TTL 的信息请参见 TTL。

示例

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> CREATE TAG IF NOT EXISTS player(name string, age int);

# 创建没有属性的 Tag。
nebula> CREATE TAG IF NOT EXISTS no_property();

# 创建包含默认值的 Tag。
nebula> CREATE TAG IF NOT EXISTS player_with_default(name string, age int DEFAULT 20);

# 对字段 create_time 设置 TTL 为 100 秒。
nebula> CREATE TAG IF NOT EXISTS woman(name string, age int, \
        married bool, salary double, create_time timestamp) \
        TTL_DURATION = 100, TTL_COL = "create_time";
```

创建 **Tag** 说明

尝试使用新创建的 Tag 可能会失败，因为创建是异步实现的。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。

如果需要修改心跳间隔，请为所有配置文件修改参数 `heartbeat_interval_secs` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9.2 DROP TAG

`DROP TAG` 语句可以删除当前工作空间内所有点上的指定 Tag。

点可以有一个或多个 Tag。

- 如果某个点只有一个 Tag，删除这个 Tag 后，用户就无法访问这个点，下次 Compaction 操作时会删除该点，但与该点相邻的边仍然存在——这会造成悬挂边。
- 如果某个点有多个 Tag，删除其中一个 Tag，仍然可以访问这个点，但是无法访问已删除 Tag 所定义的所有属性。

删除 Tag 操作仅删除 Schema 数据，硬盘上的文件或目录不会立刻删除，而是在下一次 Compaction 操作时删除。

> **Compatibility**
>
> 悦数图数据库 3.5.1 中默认不支持插入无 Tag 的点。如需使用无 Tag 的点，在集群内所有 Graph 服务的配置文件（`nebula-graphd.conf`）中新增 `--graph_use_vertex_key=true`；在所有 Storage 服务的配置文件（`nebula-storaged.conf`）中新增 `--use_vertex_key=true`。

前提条件

- 登录的用户必须拥有对应权限才能执行 `DROP TAG` 语句。详情请参见内置角色权限。
- 确保 Tag 不包含任何索引，否则 `DROP TAG` 时会报冲突错误 `[ERROR (-1005)]: Conflict!`。删除索引请参见 drop index。

语法

```
DROP TAG [IF EXISTS] <tag_name>;
```

- `IF EXISTS`：检测待删除的 Tag 是否存在，只有存在时，才会删除 Tag。
- `tag_name`：指定要删除的 Tag 名称。一次只能删除一个 Tag。

示例

```
nebula> CREATE TAG IF NOT EXISTS test(p1 string, p2 int);
nebula> DROP TAG test;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9.3 ALTER TAG

`ALTER TAG` 语句可以修改 Tag 的结构。例如增删属性、修改数据类型，也可以为属性设置、修改 TTL（Time-To-Live）。

注意事项

- 登录的用户必须拥有对应权限才能执行 `ALTER TAG` 语句。详情请参见内置角色权限。
- 确保要修改的属性不包含索引，否则 `ALTER TAG` 时会报冲突错误 `[ERROR (-1005)]: Conflict!` 。删除索引请参见 drop index。
- 确保新增的属性名不与已存在或被删除的属性名同名，否则新增属性会失败。

语法

```
ALTER TAG <tag_name>
    <alter_definition> [[, alter_definition] ...]
    [ttl_definition [, ttl_definition] ... ]
    [COMMENT = '<comment>'];

alter_definition:
| ADD    (prop_name data_type [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>'])
| DROP   (prop_name)
| CHANGE (prop_name data_type [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>'])

ttl_definition:
    TTL_DURATION = ttl_duration, TTL_COL = prop_name
```

- `tag_name`： 指定要修改的 Tag 名称。一次只能修改一个 Tag。请确保要修改的 Tag 在当前工作空间中存在，否则会报错。
- 可以在一个 `ALTER TAG` 语句中使用多个 `ADD` 、 `DROP` 和 `CHANGE` 子句，子句之间用英文逗号（,）分隔。
- 当使用 `ADD` 或 `CHANGE` 指定属性值为 `NOT NULL` 时，必需为该属性指定默认值，即定义 `DEFAULT` 的值。
- 当使用 `CHANGE` 修改属性的数据类型时：
- 仅允许修改 `FIXED_STRING` 和 `INT` 类型的长度为更大的长度，不允许减少长度。
- 仅允许修改 `FIXED_STRING` 类型为 `STRING` 类型、修改 `FLOAT` 类型为 `DOUBLE` 类型。

示例

```
nebula> CREATE TAG IF NOT EXISTS t1 (p1 string, p2 int);
nebula> ALTER TAG t1 ADD (p3 int32, p4 fixed_string(10));
nebula> ALTER TAG t1 TTL_DURATION = 2, TTL_COL = "p2";
nebula> ALTER TAG t1 COMMENT = 'test1';
nebula> ALTER TAG t1 ADD (p5 double NOT NULL DEFAULT 0.4 COMMENT 'p5') COMMENT='test2';
// 将 TAG t1 的 p3 属性类型从 INT32 改为 INT64，p4 属性类型从 FIXED_STRING(10) 改为 STRING。
nebula> ALTER TAG t1 CHANGE (p3 int64, p4 string);
```

修改 **Tag** 说明

尝试使用刚修改的 Tag 可能会失败，因为修改是异步实现的。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。

如果需要修改心跳间隔，请为所有配置文件修改参数 `heartbeat_interval_secs` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9.4 SHOW TAGS

`SHOW TAGS` 语句显示当前图空间内的所有 Tag 名称。

执行 `SHOW TAGS` 语句不需要任何权限，但是返回结果由登录的用户权限决定。

语法

```
SHOW TAGS;
```

示例

```
nebula> SHOW TAGS;
+----------+
| Name     |
+----------+
| "player" |
| "team"   |
+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9.5 DESCRIBE TAG

`DESCRIBE TAG` 显示指定 Tag 的详细信息，例如字段名称、数据类型等。

前提条件

登录的用户必须拥有对应权限才能执行 `DESCRIBE TAG` 语句。详情请参见内置角色权限。

语法

```
DESC[RIBE] TAG <tag_name>;
```

`DESCRIBE` 可以缩写为 `DESC` 。

示例

```
nebula> DESCRIBE TAG player;
+--------+----------+-------+---------+---------+
| Field  | Type     | Null  | Default | Comment |
+--------+----------+-------+---------+---------+
| "name" | "string" | "YES" |         |         |
| "age"  | "int64"  | "YES" |         |         |
+--------+----------+-------+---------+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9.6 DELETE TAG

`DELETE TAG` 语句可以删除指定点上的指定 Tag。

前提条件

登录的用户必须拥有对应权限才能执行 `DELETE TAG` 语句。详情请参见内置角色权限。

语法

```
DELETE TAG <tag_name_list> FROM <VID>;
```

- `tag_name_list`：指定 Tag 名称。多个 Tag 用英文逗号（,）分隔，也可以用 `*` 表示所有 Tag。
- `VID`：指定要删除 Tag 的点 ID。

示例

```
nebula> CREATE TAG IF NOT EXISTS test1(p1 string, p2 int);
nebula> CREATE TAG IF NOT EXISTS test2(p3 string, p4 int);
nebula> INSERT VERTEX test1(p1, p2),test2(p3, p4) VALUES "test":("123", 1, "456", 2);
nebula> FETCH PROP ON * "test" YIELD vertex AS v;
+-------------------------------------------------------+
| v                                                     |
+-------------------------------------------------------+
| ("test" :test1{p1: "123", p2: 1} :test2{p3: "456", p4: 2}) |
+-------------------------------------------------------+

nebula> DELETE TAG test1 FROM "test";
nebula> FETCH PROP ON * "test" YIELD vertex AS v;
+----------------------------------+
| v                                |
+----------------------------------+
| ("test" :test2{p3: "456", p4: 2}) |
+----------------------------------+

nebula> DELETE TAG * FROM "test";
nebula> FETCH PROP ON * "test" YIELD vertex AS v;
+---+
| v |
+---+
+---+
```

Compatibility

- 在 openCypher 中，可以使用 `REMOVE v:LABEL` 语句来移除该点 `v` 的 `LABEL`。
- 相同语意，但不同语法。在 nGQL 中使用 `DELETE TAG`。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.9.7 增加和删除标签

在 openCypher 中，有增加标签（`SET label`）和移除标签（`REMOVE label`）的功能，可以用于加速查询或者标记过程。

在悦数图数据库中，可以通过 Tag 变相实现相同操作，创建 Tag 并将 Tag 插入到已有的点上，就可以根据 Tag 名称快速查找点，也可以通过 `DELETE TAG` 删除某些点上不再需要的 Tag。

示例

例如在 basketballplayer 数据集中，部分篮球运动员同时也是球队股东，可以为股东 Tag `shareholder` 创建索引，方便快速查找。如果不再是股东，可以通过 `DELETE TAG` 语句删除相应运动员的股东 Tag。

```
//创建股东 Tag 和索引
nebula> CREATE TAG IF NOT EXISTS shareholder();
nebula> CREATE TAG INDEX IF NOT EXISTS shareholder_tag on shareholder();

//为点添加 Tag
nebula> INSERT VERTEX shareholder() VALUES "player100":();
nebula> INSERT VERTEX shareholder() VALUES "player101":();

//快速查询所有股东
nebula> MATCH (v:shareholder) RETURN v;
+-------------------------------------------------------------------+
| v                                                                 |
+-------------------------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"} :shareholder{})  |
| ("player101" :player{age: 36, name: "Tony Parker"} :shareholder{}) |
+-------------------------------------------------------------------+
nebula> LOOKUP ON shareholder YIELD id(vertex);
+-------------+
| id(VERTEX)  |
+-------------+
| "player100" |
| "player101" |
+-------------+

//如果 player100 不再是股东
nebula> DELETE TAG shareholder FROM "player100";
nebula> LOOKUP ON shareholder YIELD id(vertex);
+-------------+
| id(VERTEX)  |
+-------------+
| "player101" |
+-------------+
```

Note

如果插入测试数据后才创建索引，请用 `REBUILD TAG INDEX <index_name_list>;` 语句重建索引。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.10 Edge type 语句

5.10.1 CREATE EDGE

`CREATE EDGE` 语句可以通过指定名称创建一个 Edge type。

OpenCypher 兼容性

nGQL 中的 Edge type 和 openCypher 中的关系类型相似，但又有所不同，例如它们的创建方式。

- openCypher 中的关系类型需要在 `CREATE` 语句中与点一起创建。
- nGQL 中的 Edge type 需要使用 `CREATE EDGE` 语句独立创建。Edge type 更像是 MySQL 中的表。

前提条件

执行 `CREATE EDGE` 语句需要当前登录的用户拥有指定图空间的创建 Edge type 权限，否则会报错。

语法

创建 Edge type 前，需要先用 `USE` 语句指定工作空间。

```
CREATE EDGE [IF NOT EXISTS] <edge_type_name>
    (
      <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']
      [{, <prop_name> <data_type> [NULL | NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']} ...]
    )
    [TTL_DURATION = <ttl_duration>]
    [TTL_COL = <prop_name>]
    [COMMENT = '<comment>'];
```

参数	说明
`IF NOT EXISTS`	检测待创建的 Edge type 是否存在，只有不存在时，才会创建 Edge type。仅检测 Edge type 的名称，不会检测具体属性。
`<edge_type_name>`	1、每个图空间内的 Edge type 必须是唯一的。 2、Edge type 名称设置后无法修改。 3、不能以数字开头；支持 1~4 字节的 UTF-8 编码字符，包括英文字母（区分大小写）、数字、中文等，但是不包括除下划线外的特殊字符；使用特殊字符、保留关键字或数字开头时，需要用反引号（`）包围且不能使用英文句号（`.`）。详情参见关键字和保留字。 注意：如果以中文命名 Edge type，报 `SyntaxError` 错误时，需使用反引号（`）包围中文字符。
`<prop_name>`	属性名称。每个 Edge type 中的属性名称必须唯一。属性的命名规则与 Edge type 相同。
`<data_type>`	属性的数据类型，目前支持数值、布尔、字符串以及日期与时间。
`NULL | NOT NULL`	指定属性值是否支持为 `NULL`。默认值为 `NULL`。当指定属性值为 `NOT NULL` 时，必需指定属性的默认值，也就是 `DEFAULT` 的值。
`DEFAULT`	指定属性的默认值。默认值可以是一个文字值或悦数图数据库支持的表达式。如果插入边时没有指定某个属性的值，则使用默认值。
`COMMENT`	对单个属性或 Edge type 的描述。最大为 256 字节。默认无描述。
`TTL_DURATION`	指定时间戳差值，单位：秒。时间戳差值必须为 64 位非负整数。属性值和时间戳差值之和如果小于当前时间戳，属性就会过期。默认值为 `0`，表示属性永不过期。
`TTL_COL`	指定要设置存活时间的属性。属性的数据类型必须是 `int` 或者 `timestamp`。一个 Edge type 只能指定一个字段为 `TTL_COL`。更多 TTL 的信息请参见 TTL。

示例

```
nebula> CREATE EDGE IF NOT EXISTS follow(degree int);
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# 创建没有属性的 Edge type。
nebula> CREATE EDGE IF NOT EXISTS no_property();

# 创建包含默认值的 Edge type。
nebula> CREATE EDGE IF NOT EXISTS follow_with_default(degree int DEFAULT 20);

# 对字段 p2 设置 TTL 为 100 秒。
nebula> CREATE EDGE IF NOT EXISTS e1(p1 string, p2 int, p3 timestamp) \
        TTL_DURATION = 100, TTL_COL = "p2";
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.10.2 DROP EDGE

`DROP EDGE` 语句可以删除当前工作空间内的指定 Edge type。

一个边只能有一个 Edge type，删除这个 Edge type 后，用户就无法访问这个边，下次 Compaction 操作时会删除该边。

删除 Edge type 操作仅删除 Schema 数据，硬盘上的文件或目录不会立刻删除，而是在下一次 Compaction 操作时删除。

前提条件

- 登录的用户必须拥有对应权限才能执行 `DROP EDGE` 语句。详情请参见内置角色权限。
- 确保 Edge type 不包含任何索引，否则 `DROP EDGE` 时会报冲突错误 `[ERROR (-1005)]: Conflict!` 。删除索引请参见 drop index。

语法

```
DROP EDGE [IF EXISTS] <edge_type_name>
```

- `IF EXISTS`： 检测待删除的 Edge type 是否存在，只有存在时，才会删除 Edge type。
- `edge_type_name`： 指定要删除的 Edge type 名称。一次只能删除一个 Edge type。

示例

```
nebula> CREATE EDGE IF NOT EXISTS e1(p1 string, p2 int);
nebula> DROP EDGE e1;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.10.3 ALTER EDGE

`ALTER EDGE` 语句可以修改 Edge type 的结构。例如增删属性、修改数据类型，也可以为属性设置、修改 TTL（Time-To-Live）。

注意事项

- 登录的用户必须拥有对应权限才能执行 `ALTER EDGE` 语句。详情请参见内置角色权限。
- 确保要修改的属性不包含索引，否则 `ALTER EDGE` 时会报冲突错误 `[ERROR (-1005)]: Conflict!`。删除索引请参见 drop index。
- 确保新增的属性名不与已存在或被删除的属性名同名，否则新增属性会失败。
- 允许增加 `FIXED_STRING` 和 `INT` 类型的长度。
- 允许 `FIXED_STRING` 类型转换为 `STRING` 类型、 `FLOAT` 类型转换为 `DOUBLE` 类型。

语法

```
ALTER EDGE <edge_type_name>
    <alter_definition> [, alter_definition] ...]
    [ttl_definition [, ttl_definition] ... ]
    [COMMENT = '<comment>'];

alter_definition:
| ADD    (prop_name data_type)
| DROP   (prop_name)
| CHANGE (prop_name data_type)

ttl_definition:
    TTL_DURATION = ttl_duration, TTL_COL = prop_name
```

- `edge_type_name`： 指定要修改的 Edge type 名称。一次只能修改一个 Edge type。请确保要修改的 Edge type 在当前工作空间中存在，否则会报错。
- 可以在一个 `ALTER EDGE` 语句中使用多个 `ADD` 、 `DROP` 和 `CHANGE` 子句，子句之间用英文逗号（`,`）分隔。
- 当使用 `ADD` 或 `CHANGE` 指定属性值为 `NOT NULL` 时，必需为该属性指定默认值，即定义 `DEFAULT` 的值。

示例

```
nebula> CREATE EDGE IF NOT EXISTS e1(p1 string, p2 int);
nebula> ALTER EDGE e1 ADD (p3 int, p4 string);
nebula> ALTER EDGE e1 TTL_DURATION = 2, TTL_COL = "p2";
nebula> ALTER EDGE e1 COMMENT = 'edge1';
```

修改 **Edge type** 说明

尝试使用刚修改的 Edge type 可能会失败，因为修改是异步实现的。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。

如果需要修改心跳间隔，请为所有配置文件修改参数 `heartbeat_interval_secs`。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.10.4 SHOW EDGES

`SHOW EDGES` 语句显示当前图空间内的所有 Edge type 名称。

执行 `SHOW EDGES` 语句不需要任何权限，但是返回结果由登录的用户权限 决定。

语法

```
SHOW EDGES;
```

示例

```
nebula> SHOW EDGES;
+----------+
| Name     |
+----------+
| "follow" |
| "serve"  |
+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.10.5 DESCRIBE EDGE

`DESCRIBE EDGE` 显示指定 Edge type 的详细信息，例如字段名称、数据类型等。

前提条件

登录的用户必须拥有对应权限才能执行 `DESCRIBE EDGE` 语句。详情请参见内置角色权限。

语法

```
DESC[RIBE] EDGE <edge_type_name>
```

`DESCRIBE` 可以缩写为 `DESC` 。

示例

```
nebula> DESCRIBE EDGE follow;
+----------+---------+-------+---------+---------+
| Field    | Type    | Null  | Default | Comment |
+----------+---------+-------+---------+---------+
| "degree" | "int64" | "YES" |         |         |
+----------+---------+-------+---------+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.11 点语句

5.11.1 INSERT VERTEX

`INSERT VERTEX` 语句可以在悦数图数据库实例的指定图空间中插入一个或多个点。

前提条件

执行 `INSERT VERTEX` 语句需要当前登录的用户拥有指定图空间的插入点权限，否则会报错。

语法

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES VID: ([prop_value_list])

tag_props:
  tag_name ([prop_name_list])

prop_name_list:
   [prop_name [, prop_name] ...]

prop_value_list:
   [prop_value [, prop_value] ...]
```

- `IF NOT EXISTS`： 检测待插入的 VID 是否存在，只有不存在时，才会插入，如果已经存在，不会进行修改。

  Note

  - `IF NOT EXISTS` 仅检测 VID + Tag 的值是否相同，不会检测属性值。
  - `IF NOT EXISTS` 会先读取一次数据是否存在，因此对性能会有明显影响。

- `tag_name`： 点关联的 Tag（点类型）。Tag 的创建，详情请参见 CREATE TAG。

  Caution

  悦数图数据库 3.5.1 中默认不支持插入无 Tag 的点。如需使用无 Tag 的点，在集群内所有 Graph 服务的配置文件（`nebula-graphd.conf`）中新增 `--graph_use_vertex_key=true`；在所有 Storage 服务的配置文件（`nebula-storaged.conf`）中新增 `--use_vertex_key=true`。插入无 Tag 点的命令示例如 `INSERT VERTEX VALUES "1":();`。

- `property_name`： 需要设置的属性名称。
- `vid`： 点 ID。在悦数图数据库 3.5.1 中支持字符串和整数，需要在创建图空间时设置，详情请参见 CREATE SPACE。
- `property_value`： 根据 `prop_name_list` 填写属性值。如果没有填写属性值，而 Tag 中对应的属性设置为 `NOT NULL`，会返回错误。详情请参见 CREATE TAG。

Caution

`INSERT VERTEX` 与 openCypher 中 `CREATE` 的语意不同：

- `INSERT VERTEX` 语意更接近于 NoSQL(key-value) 方式的 INSERT 语意，或者 SQL 中的 `UPSERT`（`UPDATE` or `INSERT`）。
- 相同 `VID` 和 `TAG` 的情况下，如果没有使用 `IF NOT EXISTS`，新写入的数据会覆盖旧数据，不存在时会新写入。
- 相同 `VID` 但不同 `TAG` 的情况下，不同 TAG 对应的记录不会相互覆盖，不存在会新写入。

参考以下示例。

Copyright © Yueshu Graph Technology - All rights reserved.

示例

```
# 插入不包含属性的点。
nebula> CREATE TAG IF NOT EXISTS t1();
nebula> INSERT VERTEX t1() VALUES "10":();
```

```
nebula> CREATE TAG IF NOT EXISTS t2 (name string, age int);
nebula> INSERT VERTEX t2 (name, age) VALUES "11":("n1", 12);

# 创建失败，因为"a13"不是 int 类型。
nebula> INSERT VERTEX t2 (name, age) VALUES "12":("n1", "a13");

# 一次插入 2 个点。
nebula> INSERT VERTEX t2 (name, age) VALUES "13":("n3", 12), "14":("n4", 8);
```

```
nebula> CREATE TAG IF NOT EXISTS t3(p1 int);
nebula> CREATE TAG IF NOT EXISTS t4(p2 string);

# 一次插入两个 Tag 的属性到同一个点。
nebula> INSERT VERTEX  t3 (p1), t4(p2) VALUES "21": (321, "hello");
```

一个点可以多次插入属性值，以最后一次为准。

```
# 多次插入属性值。
nebula> INSERT VERTEX t2 (name, age) VALUES "11":("n2", 13);
nebula> INSERT VERTEX t2 (name, age) VALUES "11":("n3", 14);
nebula> INSERT VERTEX t2 (name, age) VALUES "11":("n4", 15);
nebula> FETCH PROP ON t2 "11" YIELD properties(vertex);
+-----------------------+
| properties(VERTEX)    |
+-----------------------+
| {age: 15, name: "n4"} |
+-----------------------+
```

```
nebula> CREATE TAG IF NOT EXISTS t5(p1 fixed_string(5) NOT NULL, p2 int, p3 int DEFAULT NULL);
nebula> INSERT VERTEX t5(p1, p2, p3) VALUES "001":("Abe", 2, 3);

# 插入失败，因为属性 p1 不能为 NULL。
nebula> INSERT VERTEX t5(p1, p2, p3) VALUES "002":(NULL, 4, 5);
[ERROR (-1009)]: SemanticError: No schema found for `t5'

# 属性 p3 为默认值 NULL。
nebula> INSERT VERTEX t5(p1, p2) VALUES "003":("cd", 5);
nebula> FETCH PROP ON t5 "003" YIELD properties(vertex);
+---------------------------------+
| properties(VERTEX)              |
+---------------------------------+
| {p1: "cd", p2: 5, p3: __NULL__} |
+---------------------------------+

# 属性 p1 最大长度为 5，因此会被截断。
nebula> INSERT VERTEX t5(p1, p2) VALUES "004":("shalalalala", 4);
nebula> FETCH PROP on t5 "004" YIELD properties(vertex);
+-----------------------------------+
| properties(VERTEX)                |
+-----------------------------------+
| {p1: "shala", p2: 4, p3: __NULL__} |
+-----------------------------------+
```

使用 `IF NOT EXISTS` 插入已存在的点时，不会进行修改。

```
# 插入点 1。
nebula> INSERT VERTEX t2 (name, age) VALUES "1":("n2", 13);

# 使用 IF NOT EXISTS 修改点 1，因为点 1 已存在，不会进行修改。
nebula> INSERT VERTEX IF NOT EXISTS t2 (name, age) VALUES "1":("n3", 14);
nebula> FETCH PROP ON t2 "1" YIELD properties(vertex);
+-----------------------+
| properties(VERTEX)    |
+-----------------------+
| {age: 13, name: "n2"} |
+-----------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.11.2 DELETE VERTEX

`DELETE VERTEX` 语句可以删除点，但是默认不删除该点关联的出边和入边。

Compatibility

悦数图数库 2.x 默认删除点及关联该点的出边和入边，悦数图数库 3.5.1 默认只删除点，不删除该点关联的出边和入边，此时将默认存在悬挂边。

`DELETE VERTEX` 语句一次可以删除一个或多个点。用户可以结合管道符一起使用，详情请参见管道符。

Note

- `DELETE VERTEX` 是直接删除点，不删除关联的边。
- `DELETE TAG` 是删除指定点上的指定 Tag。

语法

```
DELETE VERTEX <vid> [ , <vid> ... ] [WITH EDGE];
```

- `WITH EDGE`：删除该点关联的出边和入边。

示例

```
# 删除 VID 为 `team1` 的点，不删除该点关联的出边和入边。
nebula> DELETE VERTEX "team1";

# 删除 VID 为 `team1` 的点，并删除该点关联的出边和入边。
nebula> DELETE VERTEX "team1" WITH EDGE;
```

```
# 结合管道符，删除符合条件的点。
nebula> GO FROM "player100" OVER serve WHERE properties(edge).start_year == "2021" YIELD dst(edge) AS id | DELETE VERTEX $-.id;
```

删除过程

悦数图数库找到目标点并删除，该目标点的所有邻边（出边和入边）将成为悬挂边。

Caution

- 不支持原子性删除，如果发生错误请重试，避免出现部分删除的情况。否则会导致悬挂边。
- 删除超级节点耗时较多，为避免删除完成前连接超时，可以调整 `nebula-graphd.conf` 中的参数 `--storage_client_timeout_ms` 延长超时时间。

悬挂边视频

- 悦数图数库 的悬挂边小科普（2 分 28 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.11.3 UPDATE VERTEX

`UPDATE VERTEX` 语句可以修改点上 Tag 的属性值。

悦数图数图数据库支持 CAS（compare and set）操作。

> **Note**
>
> 一次只能修改一个 Tag。

语法

```
UPDATE VERTEX ON <tag_name> <vid>
SET <update_prop>
[WHEN <condition>]
[YIELD <output>]
```

参数	是否必须	说明	示例
`ON <tag_name>`	是	指定点的 Tag。要修改的属性必须在这个 Tag 内。	`ON player`
`<vid>`	是	指定要修改的点 ID。	`"player100"`
`SET <update_prop>`	是	指定如何修改属性值。	`SET age = age +1`
`WHEN <condition>`	否	指定过滤条件。如果 `<condition>` 结果为 `false`， `SET` 子句不会生效。	`WHEN name == "Tim"`
`YIELD <output>`	否	指定语句的输出格式。	`YIELD name AS Name`

示例

```
// 查看点"player101"的属性。
nebula> FETCH PROP ON player "player101" YIELD properties(vertex);
+-------------------------------+
| properties(VERTEX)            |
+-------------------------------+
| {age: 36, name: "Tony Parker"} |
+-------------------------------+

// 修改属性 age 的值，并返回 name 和新的 age。
nebula> UPDATE VERTEX ON player "player101" \
        SET age = age + 2 \
        WHEN name == "Tony Parker" \
        YIELD name AS Name, age AS Age;
+---------------+-----+
| Name          | Age |
+---------------+-----+
| "Tony Parker" | 38  |
+---------------+-----+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.11.4 UPSERT VERTEX

`UPSERT VERTEX` 语句结合 `UPDATE` 和 `INSERT`，如果点存在，会修改点的属性值；如果点不存在，会插入新的点。

> **Note**
>
> `UPSERT VERTEX` 一次只能修改一个 Tag。

`UPSERT VERTEX` 性能远低于 `INSERT`，因为 `UPSERT` 是一组分片级别的读取、修改、写入操作。

> **Caution**
>
> 并发 `UPSERT` 同一个 TAG 或 EDGE TYPE 会报错。

语法

```
UPSERT VERTEX ON <tag> <vid>
SET <update_prop>
[WHEN <condition>]
[YIELD <output>]
```

参数	是否必须	说明	示例
`ON <tag>`	是	指定点的 Tag。要修改的属性必须在这个 Tag 内。	`ON player`
`<vid>`	是	指定要修改或插入的点 ID。	`"player100"`
`SET <update_prop>`	是	指定如何修改属性值。	`SET age = age +1`
`WHEN <condition>`	否	指定过滤条件。	`WHEN name == "Tim"`
`YIELD <output>`	否	指定语句的输出格式。	`YIELD name AS Name`

插入不存在的点

如果点不存在，无论 `WHEN` 子句的条件是否满足，都会插入点，同时执行 `SET` 子句，因此新插入的点的属性值取决于：

- `SET` 子句。
- 属性是否有默认值。

例如：

- 要插入的点包含基于 Tag `player` 的属性 `name` 和 `age`。
- `SET` 子句指定 `age=30`。

不同情况下的属性值如下表。

是否满足 **`WHEN`** 子句条件	属性是否有默认值	**`name`** 属性值	**`age`** 属性值
是	是	默认值	`30`
是	否	`NULL`	`30`
否	是	默认值	`30`
否	否	`NULL`	`30`

示例如下：

Copyright © Yueshu Graph Technology - All rights reserved.

```
// 查看三个点是否存在，结果 “Empty set” 表示顶点不存在。
nebula> FETCH PROP ON * "player666", "player667", "player668" YIELD properties(vertex);
+--------------------+
| properties(VERTEX) |
+--------------------+
+--------------------+
Empty set

nebula> UPSERT VERTEX ON player "player666" \
        SET age = 30 \
        WHEN name == "Joe" \
        YIELD name AS Name, age AS Age;
+----------+----------+
| Name     | Age      |
+----------+----------+
| __NULL__ | 30       |
+----------+----------+

nebula> UPSERT VERTEX ON player "player666" \
        SET age = 31 \
        WHEN name == "Joe" \
        YIELD name AS Name, age AS Age;
+----------+-----+
| Name     | Age |
+----------+-----+
| __NULL__ | 30  |
+----------+-----+

nebula> UPSERT VERTEX ON player "player667" \
        SET age = 31 \
        YIELD name AS Name, age AS Age;
+----------+-----+
| Name     | Age |
+----------+-----+
| __NULL__ | 31  |
+----------+-----+

nebula> UPSERT VERTEX ON player "player668" \
        SET name = "Amber", age = age + 1 \
        YIELD name AS Name, age AS Age;
+---------+----------+
| Name    | Age      |
+---------+----------+
| "Amber" | __NULL__ |
+---------+----------+
```

上面最后一个示例中，因为 `age` 没有默认值，插入点时， `age` 默认值为 `NULL` ，执行 `age = age + 1` 后仍为 `NULL` 。如果 `age` 有默认值，则 `age = age + 1` 可以正常执行，例如：

```
nebula> CREATE TAG IF NOT EXISTS player_with_default(name string, age int DEFAULT 20);
Execution succeeded

nebula> UPSERT VERTEX ON player_with_default "player101" \
        SET age = age + 1 \
        YIELD name AS Name, age AS Age;

+----------+-----+
| Name     | Age |
+----------+-----+
| __NULL__ | 21  |
+----------+-----+
```

修改存在的点

如果点存在，且满足 `WHEN` 子句的条件，就会修改点的属性值。

```
nebula> FETCH PROP ON player "player101" YIELD properties(vertex);
+--------------------------------+
| properties(VERTEX)             |
+--------------------------------+
| {age: 36, name: "Tony Parker"} |
+--------------------------------+

nebula> UPSERT VERTEX ON player "player101" \
        SET age = age + 2 \
        WHEN name == "Tony Parker" \
        YIELD name AS Name, age AS Age;
+---------------+-----+
| Name          | Age |
+---------------+-----+
| "Tony Parker" | 38  |
+---------------+-----+
```

如果点存在，但是不满足 `WHEN` 子句的条件，修改不会生效。

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> FETCH PROP ON player "player101" YIELD properties(vertex);
+--------------------------------+
| properties(VERTEX)             |
+--------------------------------+
| {age: 38, name: "Tony Parker"} |
+--------------------------------+

nebula> UPSERT VERTEX ON player "player101" \
        SET age = age + 2 \
        WHEN name == "Someone else" \
        YIELD name AS Name, age AS Age;
+---------------+-----+
| Name          | Age |
+---------------+-----+
| "Tony Parker" | 38  |
+---------------+-----+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.12 边语句

5.12.1 INSERT EDGE

`INSERT EDGE` 语句可以在悦数图数据库实例的指定图空间中插入一条或多条边。边是有方向的，从起始点（src_vid）到目的点（dst_vid）。

`INSERT EDGE` 的执行方式为覆盖式插入。如果已有 Edge type、起点、终点、rank 都相同的边，则覆盖原边。

语法

```
INSERT EDGE [IF NOT EXISTS] <edge_type> ( <prop_name_list> ) VALUES
<src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> )
[, <src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> ), ...];

<prop_name_list> ::=
  [ <prop_name> [, <prop_name> ] ...]

<prop_value_list> ::=
  [ <prop_value> [, <prop_value> ] ...]
```

- `IF NOT EXISTS`：用户可以使用 `IF NOT EXISTS` 关键字检测待插入的边是否存在，只有不存在时，才会插入。

  Note

  - `IF NOT EXISTS` 仅检测<边的类型、起始点、目的点和 rank>是否存在，不会检测属性值是否重合。
  - `IF NOT EXISTS` 会先读取一次数据是否存在，因此对性能会有明显影响。

- `<edge_type>`：边关联的 Edge type，只能指定一个 Edge type。Edge type 必须提前创建，详情请参见 CREATE EDGE。
- `<prop_name_list>`：需要设置的属性名称列表。
- `src_vid`：起始点 ID，表示边的起点。
- `dst_vid`：目的点 ID，表示边的终点。
- `rank`：可选项。边的 rank 值。数据类型为 `int`。默认值为 `0`。

  openCypher 兼容性

  openCypher 中没有 rank 的概念。

- `<prop_value_list>`：根据 `prop_name_list` 填写属性值。如果属性值和 Edge type 中的数据类型不匹配，会返回错误。如果没有填写属性值，而 Edge type 中对应的属性设置为 `NOT NULL`，也会返回错误。详情请参见 CREATE EDGE。

示例

```
# 插入不包含属性的边。
nebula> CREATE EDGE IF NOT EXISTS e1();
nebula> INSERT EDGE e1 () VALUES "10"->"11":();

# 插入 rank 为 1 的边。
nebula> INSERT EDGE e1 () VALUES "10"->"11"@1:();
```

```
nebula> CREATE EDGE IF NOT EXISTS e2 (name string, age int);
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", 1);

# 一次插入 2 条边。
nebula> INSERT EDGE e2 (name, age) VALUES \
        "12"->"13":("n1", 1), "13"->"14":("n2", 2);
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# 创建失败，因为"a13"不是 int 类型。
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", "a13");
```

一条边可以多次插入属性值，以最后一次为准。

```
# 多次插入属性值。
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", 12);
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", 13);
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", 14);
nebula> FETCH PROP ON e2 "11"->"13" YIELD edge AS e;
+-----------------------------------------+
| e                                       |
+-----------------------------------------+
| [:e2 "11"->"13" @0 {age: 14, name: "n1"}] |
+-----------------------------------------+
```

使用 `IF NOT EXISTS` 插入已存在的边时，不会进行修改。

```
# 插入边。
nebula> INSERT EDGE e2 (name, age) VALUES "14"->"15"@1:("n1", 12);
# 使用 IF NOT EXISTS 修改边，因为边已存在，不会进行修改。
nebula> INSERT EDGE IF NOT EXISTS e2 (name, age) VALUES "14"->"15"@1:("n2", 13);
nebula> FETCH PROP ON e2 "14"->"15"@1 YIELD edge AS e;
+-----------------------------------------+
| e                                       |
+-----------------------------------------+
| [:e2 "14"->"15" @1 {age: 12, name: "n1"}] |
+-----------------------------------------+
```

Note

- 悦数图数据库 3.5.1 允许存在悬挂边（Dangling edge）。因此可以在起点或者终点存在前，先写边；此时就可以通过 `<edgetype>._src` 或 `<edgetype>._dst` 获取到（尚未写入的）点 VID（不建议这样使用）。
- 目前还不能保证操作的原子性，如果失败请重试，否则会发生部分写入。此时读取该数据的行为是未定义的。例如写入操作涉及到多个机器时，可能会出现插入单个边的正反向边只写入成功一个，或者插入多个边时只写入成功一部分，此时会返回报错，请重新执行命令。
- 并发写入同一条边会报 `edge conflict` 错误，可稍后重试。
- `边的 INSERT 速度` 大约是点的 INSERT 速度一半。原因是 INSERT 边会对应 storaged 的两个 INSERT，`INSERT 点` 对应 storaged 的一个 INSERT。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.12.2 DELETE EDGE

`DELETE EDGE` 语句可以删除边。一次可以删除一条或多条边。用户可以结合管道符一起使用，详情请参见管道符。

如果需要删除一个点的所有出边，请删除这个点。详情请参见 DELETE VERTEX。

语法

```
DELETE EDGE <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid>[@<rank>] ...]
```

Caution

如果不指定 rank，则仅仅删除 rank 为 0 的边。需要删除所有的 rank，见如下示例。

示例

```
nebula> DELETE EDGE serve "player100" -> "team204"@0;
```

```
# 结合管道符，删除两点之间同类型的所有rank的边。
nebula> GO FROM "player100" OVER follow \
        WHERE dst(edge) == "player101" \
        YIELD src(edge) AS src, dst(edge) AS dst, rank(edge) AS rank \
        | DELETE EDGE follow $-.src -> $-.dst @ $-.rank;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.12.3 UPDATE EDGE

`UPDATE EDGE` 语句可以修改边上 Edge type 的属性。

悦数图数据库支持 CAS（compare and swap）操作。

语法

```
UPDATE EDGE ON <edge_type>
<src_vid> -> <dst_vid> [@<rank>]
SET <update_prop>
[WHEN <condition>]
[YIELD <output>]
```

参数	是否必须	说明	示例
`ON <edge_type>`	是	指定 Edge type。要修改的属性必须在这个 Edge type 内。	`ON serve`
`<src_vid>`	是	指定边的起始点 ID。	`"player100"`
`<dst_vid>`	是	指定边的目的点 ID。	`"team204"`
`<rank>`	否	指定边的 rank 值。	`10`
`SET <update_prop>`	是	指定如何修改属性值。	`SET start_year = start_year +1`
`WHEN <condition>`	否	指定过滤条件。如果 `<condition>` 结果为 `false`， `SET` 子句不会生效。	`WHEN end_year < 2010`
`YIELD <output>`	否	指定语句的输出格式。	`YIELD start_year AS Start_Year`

示例

```
// 用 GO 语句查看边的属性值。
nebula> GO FROM "player100" \
        OVER serve \
        YIELD properties(edge).start_year, properties(edge).end_year;
+----------------------------+---------------------------+
| properties(EDGE).start_year | properties(EDGE).end_year |
+----------------------------+---------------------------+
| 1997                       | 2016                      |
+----------------------------+---------------------------+

// 修改属性 start_year 的值，并返回 end_year 和新的 start_year。

nebula> UPDATE EDGE ON serve "player100" -> "team204"@0 \
        SET start_year = start_year + 1 \
        WHEN end_year > 2010 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| 1998       | 2016     |
+------------+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.12.4 UPSERT EDGE

`UPSERT EDGE` 语句结合 `UPDATE` 和 `INSERT`，如果边存在，会更新边的属性；如果边不存在，会插入新的边。

`UPSERT EDGE` 性能远低于 `INSERT`，因为 `UPSERT` 是一组分片级别的读取、修改、写入操作。

Caution

并发 `UPSERT` 同一个 TAG 或 EDGE TYPE 会报错。

语法

```
UPSERT EDGE ON <edge_type>
<src_vid> -> <dst_vid> [@rank]
SET <update_prop>
[WHEN <condition>]
[YIELD <properties>]
```

参数	是否必须	说明	示例
`ON <edge_type>`	是	指定 Edge type。要修改的属性必须在这个 Edge type 内。	`ON serve`
`<src_vid>`	是	指定边的起始点 ID。	`"player100"`
`<dst_vid>`	是	指定边的目的点 ID。	`"team204"`
`<rank>`	否	指定边的 rank 值。数据类型为 `int`。	`10`
`SET <update_prop>`	是	指定如何修改属性值。	`SET start_year = start_year +1`
`WHEN <condition>`	否	指定过滤条件。	`WHEN end_year < 2010`
`YIELD <output>`	否	指定语句的输出格式。	`YIELD start_year AS Start_Year`

插入不存在的边

如果边不存在，无论 `WHEN` 子句的条件是否满足，都会插入边，同时执行 `SET` 子句，因此新插入的边的属性值取决于：

- `SET` 子句。
- 属性是否有默认值。

例如：

- 要插入的边包含基于 Edge type `serve` 的属性 `start_year` 和 `end_year`。
- `SET` 子句指定 `end_year = 2021`。

不同情况下的属性值如下表。

是否满足 **`WHEN`** 子句条件	属性是否有默认值	**`start_year`** 属性值	**`end_year`** 属性值
是	是	默认值	`2021`
是	否	`NULL`	`2021`
否	是	默认值	`2021`
否	否	`NULL`	`2021`

Copyright © Yueshu Graph Technology - All rights reserved.

示例如下：

```
// 查看如下三个点是否有 serve 类型的出边，结果 “Empty set” 表示没有 serve 类型的出边。
nebula> GO FROM "player666", "player667", "player668" \
        OVER serve \
        YIELD properties(edge).start_year, properties(edge).end_year;
+----------------------------+--------------------------+
| properties(EDGE).start_year | properties(EDGE).end_year |
+----------------------------+--------------------------+
+----------------------------+--------------------------+
Empty set

nebula> UPSERT EDGE on serve \
        "player666" -> "team200"@0 \
        SET end_year = 2021 \
        WHEN end_year == 2010 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| __NULL__   | 2021     |
+------------+----------+

nebula> UPSERT EDGE on serve \
        "player666" -> "team200"@0 \
        SET end_year = 2022 \
        WHEN end_year == 2010 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| __NULL__   | 2021     |
+------------+----------+

nebula> UPSERT EDGE on serve \
        "player667" -> "team200"@0 \
        SET end_year = 2022 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| __NULL__   | 2022     |
+------------+----------+

nebula> UPSERT EDGE on serve \
        "player668" -> "team200"@0 \
        SET start_year = 2000, end_year = end_year + 1 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| 2000       | __NULL__ |
+------------+----------+
```

上面最后一个示例中，因为 `end_year` 没有默认值，插入边时， `end_year` 默认值为 `NULL` ，执行 `end_year = end_year + 1` 后仍为 `NULL` 。如果 `end_year` 有默认值，则 `end_year = end_year + 1` 可以正常执行，例如：

```
nebula> CREATE EDGE IF NOT EXISTS serve_with_default(start_year int, end_year int DEFAULT 2010);
Execution succeeded

nebula> UPSERT EDGE on serve_with_default \
        "player668" -> "team200" \
        SET end_year = end_year + 1 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| __NULL__   | 2011     |
+------------+----------+
```

修改存在的边

如果边存在，且满足 `WHEN` 子句的条件，就会修改边的属性值。

```
nebula> MATCH (v:player{name:"Ben Simmons"})-[e:serve]-(v2) \
        RETURN e;
+----------------------------------------------------------------------+
| e                                                                    |
+----------------------------------------------------------------------+
| [:serve "player149"->"team219" @0 {end_year: 2019, start_year: 2016}] |
+----------------------------------------------------------------------+

nebula> UPSERT EDGE on serve \
        "player149" -> "team219" \
        SET end_year = end_year + 1 \
        WHEN start_year == 2016 \
        YIELD start_year, end_year;
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+------------+----------+
| start_year | end_year |
+------------+----------+
| 2016       | 2020     |
+------------+----------+
```

如果边存在，但是不满足 `WHEN` 子句的条件，修改不会生效。

```
nebula> MATCH (v:player{name:"Ben Simmons"})-[e:serve]-(v2) \
        RETURN e;
+----------------------------------------------------------------------+
| e                                                                    |
+----------------------------------------------------------------------+
| [:serve "player149"->"team219" @0 {end_year: 2020, start_year: 2016}] |
+----------------------------------------------------------------------+

nebula> UPSERT EDGE on serve \
        "player149" -> "team219" \
        SET end_year = end_year + 1 \
        WHEN start_year != 2016 \
        YIELD start_year, end_year;
+------------+----------+
| start_year | end_year |
+------------+----------+
| 2016       | 2020     |
+------------+----------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13 原生索引

5.13.1 索引介绍

为了提高查询性能，悦数图数据库支持为点的 Tag 或 Tag 的某个属性，边的 Edge type 或 Edge type 的某个属性创建索引。索引可以基于指定的 Tag、Edge type、属性查询数据，但是索引本身不存储数据，而是存储数据的位置。

悦数图数据库支持两种类型索引：原生索引和全文索引。

使用说明

- 索引可以提高查询性能，但是会降低写入性能。
- 索引是执行 `LOOKUP` 语句时用于定位到数据的前置条件，如果没有索引，执行 `LOOKUP` 语句会报错。
- 使用索引时，悦数图数据库会自动选择最优的索引。
- 具有高选择度的索引，即索引列中不同值的记录数与总记录数的比值较高（例如身份证号的比值为 1）可显著提升查询性能；而对于低选择度的索引（例如国家），查询性能可能不会带来显著提升。

原生索引

原生索引可以基于指定的属性查询数据，有如下特点：

- 包括 Tag 索引和 Edge type 索引。
- 必须手动重建索引（`REBUILD INDEX`）。
- 支持创建同一个 Tag 或 Edge type 的多个属性的索引（复合索引），但是不能跨 Tag 或 Edge type。

原生索引操作

- CREATE INDEX
- SHOW CREATE INDEX
- SHOW INDEXES
- DESCRIBE INDEX
- REBUILD INDEX
- SHOW INDEX STATUS
- DROP INDEX
- LOOKUP
- MATCH
- 地理空间索引

全文索引

全文索引是基于 Elasticsearch 来实现的，用于对字符串属性进行前缀搜索、通配符搜索、正则表达式搜索和模糊搜索，有如下特点：

- 只允许创建一个属性的索引。
- 不支持逻辑操作，例如 `AND` 、 `OR` 、 `NOT` 。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

如果需要进行整个字符串的匹配，请使用原生索引。

没有 **NULL** 值索引

不支持对值为 NULL 的属性创建索引。

没有唯一索引

在 Cypher 中，可以通过 Constrains 实现属性值的唯一性限制。在 MySQL 中，可以建立唯一索引来限制某字段只有唯一值。在 nGQL 中没有属性的唯一索引（用户自行保证属性值的唯一性）。

数字、日期和时间类型的范围查询

原生索引还支持对数字、日期和时间类型的属性进行范围查询，不支持其他属性类型的范围查询。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.2 CREATE INDEX

前提条件

创建索引之前，请确保相关的 Tag 或 Edge type 已经创建。如何创建 Tag 和 Edge type，请参见 CREATE TAG 和 CREATE EDGE。

如何创建全文索引，请参见部署全文索引。

使用索引必读

索引的概念和使用限制都较为复杂。在使用索引前，请务必阅读以下内容。

`CREATE INDEX` 语句用于对 Tag、EdgeType 或其属性创建原生索引。通常分别称为“Tag 索引”、“Edge type 索引”和“属性索引”。

- Tag 索引和 Edge type 索引应用于和 Tag、Edge type 自身相关的查询，例如用 `LOOKUP` 查找有 Tag `player` 的所有点。
- “属性索引”应用于基于属性的查询，例如基于属性 `age` 找到 `age == 19` 的所有的点。

如果已经为 Tag `T` 的属性 `A` 建立过属性索引 `i_TA` （`T` 的索引为 `i_T` ），索引之间的可替代关系如下（Edge type 索引同理）：

- 查询引擎可以使用 `i_TA` 来替代 `i_T` 。
- 在 `MATCH` 、 `LOOKUP` 语句中 `i_T` 可以替代 `i_TA` 查找属性。

> 历史版本兼容性
>
> 在此前的版本中， `LOOKUP` 语句中的 Tag 或 Edge type 索引不可替代属性索引用于属性查找。

使用替代索引进行查询虽然能获得相同的结果，但查询性能会根据选择的索引有所区别。

> **Caution**
>
> 不要任意在生产环境中使用索引，除非很清楚使用索引对业务的影响。索引会导致写性能大幅下降。
>
> 长索引会降低 Storage 服务的扫描性能，以及占用更多内存。建议将索引长度设置为和要被索引的最长字符串相同。索引长度最长为 256。

操作步骤

如果必须使用索引，通常按照如下步骤：

1. 初次导入数据至悦数图数据库。
2. 创建索引。
3. 重建索引。
4. 使用 LOOKUP 或 MATCH 语句查询数据。不需要（也无法）指定使用哪个索引，悦数图数据库会自动计算。

> **Note**
>
> 如果先创建索引再导入数据，会因为写性能的下降导致导入速度极慢。
>
> 日常增量写入时保持 `--disable_auto_compaction = false` 。
>
> 新创建的索引并不会立刻生效。创建新的索引并尝试立刻使用（例如 `LOOKUP` 或者 `REBUILD INDEX` ）通常会失败（报错 `can't find xxx in the space` ）。因为创建步骤是异步实现的，悦数图数据库要在下一个心跳周期才能完成索引的创建。为确保数据同步，后续操作能顺利进行，请等待 2 个心跳周期（20 秒）。如果需要修改心跳间隔，请为所有配置文件修改参数 `heartbeat_interval_secs` 。

Copyright © Yueshu Graph Technology - All rights reserved.

Danger

创建索引，或者删除并再次创建同名索引后，必须 `REBUILD INDEX` 。否则无法在 `MATCH` 和 `LOOKUP` 语句中返回这些数据。

语法

```
CREATE {TAG | EDGE} INDEX [IF NOT EXISTS] <index_name> ON {<tag_name> | <edge_name>} ([<prop_name_list>]) [COMMENT '<comment>'];
```

参数	说明
`TAG | EDGE`	指定要创建的索引类型。
`IF NOT EXISTS`	检测待创建的索引是否存在，只有不存在时，才会创建索引。
`<index_name>`	索引名。索引名在一个图空间中必须是唯一的。推荐的命名方式为 `i_tagName_propName` 。 索引名称不能以数字开头，支持 1~4 字节的 UTF-8 编码字符，包括英文字母（区分大小写）、数字、中文等，但是不包括除下划线外的特殊字符。使用特殊字符、保留关键字或数字开头时，需要用反引号（`）包围，详情参见关键字和保留字。 注意：如果以中文为索引命名，报 `SyntaxError` 错误时，需使用反引号（`）包围中文字符。
`<tag_name> | <edge_name>`	指定索引关联的 Tag 或 Edge 名称。
`<prop_name_list>`	为变长字符串属性创建索引时，必须用 `prop_name(length)` 指定索引长度；为 Tag 或 Edge type 本身创建索引时，忽略 `<prop_name_list>` 。
`COMMENT`	索引的描述。最大为 256 字节。默认无描述。

创建 **Tag/Edge type** 索引

```
nebula> CREATE TAG INDEX player_index on player();
```

```
nebula> CREATE EDGE INDEX follow_index on follow();
```

为 Tag 或 Edge type 创建索引后，用户可以使用 `LOOKUP` 语句查找 `带有该 Tag 的` 所有点的 VID，或者 `所有该类型的边` 的 `对应起始点 VID、目的点 VID、以及 rank` 。详情请参见 LOOKUP。

创建单属性索引

```
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_0 on player(name(10));
```

上述示例是为所有包含 Tag `player` 的点创建属性 `name` 的索引，索引长度为 10。即只使用属性 `name` 的前 10 个字符来创建索引。

```
# 变长字符串需要指定索引长度。
nebula> CREATE TAG IF NOT EXISTS var_string(p1 string);
nebula> CREATE TAG INDEX IF NOT EXISTS var ON var_string(p1(10));

# 定长字符串不需要指定索引长度。
nebula> CREATE TAG IF NOT EXISTS fix_string(p1 FIXED_STRING(10));
nebula> CREATE TAG INDEX IF NOT EXISTS fix ON fix_string(p1);
```

```
nebula> CREATE EDGE INDEX IF NOT EXISTS follow_index_0 on follow(degree);
```

创建复合属性索引

`复合属性索引` 用于查找一个 Tag（或者 Edge type) 中的多个属性（的组合）。

```
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_1 on player(name(10), age);
```


Copyright © Yueshu Graph Technology - All rights reserved.

Caution

不支持跨 Tag 或 Edge type 创建复合索引。

Note

使用复合属性索引时，遵循"最左匹配原则"，必须从复合属性索引的最左侧开始匹配。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.3 SHOW INDEXES

`SHOW INDEXES` 语句可以列出当前图空间内的所有 Tag 和 Edge type（包括属性）的索引。

语法

```
SHOW {TAG | EDGE} INDEXES;
```

示例

```
nebula> SHOW TAG INDEXES;
+-----------------+-------------+----------------+
| Index Name      | By Tag      | Columns        |
+-----------------+-------------+----------------+
| "fix"           | "fix_string" | ["p1"]         |
| "player_index_0" | "player"    | ["name"]       |
| "player_index_1" | "player"    | ["name", "age"] |
| "var"           | "var_string" | ["p1"]         |
+-----------------+-------------+----------------+

nebula> SHOW EDGE INDEXES;
+---------------+----------+--------+
| Index Name    | By Edge  | Columns |
+---------------+----------+--------+
| "follow_index" | "follow" | []     |
+---------------+----------+--------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.4 SHOW CREATE INDEX

`SHOW CREATE INDEX` 展示创建 Tag 或者 Edge type 时使用的 nGQL 语句，其中包含索引的详细信息，例如其关联的属性。

语法

```
SHOW CREATE {TAG | EDGE} INDEX <index_name>;
```

示例

用户可以先运行 `SHOW TAG INDEXES` 查看有哪些 Tag 索引，然后用 `SHOW CREATE TAG INDEX` 查看指定索引的创建信息。

```
nebula> SHOW TAG INDEXES;
+-----------------+----------+----------+
| Index Name      | By Tag   | Columns  |
+-----------------+----------+----------+
| "player_index_0" | "player" | []       |
| "player_index_1" | "player" | ["name"] |
+-----------------+----------+----------+

nebula> SHOW CREATE TAG INDEX player_index_1;
+-----------------+-----------------------------------------------+
| Tag Index Name  | Create Tag Index                              |
+-----------------+-----------------------------------------------+
| "player_index_1" | "CREATE TAG INDEX `player_index_1` ON `player` ( |
|                 |  `name`(20)                                   |
|                 | )"                                            |
+-----------------+-----------------------------------------------+
```

Edge type 索引可以用类似的方法查询：

```
nebula> SHOW EDGE INDEXES;
+---------------+----------+---------+
| Index Name    | By Edge  | Columns |
+---------------+----------+---------+
| "follow_index" | "follow" | []      |
+---------------+----------+---------+

nebula> SHOW CREATE EDGE INDEX follow_index;
+----------------+----------------------------------------------+
| Edge Index Name | Create Edge Index                            |
+----------------+----------------------------------------------+
| "follow_index"  | "CREATE EDGE INDEX `follow_index` ON `follow` ( |
|                 | )"                                           |
+----------------+----------------------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.5 DESCRIBE INDEX

`DESCRIBE INDEX` 语句可以查看指定索引的信息，包括索引的属性名称（Field）和数据类型（Type）。

语法

```
DESCRIBE {TAG | EDGE} INDEX <index_name>;
```

示例

```
nebula> DESCRIBE TAG INDEX player_index_0;
+--------+--------------------+
| Field  | Type               |
+--------+--------------------+
| "name" | "fixed_string(30)" |
+--------+--------------------+

nebula> DESCRIBE TAG INDEX player_index_1;
+--------+--------------------+
| Field  | Type               |
+--------+--------------------+
| "name" | "fixed_string(10)" |
| "age"  | "int64"            |
+--------+--------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.6 REBUILD INDEX

Danger

- 索引功能不会自动对其创建之前已存在的存量数据生效————在索引重建完成之前，无法基于该索引使用 `LOOKUP` 和 `MATCH` 语句查询到存量数据。
- 索引的重建未完成时，依赖索引的查询仅能使用部分索引，因此不能获得准确结果。

请在创建索引后，选择合适的时间为存量数据重建索引。使用索引的详情请参见 CREATE INDEX。

Performance

通过修改配置文件中的 `rebuild_index_part_rate_limit` 和 `rebuild_index_batch_size` 两个参数，可优化重建索引的速度，另外，更大参数可能会导致更高的内存和网络占用，详情请参见 Storage服务配置。

语法

```
REBUILD {TAG | EDGE} INDEX [<index_name_list>];

<index_name_list>::=
    [index_name [, index_name] ...]
```

- 可以一次重建多个索引，索引名称之间用英文逗号（,）分隔。如果没有指定索引名称，将会重建所有索引。
- 重建完成后，用户可以使用命令 `SHOW {TAG | EDGE} INDEX STATUS` 检查索引是否重建完成。详情请参见 SHOW INDEX STATUS。

示例

```
nebula> CREATE TAG IF NOT EXISTS person(name string, age int, gender string, email string);
nebula> CREATE TAG INDEX IF NOT EXISTS single_person_index ON person(name(10));

# 重建索引，返回任务 ID。
nebula> REBUILD TAG INDEX single_person_index;
+------------+
| New Job Id |
+------------+
| 31         |
+------------+

# 查看索引状态。
nebula> SHOW TAG INDEX STATUS;
+-----------------------+--------------+
| Name                  | Index Status |
+-----------------------+--------------+
| "single_person_index" | "FINISHED"   |
+-----------------------+--------------+

# 也可以使用 SHOW JOB <job_id>查看重建索引的任务状态。
nebula> SHOW JOB 31;
+---------------+--------------------+------------+-------------------------+-------------------------+-------------+
| Job Id(TaskId) | Command(Dest)      | Status     | Start Time              | Stop Time               | Error Code  |
+---------------+--------------------+------------+-------------------------+-------------------------+-------------+
| 31            | "REBUILD_TAG_INDEX" | "FINISHED" | 2021-07-07T09:04:24.000 | 2021-07-07T09:04:24.000 | "SUCCEEDED" |
| 0             | "storaged1"        | "FINISHED" | 2021-07-07T09:04:24.000 | 2021-07-07T09:04:28.000 | "SUCCEEDED" |
| 1             | "storaged2"        | "FINISHED" | 2021-07-07T09:04:24.000 | 2021-07-07T09:04:28.000 | "SUCCEEDED" |
| 2             | "storaged0"        | "FINISHED" | 2021-07-07T09:04:24.000 | 2021-07-07T09:04:28.000 | "SUCCEEDED" |
| "Total:3"     | "Succeeded:3"      | "Failed:0" | "In Progress:0"         | ""                      | ""          |
+---------------+--------------------+------------+-------------------------+-------------------------+-------------+
+---------------+--------------------+------------+-------------------------+-------------------------+-------------+
```

悦数图数据库创建一个任务去重建索引，因此可以根据返回的任务 ID，通过 `SHOW JOB <job_id>` 语句查看任务状态。详情请参见 SHOW JOB。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.7 SHOW INDEX STATUS

`SHOW INDEX STATUS` 语句可以查看索引名称和对应作业的状态。

索引状态包括:

- `QUEUE`： 队列中
- `RUNNING`： 执行中
- `FINISHED`： 已完成
- `FAILED`： 失败
- `STOPPED`： 停止
- `INVALID`： 失效

> **Note**
>
> 如何创建索引请参见 CREATE INDEX。

语法

```
SHOW {TAG | EDGE} INDEX STATUS;
```

示例

```
nebula> SHOW TAG INDEX STATUS;
+---------------------+--------------+
| Name                | Index Status |
+---------------------+--------------+
| "player_index_0"    | "FINISHED"   |
| "player_index_1"    | "FINISHED"   |
+---------------------+--------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.13.8 DROP INDEX

`DROP INDEX` 语句可以删除当前图空间中已存在的索引。

前提条件

执行 `DROP INDEX` 语句需要当前登录的用户拥有指定图空间的 `DROP TAG INDEX` 和 `DROP EDGE INDEX` 权限，否则会报错。

语法

```
DROP {TAG | EDGE} INDEX [IF EXISTS] <index_name>;
```

`IF EXISTS`： 检测待删除的索引是否存在，只有存在时，才会删除索引。

示例

```
nebula> DROP TAG INDEX player_index_0;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.14 全文索引

5.14.1 全文索引限制

本文介绍全文索引的限制，请在使用全文索引前仔细阅读。

Caution

3.5.0 版本重做了全文索引功能，不兼容之前版本的全文索引。如果你想继续使用通配符、正则、模糊匹配等方式，有如下 3 种方式：

- 删除原有全文索引，使用新的方式重建全文索引，使用新的查询语法。
- 删除原有全文索引，直接用悦数图数据库的原生索引和字符串运算符。
- 继续使用之前版本的悦数图数据库及其全文索引功能。

全文索引有如下限制：

- 全文索引当前仅支持 `LOOKUP` 语句。
- 全文索引名称只能包含数字、小写字母、下划线。
- 不同图空间内的全文索引名称不能重复。
- 查询默认返回 10 条数据。可以使用 `LIMIT` 子句返回更多数据，最多可以返回 10000 条。可以修改 ElasticSearch 的参数调整最大返回条数。
- 如果 Tag/Edge type 上存在全文索引，无法删除或修改 Tag/Edge type。
- 属性的类型必须为 `STRING` 或 `FIXED_STRING` 。
- 全文索引不支持多个 Tag/Edge type 的搜索。
- 全文索引不支持搜索属性值为 `NULL` 的属性。
- 不支持修改 Elasticsearch 中的索引，只能删除重建。
- 不支持修改分词器，需要删除索引数据后重建索引时指定。
- 确保同时启动了 Elasticsearch 集群和悦数图数据库，否则可能导致 Elasticsearch 集群写入的数据不完整。
- 从写入悦数图数据库，到写入 listener，再到写入 Elasticsearch 并创建索引可能需要一段时间。如果访问全文索引时返回未找到索引，可检查索引任务的状态。
- 使用 K8s 方式部署的悦数图数据库集群不支持自动部署全文索引，但支持手动部署。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.14.2 部署全文索引

悦数图数据库的全文索引是基于 Elasticsearch 实现，这意味着用户可以使用 Elasticsearch 全文查询语言来检索想要的内容。全文索引由内置的进程管理，当 listener 集群和 Elasticsearch 集群部署后，内置的进程只能为数据类型为定长字符串或变长字符串的属性创建全文索引。

注意事项

使用全文索引前，请确认已经了解全文索引的使用限制。

部署 Elasticsearch 集群

部署 Elasticsearch 集群请参见 Kubernetes 安装 Elasticsearch 或单机安装 Elasticsearch。目前仅支持 7.x 版本的 Elasticsearch。

> **Note**
>
> 为了支持外网访问 Elasticsearch，请将 `config/elasticsearch.yml` 中的 `network.host` 设置为 `0.0.0.0` 。

用户可以配置 Elasticsearch 来满足业务需求，如果需要定制 Elasticsearch，请参见 Elasticsearch 官方文档。

登录文本搜索客户端

部署 Elasticsearch 集群之后，可以使用 `SIGN IN` 语句登录 Elasticsearch 客户端。必须使用 Elasticsearch 配置文件中的 IP 地址和端口才能正常连接，同时登录多个客户端，请在多个 `elastic_ip:port` 之间用英文逗号（,）分隔。

语法

```
SIGN IN TEXT SERVICE (<elastic_ip:port>, {HTTP | HTTPS} [,"<username>", "<password>"]) [, (<elastic_ip:port>, ...)];
```

示例

```
nebula> SIGN IN TEXT SERVICE (192.168.8.100:9200, HTTP);
```

> **Note**
>
> Elasticsearch 默认没有用户名和密码，如果设置了用户名和密码，请在 `SIGN IN` 语句中指定。

> **Caution**
>
> Elasticsearch 客户端只能登录一次，如有修改，需要 `SIGN OUT` 后重新 `SIGN IN` ，且客户端对全局生效，多个图空间共享相同的 Elasticsearch 客户端。

显示文本搜索客户端

`SHOW TEXT SEARCH CLIENTS` 语句可以列出文本搜索客户端。

语法

```
SHOW TEXT SEARCH CLIENTS;
```

示例

```
nebula> SHOW TEXT SEARCH CLIENTS;
+-----------------+-----------------+------+
| Type            | Host            | Port |
+-----------------+-----------------+------+
| "ELASTICSEARCH" | "192.168.8.100" | 9200 |
+-----------------+-----------------+------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

退出文本搜索客户端

`SIGN OUT TEXT SERVICE` 语句可以退出所有文本搜索客户端。

语法

```
SIGN OUT TEXT SERVICE;
```

示例

```
nebula> SIGN OUT TEXT SERVICE;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.14.3 部署 Raft listener

全文索引的数据是异步写入 Elasticsearch 集群的。流程是通过 Storage 服务的 Raft listener（简称 listener）这个单独部署的进程，从 Storage 服务读取数据，然后将它们写入 Elasticsearch 集群。

前提条件

- 已经了解全文索引的使用限制。
- 已经部署悦数图数据库集群。
- 完成部署 Elasticsearch 集群。
- 准备一台或者多台服务器，以部署 Raft listener。

注意事项

- 请保证悦数图数据库各服务（Metad、Storaged、Graphd、listener）有相同的版本。
- 请保证 listener 所在机器的时区与悦数图数据库各服务所在机器的时区一致。
- 只能为一个图空间一次性添加所有的 **listener** 机器。尝试向已经存在有 listener 的图空间再添加新 listener 会失败。因此，需在一个命令语句里完整地添加全部的 listener。

部署流程

第一步：安装 **LISTENER** 服务

listener 服务与 storaged 服务使用相同的二进制文件，但是二者配置文件不同，进程使用端口也不同。我们在所有需要部署 listener 的服务器上都安装悦数图数据库，但是不启动服务。详情请参见使用 RPM 或 DEB 安装包安装悦数图数据库。

第二步：准备 **LISTENER** 的配置文件

在 `etc` 目录内将 `nebula-storaged-listener.conf.default` 或 `nebula-storaged-listener.conf.production` 去掉后缀命名为 `nebula-storaged-listener.conf` ，然后修改配置内容。

Copyright © Yueshu Graph Technology - All rights reserved.

大部分配置与 Storage 服务的配置文件相同，本文仅介绍差异部分。

名称	预设值	说明
`daemonize`	`true`	是否启动守护进程。
`pid_file`	`pids/nebula-storaged-listener.pid`	记录进程 ID 的文件。
`meta_server_addrs`	-	全部 Meta 服务的 IP 地址和端口。多个 Meta 服务用英文逗号（,）分隔。
`local_ip`	-	listener 服务的本地 IP 地址。请使用真实的 listener 机器 IP 地址替换 `127.0.0.1` 。
`port`	-	listener 服务的 RPC 守护进程监听端口。
`heartbeat_interval_secs`	`10`	Meta 服务的心跳间隔。单位：秒（s）。
`listener_path`	`data/listener`	listener 的 WAL 目录。只允许使用一个目录。
`data_path`	`data`	出于兼容性考虑，可以忽略此参数。填充一个默认值 `data` 。
`part_man_type`	`memory`	部件管理器类型，可选值为 `memory` 和 `meta` 。
`rocksdb_batch_size`	`4096`	批处理操作的默认保留字节。
`rocksdb_block_cache`	`4`	BlockBasedTable 的默认块缓存大小。单位：兆字节（MB）。
`engine_type`	`rocksdb`	存储引擎类型，例如 `rocksdb` 、 `memory` 等。
`part_type`	`simple`	部件类型，例如 `simple` 、 `consensus` 等。

第三步：启动 **LISTENER**

在需要部署 listener 集群的安装目录下，执行如下命令启动 listener：

```
./bin/nebula-storaged --flagfile etc/nebula-storaged-listener.conf
```

第四步：添加 **LISTENER** 到悦数图数据库集群

用命令行连接到悦数图数据库，然后执行 `USE <space>` 进入需要创建全文索引的图空间。然后执行如下命令添加 listener：

```
ADD LISTENER ELASTICSEARCH <listener_ip:port> [,<listener_ip:port>, ...]
```

Warning

listener 必须使用真实的 IP 地址。

请在一个语句里完整地添加所有 listener。例如：

```
nebula> ADD LISTENER ELASTICSEARCH 192.168.8.100:9789,192.168.8.101:9789;
```

查看 listener

执行 `SHOW LISTENER` 语句可以列出所有的 listener。

示例

```
nebula> SHOW LISTENER;
+--------+-----------------+------------------------+-------------+
| PartId | Type            | Host                   | Host Status |
+--------+-----------------+------------------------+-------------+
| 1      | "ELASTICSEARCH" | ""192.168.8.100":9789" | "ONLINE"    |
| 2      | "ELASTICSEARCH" | ""192.168.8.100":9789" | "ONLINE"    |
| 3      | "ELASTICSEARCH" | ""192.168.8.100":9789" | "ONLINE"    |
+--------+-----------------+------------------------+-------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

删除 listener

执行 `REMOVE LISTENER ELASTICSEARCH` 语句可以删除图空间的所有 listener。

示例

```
nebula> REMOVE LISTENER ELASTICSEARCH;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.14.4 全文搜索

全文搜索是基于全文索引对值为字符串类型的属性进行前缀搜索、通配符搜索、正则表达式搜索和模糊搜索。

在 `LOOKUP` 语句中，使用 `WHERE` 子句指定字符串的搜索条件。

前提条件

请确保已经部署全文索引。详情请参见部署全文索引和部署 listener。

注意事项

使用全文索引前，请确认已经了解全文索引的使用限制。

全文本查询

全文本查询使您能够搜索经过分析的文本字段，使用具有严格语法的解析器，根据提供的查询字符串返回内容。详情参见Query string query。

语法

创建全文索引

```
CREATE FULLTEXT {TAG | EDGE} INDEX <index_name> ON {<tag_name> | <edge_name>} (<prop_name> [,<prop_name>]...) [ANALYZER="<analyzer_name>"];
```

- 创建全文索引时支持多属性的复合索引。
- `<analyzer_name>` 为分词器名称。默认为 `standard` 。使用其他分词器（例如 IK Analysis）需要确保已经提前在 Elasticsearch 中安装对应分词器。

显示全文索引

```
SHOW FULLTEXT INDEXES;
```

重建全文索引

```
REBUILD FULLTEXT INDEX;
```

Caution

数据量大时，重建全文索引速度较慢，可以修改 Storage 服务的配置文件（`nebula-storaged.conf`）中 `snapshot_send_files=false` 。

删除全文索引

```
DROP FULLTEXT INDEX <index_name>;
```

使用查询选项

```
LOOKUP ON {<tag> | <edge_type>} WHERE ES_QUERY(<index_name>, "<text>") YIELD <return_list> [| LIMIT [<offset>,] <number_rows>];

<return_list>
    <prop_name> [AS <prop_alias>] [, <prop_name> [AS <prop_alias>] ...] [, id(vertex)  [AS <prop_alias>]] [, score() AS <score_alias>]
```

- `index_name`： 索引名称。
- `text`： 搜索条件。WHERE 后只能跟一个 ES_QUERY，所有判断条件必须写在 text 里。详细语法请参见Query string syntax。
- `score()`： 对符合条件的点做 N 度扩展计算出的分数。默认值为 `1.0` 。分数越高，匹配程度越高。返回值默认按照分数从高到低排序。详情参见 Search and Scoring in Lucene。

示例

Copyright © Yueshu Graph Technology - All rights reserved.

```
//创建图空间。
nebula> CREATE SPACE IF NOT EXISTS basketballplayer (partition_num=3,replica_factor=1, vid_type=fixed_string(30));

//登录文本搜索客户端。
nebula> SIGN IN TEXT SERVICE (192.168.8.100:9200, HTTP);

//检查是否成功登录。
nebula> SHOW TEXT SEARCH CLIENTS;
+-----------------+-----------------+------+
| Type            | Host            | Port |
+-----------------+-----------------+------+
| "ELASTICSEARCH" | "192.168.8.100" | 9200 |
+-----------------+-----------------+------+

//切换图空间。
nebula> USE basketballplayer;

//添加 listener 到悦数图数据库集群。
nebula> ADD LISTENER ELASTICSEARCH 192.168.8.100:9789;

//检查是否成功添加 listener，当状态为 Online 时表示成功添加。
nebula> SHOW LISTENER;
+--------+-----------------+------------------------+-------------+
| PartId | Type            | Host                   | Host Status |
+--------+-----------------+------------------------+-------------+
| 1      | "ELASTICSEARCH" | ""192.168.8.100":9789" | "ONLINE"    |
| 2      | "ELASTICSEARCH" | ""192.168.8.100":9789" | "ONLINE"    |
| 3      | "ELASTICSEARCH" | ""192.168.8.100":9789" | "ONLINE"    |
+--------+-----------------+------------------------+-------------+

//创建 Tag。
nebula> CREATE TAG IF NOT EXISTS player(name string, city string);

//创建单属性全文索引。
nebula> CREATE FULLTEXT TAG INDEX fulltext_index_1 ON player(name) ANALYZER="standard";

//创建多属性全文索引。
nebula> CREATE FULLTEXT TAG INDEX fulltext_index_2 ON player(name,city) ANALYZER="standard";

//重建全文索引。
nebula> REBUILD FULLTEXT INDEX;

//查看全文索引。
nebula> SHOW FULLTEXT INDEXES;
+--------------------+-------------+-------------+--------------+------------+
| Name               | Schema Type | Schema Name | Fields       | Analyzer   |
+--------------------+-------------+-------------+--------------+------------+
| "fulltext_index_1" | "Tag"       | "player"    | "name"       | "standard" |
| "fulltext_index_2" | "Tag"       | "player"    | "name, city" | "standard" |
+--------------------+-------------+-------------+--------------+------------+

//插入测试数据。
nebula> INSERT VERTEX player(name, city) VALUES \
  "Russell Westbrook": ("Russell Westbrook", "Los Angeles"), \
  "Chris Paul": ("Chris Paul", "Houston"),\
  "Boris Diaw": ("Boris Diaw", "Houston"),\
  "David West": ("David West", "Philadelphia"),\
  "Danny Green": ("Danny Green", "Philadelphia"),\
  "Tim Duncan": ("Tim Duncan", "New York"),\
  "James Harden": ("James Harden", "New York"),\
  "Tony Parker": ("Tony Parker", "Chicago"),\
  "Aron Baynes": ("Aron Baynes", "Chicago"),\
  "Ben Simmons": ("Ben Simmons", "Phoenix"),\
  "Blake Griffin": ("Blake Griffin", "Phoenix");

//测试查询
nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"Chris") YIELD id(vertex);
+--------------+
| id(VERTEX)   |
+--------------+
| "Chris Paul" |
+--------------+

nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"Harden") YIELD properties(vertex);
+-------------------------------------------------------------+
| properties(VERTEX)                                          |
+-------------------------------------------------------------+
| {_vid: "James Harden", city: "New York", name: "James Harden"} |
+-------------------------------------------------------------+

nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"Da*") YIELD properties(vertex);
+-------------------------------------------------------------+
| properties(VERTEX)                                          |
+-------------------------------------------------------------+
| {_vid: "David West", city: "Philadelphia", name: "David West"}   |
| {_vid: "Danny Green", city: "Philadelphia", name: "Danny Green"} |
+-------------------------------------------------------------+

nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"*b*") YIELD id(vertex);
+---------------------+
| id(VERTEX)          |
+---------------------+
| "Russell Westbrook" |
| "Boris Diaw"        |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| "Aron Baynes"      |
| "Ben Simmons"      |
| "Blake Griffin"    |
+--------------------+

nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"*b*") YIELD id(vertex) | LIMIT 2,3;
+----------------+
| id(VERTEX)     |
+----------------+
| "Aron Baynes"  |
| "Ben Simmons"  |
| "Blake Griffin" |
+----------------+

nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"*b*") YIELD id(vertex) | YIELD count(*);
+----------+
| count(*) |
+----------+
| 5        |
+----------+

nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"*b*") YIELD id(vertex), score() AS score;
+--------------------+-------+
| id(VERTEX)         | score |
+--------------------+-------+
| "Russell Westbrook" | 1.0   |
| "Boris Diaw"       | 1.0   |
| "Aron Baynes"      | 1.0   |
| "Ben Simmons"      | 1.0   |
| "Blake Griffin"    | 1.0   |
+--------------------+-------+

//对于包含 b 的词的文档，它的得分将乘以加权因子 4，而对于包含 c 的词的文档，使用默认加权因子 1。
nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_1,"*b*^4 OR *c*") YIELD id(vertex), score() AS score;
+--------------------+-------+
| id(VERTEX)         | score |
+--------------------+-------+
| "Russell Westbrook" | 4.0   |
| "Boris Diaw"       | 4.0   |
| "Aron Baynes"      | 4.0   |
| "Ben Simmons"      | 4.0   |
| "Blake Griffin"    | 4.0   |
| "Chris Paul"       | 1.0   |
| "Tim Duncan"       | 1.0   |
+--------------------+-------+

//使用多属性全文索引查询，此时条件会在索引的所有属性内进行匹配。
nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_2,"*h*") YIELD properties(vertex);
+------------------------------------------------------------+
| properties(VERTEX)                                         |
+------------------------------------------------------------+
| {_vid: "Chris Paul", city: "Houston", name: "Chris Paul"}  |
| {_vid: "Boris Diaw", city: "Houston", name: "Boris Diaw"}  |
| {_vid: "David West", city: "Philadelphia", name: "David West"} |
| {_vid: "James Harden", city: "New York", name: "James Harden"} |
| {_vid: "Tony Parker", city: "Chicago", name: "Tony Parker"} |
| {_vid: "Aron Baynes", city: "Chicago", name: "Aron Baynes"} |
| {_vid: "Ben Simmons", city: "Phoenix", name: "Ben Simmons"} |
| {_vid: "Blake Griffin", city: "Phoenix", name: "Blake Griffin"} |
| {_vid: "Danny Green", city: "Philadelphia", name: "Danny Green"} |
+------------------------------------------------------------+

//使用多属性全文索引查询时，可以对不同的属性指定不同的文本进行查询。
nebula> LOOKUP ON player WHERE ES_QUERY(fulltext_index_2,"name:*b* AND city:Houston") YIELD properties(vertex);
+-------------------------------------------------------+
| properties(VERTEX)                                    |
+-------------------------------------------------------+
| {_vid: "Boris Diaw", city: "Houston", name: "Boris Diaw"} |
+-------------------------------------------------------+

//删除单属性全文索引。
nebula> DROP FULLTEXT INDEX fulltext_index_1;
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.15 子图和路径

5.15.1 GET SUBGRAPH

`GET SUBGRAPH` 语句查询并返回一个通过从指定点出发对图进行游走而生成的子图。在 `GET SUBGRAPH` 语句中，用户可以指定游走的步数以及游走所经过的边的类型或方向。

语法

```
GET SUBGRAPH [WITH PROP] [<step_count> {STEP|STEPS}] FROM {<vid>, <vid>...}
[{IN | OUT | BOTH} <edge_type>, <edge_type>...]
[WHERE <expression> [AND <expression> ...]]
YIELD [VERTICES AS <vertex_alias>] [, EDGES AS <edge_alias>];
```

- `WITH PROP`： 展示属性。不添加本参数则隐藏属性。
- `step_count`： 指定从起始点开始的跳数，返回从 0 到 `step_count` 跳的子图。必须是非负整数。默认值为 1。
- `vid`： 指定起始点 ID。
- `edge_type`： 指定 Edge type。可以用 `IN` 、 `OUT` 和 `BOTH` 来指定起始点上该 Edge type 的方向。默认为 `BOTH` 。
- `WHERE`： 指定遍历的过滤条件，可以结合布尔运算符 AND 使用。
- `YIELD`： 定义需要返回的输出。可以仅返回点或边。必须设置别名。

Note

`GET SUBGRAPH` 语句检索的路径类型为 `trail` ，即检索的路径只有点可以重复，边不可以重复。详情请参见路径。

WHERE 语句限制

在 `GET SUBGRAPH` 语句中使用 `WHERE` 子句，注意以下限制：

- 仅支持 `AND` 运算符。
- 仅支持过滤目的点，点的格式为 `$$.tagName.propName` 。
- 支持过滤边，边的格式为 `edge_type.propName` 。
- 支持数学函数、聚合函数、字符串函数、日期时间函数、列表函数中的通用函数和类型转化函数。
- 不支持聚合函数、Schema 相关函数、条件表达式函数、谓词函数、geo 函数和自定义函数，列表函数中除通用函数以外的函数。

示例

以下面的示例图进行演示。

Copyright © Yueshu Graph Technology - All rights reserved.

插入测试数据：

```
nebula> CREATE SPACE IF NOT EXISTS subgraph(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
nebula> USE subgraph;
nebula> CREATE TAG IF NOT EXISTS player(name string, age int);
nebula> CREATE TAG IF NOT EXISTS team(name string);
nebula> CREATE EDGE IF NOT EXISTS follow(degree int);
nebula> CREATE EDGE IF NOT EXISTS serve(start_year int, end_year int);
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);
nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);
nebula> INSERT VERTEX player(name, age) VALUES "player102":("LaMarcus Aldridge", 33);
nebula> INSERT VERTEX team(name) VALUES "team203":("Trail Blazers"), "team204":("Spurs");
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player102":(90);
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> INSERT EDGE follow(degree) VALUES "player102" -> "player100":(75);
nebula> INSERT EDGE serve(start_year, end_year) VALUES "player101" -> "team204":(1999, 2018),"player102" -> "team203":(2006,  2015);
```

- 查询从点 player101 开始、0~1 跳、所有 Edge type 的子图。

```
nebula> GET SUBGRAPH 1 STEPS FROM "player101" YIELD VERTICES AS nodes, EDGES AS relationships;
+-----------------------------------------------------------------------
+-------------------------------------------------------------------------------------------------------------------------------------------------+
| nodes                                                                 |
relationships                                                                                                                                    |
+-----------------------------------------------------------------------
+-------------------------------------------------------------------------------------------------------------------------------------------------+
| [("player101" :player{})]                                             | [[:serve "player101"->"team204" @0 {}], [:follow "player101"->"player100" @0 {}], [:follow "player101"-
>"player102" @0 {}]] |
| [("team204" :team{}), ("player100" :player{}), ("player102" :player{})] | [[:follow "player102"->"player100" @0
{}]]                                                                   |
+-----------------------------------------------------------------------
+-------------------------------------------------------------------------------------------------------------------------------------------------+
```

返回的子图如下。

Copyright © Yueshu Graph Technology - All rights reserved.

- 查询从点 `player101` 开始、0~1 跳、 `follow` 类型的入边的子图。

```
nebula> GET SUBGRAPH 1 STEPS FROM "player101" IN follow YIELD VERTICES AS nodes, EDGES AS relationships;
+--------------------------+--------------+
| nodes                    | relationships |
+--------------------------+--------------+
| [("player101" :player{})] | []           |
+--------------------------+--------------+
```

因为 `player101` 没有 `follow` 类型的入边。所以仅返回点 `player101` 。

- 查询从点 `player101` 开始、0~1 跳、 `serve` 类型的出边的子图，同时展示边的属性。

```
nebula> GET SUBGRAPH WITH PROP 1 STEPS FROM "player101" OUT serve YIELD VERTICES AS nodes, EDGES AS relationships;
+-----------------------------------------------------+-------------------------------------------------------------------+
| nodes                                               | relationships                                                     |
+-----------------------------------------------------+-------------------------------------------------------------------+
| [("player101" :player{age: 36, name: "Tony Parker"})] | [[:serve "player101"->"team204" @0 {end_year: 2018, start_year: 1999}]] |
| [("team204" :team{name: "Spurs"})]                  | []                                                                |
+-----------------------------------------------------+-------------------------------------------------------------------+
```

返回的子图如下。

- 查询从点 `player101` 开始、0~2 跳、 `follow` 类型边 `degree` 大于 90，年龄大于 30 的子图，同时展示边的属性。

```
nebula> GET SUBGRAPH WITH PROP 2 STEPS FROM "player101" \
    WHERE follow.degree > 90 AND $$.player.age > 30 \
    YIELD VERTICES AS nodes, EDGES AS relationships;
+-----------------------------------------------------+-----------------------------------------------------+
| nodes                                               | relationships                                       |
+-----------------------------------------------------+-----------------------------------------------------+
| [("player101" :player{age: 36, name: "Tony Parker"})] | [[:follow "player101"->"player100" @0 {degree: 95}]] |
| [("player100" :player{age: 42, name: "Tim Duncan"})]  | []                                                  |
+-----------------------------------------------------+-----------------------------------------------------+
```

FAQ

为什么返回结果中会出现超出 **STEP_COUNT** 跳数之外的关系?

为了展示子图的完整性，会在满足条件的所有点上额外查询一跳。例如下图。

Copyright © Yueshu Graph Technology - All rights reserved.

- 用 `GET SUBGRAPH 1 STEPS FROM "A";` 查询的满足结果的路径是 `A->B` 、 `B->A` 和 `A->C` ，为了子图的完整性，会在满足结果的点上额外查询一跳，即 `B->C` 。
- 用 `GET SUBGRAPH 1 STEPS FROM "A" IN follow;` 查询的满足结果的路径是 `B->A` ，在满足结果的点上额外查询一跳，即 `A->B` 。

如果只是查询满足条件的路径或点，建议使用 MATCH 或 GO 语句。例如：

```
nebula> MATCH p= (v:player) -- (v2) WHERE id(v)=="A" RETURN p;

nebula> GO 1 STEPS FROM "A" OVER follow YIELD src(edge),dst(edge);
```

为什么返回结果中会出现低于 **STEP_COUNT** 跳数的关系?

查询到没有多余子图数据时会停止查询，且不会返回空值。

```
nebula> GET SUBGRAPH 100 STEPS FROM "player101" OUT follow YIELD VERTICES AS nodes, EDGES AS relationships;
+-------------------------------------------------+-------------------------------------------------------------------------------------+
| nodes                                           | relationships                                                                       |
+-------------------------------------------------+-------------------------------------------------------------------------------------+
| [("player101" :player{})]                       | [[:follow "player101"->"player100" @0 {}], [:follow "player101"->"player102" @0 {}]] |
| [("player100" :player{}), ("player102" :player{})] | [[:follow "player102"->"player100" @0 {}]]                                          |
+-------------------------------------------------+-------------------------------------------------------------------------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.15.2 FIND PATH

`FIND PATH` 语句查找指定起始点和目的点之间的路径。

> **Note**
>
> 用户可在配置文件 `nebula-graphd.conf` 中添加 `num_operator_threads` 参数提高 `FIND PATH` 的查询性能。 `num_operator_threads` 的取值为 `2` ~ `10` ，该值不能超过 Graph 服务所在机器的 CPU 核心个数，建议设置为 Graph 服务所在机器的 CPU 核心个数。关于配置文件的详细信息，参见 Graph 服务配置。

语法

```
FIND { SHORTEST | ALL | NOLOOP } PATH [WITH PROP] FROM <vertex_id_list> TO <vertex_id_list>
OVER <edge_type_list> [REVERSELY | BIDIRECT]
[<WHERE clause>] [UPTO <N> {STEP|STEPS}]
YIELD path as <alias>
[| ORDER BY $-.path] [| LIMIT <M>];

<vertex_id_list> ::=
    [vertex_id [, vertex_id] ...]
```

- `SHORTEST` ： 查找最短路径。
- `ALL` ： 查找所有路径。
- `NOLOOP` ： 查找非循环路径。
- `WITH PROP` ： 展示点和边的属性。不添加本参数则隐藏属性。
- `<vertex_id_list>` ： 点 ID 列表。多个点用英文逗号（,）分隔。支持 `$-` 和 `$var` 。
- `<edge_type_list>` ： Edge type 列表。多个 Edge type 用英文逗号（,）分隔。 `*` 表示所有 Edge type。
- `REVERSELY` | `BIDIRECT` ： `REVERSELY` 表示反向， `BIDIRECT` 表示双向。
- `<WHERE clause>` ： 可以使用 `WHERE` 子句过滤边属性。
- `<N>` ： 路径的最大跳数。默认值为 `5` 。
- `<M>` ： 指定返回的最大行数。

> **Note**
>
> `FIND PATH` 语句检索的路径类型为 `trail` ，即检索的路径只有点可以重复，边不可以重复。详情请参见路径。

限制

- 指定起始点和目的点的列表后，会返回起始点和目的点所有组合的路径。
- 搜索所有路径时可能会出现循环。
- 使用 `WHERE` 子句时只能过滤边属性，暂不支持过滤点属性，且不支持函数。
- graphd 是单进程查询，会占用很多内存。

示例

返回的路径格式类似于 `(<vertex_id>)-[:<edge_type_name>@<rank>]->(<vertex_id)` 。

```
nebula> FIND SHORTEST PATH FROM "player102" TO "team204" OVER * YIELD path AS p;
+-----------------------------------------+
| p                                       |
+-----------------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| <("player102")-[:serve@0 {}]->("team204")> |
+--------------------------------------------+
```

```
nebula> FIND SHORTEST PATH WITH PROP FROM "team204" TO "player100" OVER * REVERSELY YIELD path AS p;
+------------------------------------------------------------------------------------------------------------------------------------------------+
| p                                                                                                                                              |
+------------------------------------------------------------------------------------------------------------------------------------------------+
| <("team204" :team{name: "Spurs"})<-[:serve@0 {end_year: 2016, start_year: 1997}]-("player100" :player{age: 42, name: "Tim Duncan"})> |
+------------------------------------------------------------------------------------------------------------------------------------------------+
```

```
nebula> FIND ALL PATH FROM "player100" TO "team204" OVER * WHERE follow.degree is EMPTY or follow.degree >=0 YIELD path AS p;
+------------------------------------------------------------------------------+
| p                                                                            |
+------------------------------------------------------------------------------+
| <("player100")-[:serve@0 {}]->("team204")>                                   |
| <("player100")-[:follow@0 {}]->("player125")-[:serve@0 {}]->("team204")>     |
| <("player100")-[:follow@0 {}]->("player101")-[:serve@0 {}]->("team204")>     |
| ...                                                                          |
+------------------------------------------------------------------------------+
```

```
nebula> FIND NOLOOP PATH FROM "player100" TO "team204" OVER * YIELD path AS p;
+------------------------------------------------------------------------------------------------------------+
| p                                                                                                          |
+------------------------------------------------------------------------------------------------------------+
| <("player100")-[:serve@0 {}]->("team204")>                                                                 |
| <("player100")-[:follow@0 {}]->("player125")-[:serve@0 {}]->("team204")>                                   |
| <("player100")-[:follow@0 {}]->("player101")-[:serve@0 {}]->("team204")>                                   |
| <("player100")-[:follow@0 {}]->("player101")-[:follow@0 {}]->("player125")-[:serve@0 {}]->("team204")> |
| <("player100")-[:follow@0 {}]->("player101")-[:follow@0 {}]->("player102")-[:serve@0 {}]->("team204")> |
| ...                                                                                                        |
+------------------------------------------------------------------------------------------------------------+
```

FAQ

是否支持 **WHERE** 子句，以实现图遍历过程中的条件过滤?

支持使用 `WHERE` 子句过滤，但只能过滤边属性，不支持过滤点属性。

如示例中的 `WHERE follow.degree is EMPTY or follow.degree >= 0` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.16 查询调优与终止

5.16.1 EXPLAIN 和 PROFILE

`EXPLAIN` 语句输出 nGQL 语句的执行计划，但不会执行 nGQL 语句。

`PROFILE` 语句执行 nGQL 语句，然后输出执行计划和执行概要。用户可以根据执行计划和执行概要优化查询性能。

执行计划

执行计划由悦数图数据库查询引擎中的执行计划器决定。

执行计划器将解析后的 nGQL 语句处理为 `action` 。 `action` 是最小的执行单元。典型的 `action` 包括获取指定点的所有邻居、获取边的属性、根据条件过滤点或边等。每个 `action` 都被分配给一个 `operator` 。

例如 `SHOW TAGS` 语句分为两个 `action` ， `operator` 为 `Start` 和 `ShowTags` 。更复杂的 `GO` 语句可能会被处理成 10 个以上的 `action` 。

语法

- `EXPLAIN`

```
EXPLAIN [format= {"row" | "dot" | "tck"}] <your_nGQL_statement>;
```

- `PROFILE`

```
PROFILE [format= {"row" | "dot" | "tck"}] <your_nGQL_statement>;
```

输出格式

`EXPLAIN` 或 `PROFILE` 语句的输出有三种格式： `row` （默认）、 `dot` 和 `tck` 。用户可以使用 `format` 选项修改输出格式。

Copyright © Yueshu Graph Technology - All rights reserved.

row 格式

row 格式将返回信息输出到一个表格中。

- EXPLAIN

```
nebula> EXPLAIN format="row" SHOW TAGS;
Execution succeeded (time spent 327/892 us)

Execution Plan

-----+----------+--------------+----------------+-------------------------------------------------------------------
| id | name     | dependencies | profiling data | operator info                                                     |
-----+----------+--------------+----------------+-------------------------------------------------------------------
|  1 | ShowTags | 0            |                | outputVar: [{"colNames":[],"name":"__ShowTags_1","type":"DATASET"}] |
|    |          |              |                | inputVar:                                                         |
-----+----------+--------------+----------------+-------------------------------------------------------------------
|  0 | Start    |              |                | outputVar: [{"colNames":[],"name":"__Start_0","type":"DATASET"}]    |
-----+----------+--------------+----------------+-------------------------------------------------------------------
```

- PROFILE

```
nebula> PROFILE format="row" SHOW TAGS;
+--------+
| Name   |
+--------+
| player |
| team   |
+--------+
Got 2 rows (time spent 2038/2728 us)

Execution Plan

-----+----------+--------------+-------------------------------------------------+-------------------------------------------------------------------
| id | name     | dependencies | profiling data                                  | operator info                                                     |
-----+----------+--------------+-------------------------------------------------+-------------------------------------------------------------------
|  1 | ShowTags | 0            | ver: 0, rows: 1, execTime: 42us, totalTime: 1177us | outputVar: [{"colNames":[],"name":"__ShowTags_1","type":"DATASET"}] |
|    |          |              |                                                 | inputVar:                                                         |
-----+----------+--------------+-------------------------------------------------+-------------------------------------------------------------------
|  0 | Start    |              | ver: 0, rows: 0, execTime: 1us, totalTime: 57us | outputVar: [{"colNames":[],"name":"__Start_0","type":"DATASET"}]    |
-----+----------+--------------+-------------------------------------------------+-------------------------------------------------------------------
```

参数	说明
id	operator 的 ID。
name	operator 的名称。
dependencies	当前 operator 所依赖的 operator 的 ID。
profiling data	执行概要文件内容。 ver 表示 operator 的版本； rows 表示 operator 输出结果的行数； execTime 表示执行 action 的时间； totalTime 表示执行 action 的时间、系统调度时间、排队时间的总和。
operator info	operator 的详细信息。

dot 格式

dot 格式将返回 DOT 语言的信息，然后用户可以使用 Graphviz 生成计划图。

> **Note**
>
> Graphviz 是一款开源可视化图工具，可以绘制 DOT 语言脚本描述的图。Graphviz 提供一个在线工具，可以预览 DOT 语言文件，并将它们导出为 SVG 或 JSON 等其他格式。详情请参见 Graphviz Online。

```
nebula> EXPLAIN format="dot" SHOW TAGS;
Execution succeeded (time spent 161/665 us)
Execution Plan
--------------------------------------------------------------------------------------------------------------------------------------------------------------  --------------
  plan
--------------------------------------------------------------------------------------------------------------------------------------------------------------  --------------
  digraph exec_plan {
      rankdir=LR;
      "ShowTags_0"[label="ShowTags_0|outputVar: \[\{\"colNames\":\[\],\"name\":\"__ShowTags_0\",\"type\":\"DATASET\"\}\]\l|inputVar:\l",   shape=Mrecord];
      "Start_2"->"ShowTags_0";
      "Start_2"[label="Start_2|outputVar: \[\{\"colNames\":\[\],\"name\":\"__Start_2\",\"type\":\"DATASET\"\}\]\l|inputVar: \l",   shape=Mrecord];
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  }
------------------
```

5.16.2 终止查询

`KILL QUERY` 命令可以终止正在执行的查询，常用于终止慢查询。

> **Note**
>
> 仅 God 角色权限可以终止任意查询，其他角色只能终止自己的查询。

语法

```
KILL QUERY (session=<session_id>, plan=<plan_id>);
```

- `session_id`： 会话 ID。
- `plan_id`： 执行计划 ID。

会话 ID 和执行计划 ID 可以唯一确定一个查询。二者可以通过 SHOW QUERIES 语句获取。

示例

在一个会话中执行命令终止另一个会话中的查询：

```
nebula> KILL QUERY(SESSION=1625553545984255,PLAN=163);
```

另一个会话中的查询会终止，并返回如下信息：

```
[ERROR (-1005)]: ExecutionPlanId[1001] does not exist in current Session.
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.16.3 终止会话（KILL SESSION）

`KILL SESSION` 语句用于关闭未登出的会话（Session）。

> **Note**

- 只有 `root` 用户可以终止会话。
- 执行 `KILL SESSION` 命令后，所有的 Graph 服务同步最新的会话信息需要等待 `2* session_reclaim_interval_secs` 秒，默认等待 `120` 秒。

语法

`KILL SESSION` 语句支持终止单个和多个会话，语法如下：

- 终止单个会话

```
KILL {SESSION|SESSIONS} <SessionId>
```

- `{SESSION|SESSIONS}`： 支持 `SESSION` 和 `SESSIONS` 的写法。
- `<SessionId>`： 指会话 Session 的 ID。可执行 SHOW SESSIONS 命令查看会话 ID。

- 终止多个会话

```
SHOW SESSIONS
| YIELD $-.SessionId AS sid [WHERE <filter_clause>]
| KILL {SESSION|SESSIONS} $-.sid
```

> **Note**
>
> `KILL SESSION` 语句支持管道操作，即将 `SHOW SESSIONS` 语句与 `KILL SESSION` 语句结合使用，以终止多个会话。

- `[WHERE <filter_clause>]`：
- 可选项，使用 `WHERE` 子句过滤会话； `<filter_expression>` 指滤过表达式，例如 `WHERE $-.CreateTime < datetime("2022-12-14T18:00:00")` 。如果不加该选项，则关闭所有当前会话。
- `WHERE` 子句中支持的过滤项有： `SessionId` 、 `UserName` 、 `SpaceName` 、 `CreateTime` 、 `UpdateTime` 、 `GraphAddr` 、 `Timezone` 、 `ClientIp` 。可以执行 SHOW SESSIONS 命令查看这些过滤项的含义。
- `{SESSION|SESSIONS}`： 支持 `SESSION` 和 `SESSIONS` 的写法。

> **Caution**
>
> 请谨慎使用过滤条件以防误删会话。

示例

- 终止单个会话。

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> KILL SESSION 1672887983842984
```

- 终止多个会话。
- 终止创建时间小于 `2023-01-05T18:00:00` 的所有会话。

```
nebula> SHOW SESSIONS | YIELD $-.SessionId AS sid WHERE $-.CreateTime < datetime("2023-01-05T18:00:00") | KILL SESSIONS $-.sid
```

- 终止创建时间最早的两个会话。

```
nebula> SHOW SESSIONS | YIELD $-.SessionId AS sid, $-.CreateTime as CreateTime | ORDER BY $-.CreateTime ASC | LIMIT 2 | KILL SESSIONS $-.sid
```

- 终止用户名为 `session_user1` 创建的所有会话。

```
nebula> SHOW SESSIONS | YIELD $-.SessionId as sid WHERE $-.UserName == "session_user1" | KILL SESSIONS $-.sid
```

- 终止所有会话

```
nebula> SHOW SESSIONS | YIELD $-.SessionId as sid | KILL SESSION $-.sid

// 或者
nebula> SHOW SESSIONS | KILL SESSIONS $-.SessionId
```

Caution

终止所有会话时，当前会话也会被终止。请谨慎使用。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

5.17 作业管理

在 Storage 服务上长期运行的任务称为作业，例如 `COMPACT` 、 `FLUSH` 和 `STATS` 。 如果图空间的数据量很大，这些作业可能耗时很长。作业管理可以帮助执行、查看、停止和恢复作业。

Note

所有作业管理命令都需要先选择图空间后才能执行。

5.17.1 SUBMIT JOB BALANCE DATA

Caution

- 作业执行前，建议创建快照。
- 作业执行过程中，不要执行其他作业，例如 `SUBMIT JOB STATS` 、 `REBUILD INDEX` 等。
- 作业执行过程中，不建议大批量写入和读取数据。

`SUBMIT JOB BALANCE DATA` 语句会在当前图空间内启动任务均衡分布分片。该命令会返回任务 ID。

示例：

```
nebula> SUBMIT JOB BALANCE DATA;
+------------+
| New Job Id |
+------------+
| 28         |
+------------+
```

5.17.2 SUBMIT JOB BALANCE DATA REMOVE

`SUBMIT JOB BALANCE DATA REMOVE <ip:port> [,<ip>:<port> ...]` 语句会启动任务迁空当前图空间指定的 Storage 服务中的分片。端口默认为 `9779` 。该命令会返回任务 ID。

Caution

- 执行该命令迁移指定 Storage 节点中的分片数据前，需要确保其他 Storage 节点数量足够以满足设置的副本数。例如，如果设置了副本数为 3，那么在执行该命令前，需要确保其他 Storage 节点数量大于等于 3。
- 如果待迁移的 Storage 节点中有多个图空间的分片，需要分别在每个图空间内执行该命令以迁移该 Storage 节点中的所有分片。

示例：

```
nebula> SUBMIT JOB BALANCE DATA REMOVE 192.168.8.100:9779;
+------------+
| New Job Id |
+------------+
| 29         |
+------------+
```

5.17.3 SUBMIT JOB BALANCE LEADER

`SUBMIT JOB BALANCE LEADER` 语句会启动任务均衡分布所有图空间中的 leader。该命令会返回任务 ID。

示例：

```
nebula> SUBMIT JOB BALANCE LEADER;
+------------+
| New Job Id |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+------------+
| 33         |
+------------+
```

5.17.4 SUBMIT JOB COMPACT

`SUBMIT JOB COMPACT` 语句会在当前图空间内触发 RocksDB 的长耗时 `compact` 操作。

`compact` 配置详情请参见 Storage 服务配置。

示例：

```
nebula> SUBMIT JOB COMPACT;
+------------+
| New Job Id |
+------------+
| 40         |
+------------+
```

5.17.5 SUBMIT JOB FLUSH

`SUBMIT JOB FLUSH` 语句将当前图空间内存中的 RocksDB memfile 写入硬盘。

示例：

```
nebula> SUBMIT JOB FLUSH;
+------------+
| New Job Id |
+------------+
| 96         |
+------------+
```

5.17.6 SUBMIT JOB STATS

`SUBMIT JOB STATS` 语句会在当前图空间内启动一个作业，该作业对当前图空间进行统计。作业完成后，用户可以使用 `SHOW STATS` 语句列出统计结果。详情请参见 SHOW STATS。

> **Note**
>
> 如果存储在悦数图数据库中的数据有变化，为了获取最新的统计结果，请重新执行 `SUBMIT JOB STATS` 。

示例：

```
nebula> SUBMIT JOB STATS;
+------------+
| New Job Id |
+------------+
| 9          |
+------------+
```

5.17.7 SUBMIT JOB DOWNLOAD/INGEST

`SUBMIT JOB DOWNLOAD HDFS` 和 `SUBMIT JOB INGEST` 命令用于将 SST 文件导入悦数图数据库。详情参见导入 SST 文件数据。

`SUBMIT JOB DOWNLOAD HDFS` 语句会下载指定的 HDFS 上的 SST 文件。

`SUBMIT JOB INGEST` 语句会将下载的 SST 文件导入图空间。

示例：

```
nebula> SUBMIT JOB DOWNLOAD HDFS "hdfs://192.168.10.100:9000/sst";
+------------+
| New Job Id |
+------------+
| 10         |
+------------+

nebula> SUBMIT JOB INGEST;
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+------------+
| New Job Id |
+------------+
| 11         |
+------------+
```

5.17.8 SHOW JOB

Meta 服务将 `SUBMIT JOB` 请求解析为多个任务，然后分配给进程 nebula-storaged。`SHOW JOB <job_id>` 语句显示当前图空间内指定作业和相关任务的信息。

`job_id` 在执行 `SUBMIT JOB` 语句时会返回。

示例：

```
nebula> SHOW JOB 9;
+---------------+-----------------+------------+----------------------------+----------------------------+-------------+
| Job Id(TaskId) | Command(Dest)   | Status     | Start Time                 | Stop Time                  | Error Code  |
+---------------+-----------------+------------+----------------------------+----------------------------+-------------+
| 8             | "STATS"         | "FINISHED" | 2022-10-18T08:14:45.000000 | 2022-10-18T08:14:45.000000 | "SUCCEEDED" |
| 0             | "192.168.8.129" | "FINISHED" | 2022-10-18T08:14:45.000000 | 2022-10-18T08:15:13.000000 | "SUCCEEDED" |
| "Total:1"     | "Succeeded:1"   | "Failed:0" | "In Progress:0"            | ""                         | ""          |
+---------------+-----------------+------------+----------------------------+----------------------------+-------------+
```

参数	说明
`Job Id(TaskId)`	第一行显示作业 ID，最后一行显示作业相关的任务总数，其他行显示作业相关的任务 ID。
`Command(Dest)`	第一行显示执行的作业命令名称，最后一行显示作业相关的成功的任务数。其他行显示任务对应的 nebula-storaged 进程。
`Status`	显示作业或任务的状态，最后一行显示作业相关的失败的任务数。详情请参见作业状态。
`Start Time`	显示作业或任务开始执行的时间，最后一行显示作业相关的正在进行的任务数。
`Stop Time`	显示作业或任务结束执行的时间，结束后的状态包括 `FINISHED` 、 `FAILED` 或 `STOPPED` 。
`Error Code`	显示作业或任务的错误码。

作业状态

作业状态的说明如下。

状态	说明
QUEUE	作业或任务在等待队列中。此阶段 `Start Time` 为空。
RUNNING	作业或任务在执行中。 `Start Time` 为该阶段的起始时间。
FINISHED	作业或任务成功完成。 `Stop Time` 为该阶段的起始时间。
FAILED	作业或任务失败。 `Stop Time` 为该阶段的起始时间。
STOPPED	作业或任务停止。 `Stop Time` 为该阶段的起始时间。
REMOVED	作业或任务被删除。

状态转换的说明如下。

```
Queue -- running -- finished -- removed
     \          \                /
      \          \ -- failed -- /
       \          \            /
        \ ---------- stopped -/
```

5.17.9 SHOW JOBS

`SHOW JOBS` 语句列出当前图空间内所有未过期的作业。

Copyright © Yueshu Graph Technology - All rights reserved.

作业的默认过期时间为一周。如果需要修改过期时间，请修改 Meta 服务的参数 `job_expired_secs` 。详情请参见 Meta 服务配置。

示例：

```
nebula> SHOW JOBS;
+--------+---------------------+------------+----------------------------+----------------------------+
| Job Id | Command             | Status     | Start Time                 | Stop Time                  |
+--------+---------------------+------------+----------------------------+----------------------------+
| 34     | "STATS"             | "FINISHED" | 2021-11-01T03:32:27.000000 | 2021-11-01T03:32:27.000000 |
| 33     | "FLUSH"             | "FINISHED" | 2021-11-01T03:32:15.000000 | 2021-11-01T03:32:15.000000 |
| 32     | "COMPACT"           | "FINISHED" | 2021-11-01T03:32:06.000000 | 2021-11-01T03:32:06.000000 |
| 31     | "REBUILD_TAG_INDEX" | "FINISHED" | 2021-10-29T05:39:16.000000 | 2021-10-29T05:39:17.000000 |
| 10     | "COMPACT"           | "FINISHED" | 2021-10-26T02:27:05.000000 | 2021-10-26T02:27:05.000000 |
+--------+---------------------+------------+----------------------------+----------------------------+
```

5.17.10 STOP JOB

`STOP JOB <job_id>` 语句可以停止当前图空间内未完成的作业。

示例：

```
nebula> STOP JOB 22;
+---------------+
| Result        |
+---------------+
| "Job stopped" |
+---------------+
```

5.17.11 RECOVER JOB

`RECOVER JOB [<job_id>]` 语句可以重新执行当前图空间内状态为 `FAILED` 、 `STOPPED` 的作业，未指定 `<job_id>` 时，会从最早的作业开始尝试重新执行，并返回已恢复的作业数量。

示例：

```
nebula> RECOVER JOB;
+-------------------+
| Recovered job num |
+-------------------+
| 5 job recovered   |
+-------------------+
```

5.17.12 FAQ

如何排查作业问题?

`SUBMIT JOB` 操作使用的是 HTTP 端口，请检查 Storage 服务机器上的 HTTP 端口是否正常工作。用户可以执行如下命令调试：

```
curl "http://{storaged-ip}:19779/admin?space={space_name}&op=compact"
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6. 安装部署

6.1 准备安装和运行悦数图数据库的环境

本文介绍编译、安装悦数图数据库的要求和建议，以及如何预估集群运行所需的资源。

6.1.1 关于存储硬件

悦数图数据库是针对 NVMe SSD 进行设计和实现的，所有默认参数都是基于 SSD 设备进行调优，要求极高的 IOPS 和极低的 Latency。

- 不建议使用 HDD；因为其 IOPS 性能差，随机寻道延迟高；会遇到大量问题。
- 不要使用远端存储设备（如 NAS 或 SAN），不要外接基于 HDFS 或者 Ceph 的虚拟硬盘。
- 不要使用磁盘阵列（RAID）。
- 使用本地 SSD 设备；或 AWS Provisioned IOPS SSD 或等价云产品。

6.1.2 关于 CPU 架构

Enterpriseonly

企业版支持在 ARM 架构（包括 Apple Mac M1 和华为鲲鹏）上运行。访问官网获取商业支持。

Note

从 3.0.2 开始，悦数图数据库在 Docker Hub 上的 Docker 支持 ARM64 架构。社区用户可以在 ARM macOS 的 Docker Desktop 上或者 ARM Linux Server 上运行容器化的悦数图数据库。

6.1.3 测试环境要求

硬件要求

单个机器的硬件要求如下。

类型	要求
CPU 架构	x86_64
CPU 核数	4
内存	8 GB
硬盘	100 GB，SSD

操作系统要求

当前仅支持在 Linux 系统中安装悦数图数据库，建议在测试环境中使用内核版本为 `3.9` 及以上版本的 Linux 系统。

Copyright © Yueshu Graph Technology - All rights reserved.

服务架构建议

进程	建议数量
metad（meta 数据服务进程）	1
storaged（存储服务进程）	≥1
graphd（查询引擎服务进程）	≥1

例如单机测试环境，用户可以在机器上部署 1 个 metad、1 个 storaged 和 1 个 graphd 进程。

对于更常见的测试环境，例如三台机器构成的集群，用户可以按照如下方案部署悦数图数据库。

机器名称	**metad** 进程数量	**storaged** 进程数量	**graphd** 进程数量
A	1	1	1
B	-	1	1
C	-	1	1

6.1.4 生产环境运行要求

硬件要求

单个机器的硬件要求如下。

类型	要求
CPU 架构	x86_64
CPU 核数	48
内存	256 GB
硬盘	2 * 1.6 TB，NVMe SSD

操作系统要求

当前仅支持在 Linux 系统中安装悦数图数据库，建议在生产环境中使用内核版本为 `3.9` 及以上版本的 Linux 系统。

用户可以通过调整一些内核参数来提高悦数图数据库性能，详情请参见内核配置。

服务架构建议

Danger

不要跨机房部署单个集群（企业版支持跨机房集群间同步）。

进程	数量
metad（meta 数据服务进程）	3
storaged（存储服务进程）	≥3
graphd（查询引擎服务进程）	≥3

有且仅有 3 个 metad 进程，每个 metad 进程会自动创建并维护 meta 数据的一个副本。

storaged 进程的数量不会影响图空间副本的数量。

Copyright © Yueshu Graph Technology - All rights reserved.

用户可以在一台机器上部署多个不同进程，例如五台机器构成的集群，用户可以按照如下方案部署悦数图数据库。

机器名称	**metad** 进程数量	**storaged** 进程数量	**graphd** 进程数量
A	1	1	1
B	1	1	1
C	1	1	1
D	-	1	1
E	-	1	1

6.1.5 悦数图数据库资源要求

用户可以预估一个 3 副本悦数图数据库集群所需的内存、硬盘空间和分区数量。

资源	单位	计算公式	说明
硬盘空间	Bytes	`点和边的总数` * `属性的平均字节大小` * 7.5 * 120%	由于边存在存储放大现象，所以需要 `点和边的总数` * `属性的平均字节大小` * 7.5 的空间，详情请参见切边与存储放大。
内存	Bytes	[`点和边的总数` * 16 + `RocksDB 实例数量` * (`write_buffer_size` * `max_write_buffer_number`) + `块缓存大小`] * 120%	`点和边的总数` * 16 是 BloomFilter 需要占用的内存空间，`write_buffer_size` 和 `max_write_buffer_number` 是 RocksDB 内存相关参数，详情请参见 MemTable。块缓存大小请参见 Memory usage in RocksDB。
分区数量	-	`集群硬盘数量` * `disk_partition_num_multiplier`	`disk_partition_num_multiplier` 是一个用于衡量硬盘性能的整数，取值范围 2~20。建议在计算 SSD 硬盘的分区数量时使用 20 做参数值，HDD 硬盘使用 2。

- 问题 1：为什么硬盘空间的预估公式中需要乘以 7.5?

  答：因为根据测试值，相比原始数据文件（csv），单副本所占用的空间约为之前的 2.5 倍；另外索引会另外占用硬盘空间，每个被索引的点或边占用 16 字节的内存，索引的占用硬盘空间可按经验估算为：被索引的点或边总数 * 50 字节。

- 问题 2：为什么磁盘空间和内存都要乘以 120%?

  答：额外的 20% 用于缓冲。

- 问题 3：如何获取 RocksDB 实例数量?

  答：对于社区版悦数图数据库，每个图空间对应一个 RocksDB 实例，并且 `--data_path` 选项（`etc` 目录下的 `nebula-storaged.conf` 文件中）中的每个目录对应一个 RocksDB 实例。即，RocksDB 实例数量 = 图空间总数 * 目录总数。对于企业版悦数图数据库，一个分区对应一个 RocksDB 实例。

Note

用户可以在配置文件 `nebula-storaged.conf` 中添加 `--enable_partitioned_index_filter=true` 来降低 bloom 过滤器占用的内存大小，但是在某些随机寻道（random-seek）的情况下，可能会降低读取性能。

Copyright © Yueshu Graph Technology - All rights reserved.

Caution

每个 RocksDB 实例即使还未写入任何数据时，仍会占用 70M 左右的磁盘空间。一个分区对应一个 RocksDB 实例，当分区设置特别多时，例如 100，图空间创建后即占用了大量磁盘空间。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.2 单机安装

6.2.1 使用 RPM 或 DEB 包安装悦数图数据库

RPM 和 DEB 是 Linux 系统下常见的两种安装包格式，本文介绍如何使用 RPM 或 DEB 文件在一台机器上快速安装悦数图数据库。

> **Note**
>
> 部署悦数图数据库集群的方式参见使用 RPM/DEB 包部署集群。

前提条件

- 安装 `wget` 工具。
- 已在 LM 中加载 License Key。

下载安装包

联系我们获取悦数图数据库安装包。

> **Note**
>
> - 当前仅支持在 Linux 系统下安装悦数图数据库，且仅支持 CentOS 7.x、CentOS 8.x、Ubuntu 16.04、Ubuntu 18.04、Ubuntu 20.04 操作系统。
> - 如果用户使用的是国产化的 Linux 操作系统，请安装企业版悦数图数据库。

安装悦数图数据库

- 安装 RPM 包

```
$ sudo rpm -ivh --prefix=<installation_path> <package_name>
```

`--prefix` 为可选项，用于指定安装路径。如不设置，系统会将悦数图数据库安装到默认路径 `/usr/local/nebula/` 。

例如，要在默认路径下安装3.5.1版本的 RPM 包，运行如下命令：

```
sudo rpm -ivh yueshu-graph-3.5.1.el7.x86_64.rpm
```

- 安装 DEB 包

```
$ sudo dpkg -i <package_name>
```

> **Note**
>
> 使用 DEB 包安装悦数图数据库时不支持自定义安装路径。默认安装路径为 `/usr/local/nebula/` 。

例如安装3.5.1版本的 DEB 包：

```
sudo dpkg -i yueshu-graph-3.5.1.ubuntu1804.amd64.deb
```


Copyright © Yueshu Graph Technology - All rights reserved.

配置 **License** 管理工具（**LM**）地址

1. 在悦数图数据库的 Meta 服务配置文件（`nebula-metad.conf`）中，设置 `license_manager_url` 的值为许可证管理工具所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。
2. 将 Meta、Storage 和 Graph 服务的配置文件（`nebula-metad.conf`、`nebula-graphd.conf`、`nebula-storaged.conf`）中的所有 `local_ip`（默认 `127.0.0.1`）替换为各服务所在主机的真实 IP，以及将 `meta_server_addrs` 地址替换为 Meta 服务所在主机 IP 地址和端口号 `9559`。

后续操作

- 启动悦数图数据库
- 连接悦数图数据库

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.2.2 使用 tar.gz 文件安装悦数图数据库

用户可以下载打包好的 tar.gz 文件快速安装悦数图数据库。

Note

- 悦数图数据库从 2.6.0 版本起提供 tar.gz 文件。
- 当前仅支持在 Linux 系统下安装悦数图数据库，且仅支持 CentOS 7.x、CentOS 8.x、Ubuntu 16.04、Ubuntu 18.04、Ubuntu 20.04 操作系统。
- 如果用户使用的是国产化的 Linux 操作系统，请安装悦数图数据库。

前提条件

- 已在 LM 中加载 License Key

操作步骤

1. 联系我们获取悦数图数据库安装包。
2. 解压 tar.gz 文件到悦数图数据库安装目录。

```
tar -xvzf <tar.gz_file_name> -C <install_path>
```

- `tar.gz_file_name` 表示 tar.gz 文件的名称。
- `install_path` 表示安装路径。

例如：

```
tar -xvzf yueshu-graph-3.5.1.el7.x86_64.tar.gz -C /home/joe/nebula/install
```

3. 修改配置文件名以应用配置。

进入解压出的目录，将子目录 `etc` 中的文件 `nebula-graphd.conf.default` 、 `nebula-metad.conf.default` 和 `nebula-storaged.conf.default` 重命名，删除 `.default` ，即可应用悦数图数据库的默认配置。

4. 配置 License 管理工具 License Manager 地址。

a. 在悦数图数据库的 Meta 服务配置文件（ `nebula-metad.conf` ）中，设置 `license_manager_url` 的值为许可证管理工具所在的主机 IP 和端口号 `9119` ，例如 `192.168.8.100:9119` 。

b. 将 Meta、Storage 和 Graph 服务的配置文件（ `nebula-metad.conf` 、 `nebula-graphd.conf` 、 `nebula-storaged.conf` ）中的所有 `local_ip` （默认 `127.0.0.1` ）替换为各服务所在主机的真实 IP，以及将 `meta_server_addrs` 地址替换为 Meta 服务所在主机 IP 地址和端口号 `9559` 。

Note

如需修改更多配置，参见配置管理。

至此，悦数图数据库安装完毕。

下一步

管理悦数图数据库服务

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.3 使用 RPM/DEB 包部署悦数图数据库多机集群

本文介绍通过 RPM 或 DEB 文件部署集群的示例。

Note

用户还可以通过官方工具部署悦数图数据库多机集群。详情参见使用生态工具安装集群。

6.3.1 部署方案

机器名称	**IP** 地址	**graphd** 进程数量	**storaged** 进程数量	**metad** 进程数量
A	192.168.10.111	1	1	1
B	192.168.10.112	1	1	1
C	192.168.10.113	1	1	1
D	192.168.10.114	1	1	-
E	192.168.10.115	1	1	-

6.3.2 前提条件

- 准备 5 台用于部署集群的机器。
- 在集群中通过 NTP 服务同步时间。
- 已在 LM 中加载 License Key。

6.3.3 手动部署流程

安装悦数图数据库

在集群的每一台服务器上都安装悦数图数据库，安装后暂不需要启动服务。安装方式请参见：

- 使用 RPM 或 DEB 包安装悦数图数据库

修改配置文件

修改每个服务器上的悦数图数据库配置文件。

悦数图数据库的所有配置文件均位于安装目录的 `etc` 目录内，包括 `nebula-graphd.conf` 、 `nebula-metad.conf` 和 `nebula-storaged.conf` ，用户可以只修改所需服务的配置文件。各个机器需要修改的配置文件如下。

机器名称	待修改配置文件
A	`nebula-graphd.conf` 、 `nebula-storaged.conf` 、 `nebula-metad.conf`
B	`nebula-graphd.conf` 、 `nebula-storaged.conf` 、 `nebula-metad.conf`
C	`nebula-graphd.conf` 、 `nebula-storaged.conf` 、 `nebula-metad.conf`
D	`nebula-graphd.conf` 、 `nebula-storaged.conf`
E	`nebula-graphd.conf` 、 `nebula-storaged.conf`

用户可以参考如下配置文件的内容，仅展示集群通信的部分设置，未展示的内容为默认设置，便于用户了解集群间各个服务器的关系。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

主要修改的配置是`meta_server_addrs`，所有配置文件都需要填写所有 Meta 服务的 IP 地址和端口，同时需要修改`local_ip`为机器本身的联网 IP 地址。配置参数的详细说明请参见：

- Meta 服务配置
- Graph 服务配置
- Storage 服务配置

Copyright © Yueshu Graph Technology - All rights reserved.

Note

在所有的 Meta 服务配置文件中，需要设置 `license_manager_url` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。

- 机器 A 配置

- `nebula-graphd.conf`

```
########## networking ##########
# Comma separated Meta Server Addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-graphd process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.111
# Network device to listen on
--listen_netdev=any
# Port to listen on
--port=9669
```

- `nebula-storaged.conf`

```
########## networking ##########
# Comma separated Meta server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-storaged process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.111
# Storage daemon listening port
--port=9779
```

- `nebula-metad.conf`

```
########## networking ##########
# Comma separated Meta Server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-metad process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.111
# Meta daemon listening port
--port=9559
```


Copyright © Yueshu Graph Technology - All rights reserved.

- 机器 B 配置

- `nebula-graphd.conf`

```
########## networking ##########
# Comma separated Meta Server Addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-graphd process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.112
# Network device to listen on
--listen_netdev=any
# Port to listen on
--port=9669
```

- `nebula-storaged.conf`

```
########## networking ##########
# Comma separated Meta server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-storaged process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.112
# Storage daemon listening port
--port=9779
```

- `nebula-metad.conf`

```
########## networking ##########
# Comma separated Meta Server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-metad process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.112
# Meta daemon listening port
--port=9559
```


Copyright © Yueshu Graph Technology - All rights reserved.

- 机器 C 配置

- `nebula-graphd.conf`

```
########## networking ##########
# Comma separated Meta Server Addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-graphd process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.113
# Network device to listen on
--listen_netdev=any
# Port to listen on
--port=9669
```

- `nebula-storaged.conf`

```
########## networking ##########
# Comma separated Meta server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-storaged process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.113
# Storage daemon listening port
--port=9779
```

- `nebula-metad.conf`

```
########## networking ##########
# Comma separated Meta Server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-metad process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.113
# Meta daemon listening port
--port=9559
```

- 机器 D 配置

- `nebula-graphd.conf`

```
########## networking ##########
# Comma separated Meta Server Addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-graphd process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.114
# Network device to listen on
--listen_netdev=any
# Port to listen on
--port=9669
```

- `nebula-storaged.conf`

```
########## networking ##########
# Comma separated Meta server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-storaged process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.114
# Storage daemon listening port
--port=9779
```


Copyright © Yueshu Graph Technology - All rights reserved.

- 机器 E 配置

- `nebula-graphd.conf`

```
########## networking ##########
# Comma separated Meta Server Addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-graphd process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.115
# Network device to listen on
--listen_netdev=any
# Port to listen on
--port=9669
```

- `nebula-storaged.conf`

```
########## networking ##########
# Comma separated Meta server addresses
--meta_server_addrs=192.168.10.111:9559,192.168.10.112:9559,192.168.10.113:9559
# Local IP used to identify the nebula-storaged process.
# Change it to an address other than loopback if the service is distributed or
# will be accessed remotely.
--local_ip=192.168.10.115
# Storage daemon listening port
--port=9779
```

启动集群

依次启动各个服务器上的对应进程。

机器名称	待启动的进程
A	graphd、storaged、metad
B	graphd、storaged、metad
C	graphd、storaged、metad
D	graphd、storaged
E	graphd、storaged

启动悦数图数据库进程的命令如下：

```
sudo /usr/local/nebula/scripts/nebula.service start <metad|graphd|storaged|all>
```

Note

- 确保每个服务器中的对应进程都已启动，否则服务将启动失败。
- 当需都启动 graphd、storaged 和 metad 时，可以用 all 代替。
- `/usr/local/nebula` 是悦数图数据库的默认安装路径，如果修改过安装路径，请使用实际路径。更多启停服务的内容，请参见管理悦数图数据库服务。

检查集群

安装原生 CLI 客户端 NebulaGraph Console，然后连接任何一个已启动 graphd 进程的机器，添加 Storage 主机，然后执行命令 `SHOW HOSTS` 检查集群状态。例如：

```
$ ./nebula-console --addr 192.168.10.111 --port 9669 -u root -p nebula

2021/05/25 01:41:19 [INFO] connection pool is initialized successfully
Welcome to NebulaGraph!

> ADD HOSTS 192.168.10.111:9779, 192.168.10.112:9779, 192.168.10.113:9779, 192.168.10.114:9779, 192.168.10.115:9779;
> SHOW HOSTS;
+-----------------+------+----------+-------------+----------------------+------------------------+----------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| Host             | Port | Status   | Leader count | Leader distribution  | Partition distribution | Version              |
+------------------+------+----------+--------------+----------------------+------------------------+----------------------+
| "192.168.10.111" | 9779 | "ONLINE" | 0            | "No valid partition" | "No valid partition"   | "3.5.1" |
| "192.168.10.112" | 9779 | "ONLINE" | 0            | "No valid partition" | "No valid partition"   | "3.5.1" |
| "192.168.10.113" | 9779 | "ONLINE" | 0            | "No valid partition" | "No valid partition"   | "3.5.1" |
| "192.168.10.114" | 9779 | "ONLINE" | 0            | "No valid partition" | "No valid partition"   | "3.5.1" |
| "192.168.10.115" | 9779 | "ONLINE" | 0            | "No valid partition" | "No valid partition"   | "3.5.1" |
+------------------+------+----------+--------------+----------------------+------------------------+----------------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.4 使用生态工具安装悦数图数据库

用户可以使用以下生态工具安装悦数图数据库:

- 悦数运维监控
- NebulaGraph Operator

6.4.1 安装详情

- 使用悦数运维监控 安装悦数图数据库的详情，参见创建集群。
- 使用 **NebulaGraph Operator** 安装悦数图数据库的详情，参见使用 Kubectl 部署悦数图数据库集群 或 使用 Helm 部署悦数图数据库集群。

Note

联系销售（inqury@vesoft.com）获取悦数图数据库的安装包。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.5 管理悦数图数据库服务

悦数图数据库支持使用脚本管理服务和使用 systemd 管理服务。

Caution

这两种方式互不兼容，选择使用其中一种。

6.5.1 使用脚本管理服务

使用脚本 `nebula.service` 管理服务，包括启动、停止、重启、中止和查看。

Note

`nebula.service` 的默认路径是 `/usr/local/nebula/scripts`，如果修改过安装路径，请使用实际路径。

语法

```
$ sudo /usr/local/nebula/scripts/nebula.service
[-v] [-c <config_file_path>]
<start | stop | restart | kill | status>
<metad | graphd | storaged | all>
```

参数	说明
`-v`	显示详细调试信息。
`-c`	指定配置文件路径，默认路径为 `/usr/local/nebula/etc/` 。
`start`	启动服务。
`stop`	停止服务。
`restart`	重启服务。
`kill`	中止服务。
`status`	查看服务状态。
`metad`	管理 Meta 服务。
`graphd`	管理 Graph 服务。
`storaged`	管理 Storage 服务。
`all`	管理所有服务。

6.5.2 使用 systemd 管理服务

为方便使用，悦数图数据库企业版支持用 systemd 管理服务，通过 `systemctl` 启动、停止、重启和查看服务。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

- 安装悦数图数据库企业版后，systemd 所需的 `.service` 文件在安装目录的 `etc/unit` 目录内，使用 RPM/DEB 包安装的 NebulaGraph，会自动将这些 `.service` 文件放入 `/usr/lib/systemd/system` 目录内，并且 `ExecStart` 也会根据指定的悦数图数据库安装路径进行生成，因此可以直接使用 `systemctl` 命令。
- 对于使用悦数运维监控安装的悦数图数据库，不支持使用 `systemctl` 管理服务。
- 对于其他方式安装的企业版 NebulaGraph，需要用户手动将 `.service` 文件移动到 `/usr/lib/systemd/system` 目录内，并修改 `.service` 文件内的 `ExecStart` 的文件路径，才可以正常使用 `systemctl` 命令。

语法

```
$ systemctl <start | stop | restart | status > <nebula | nebula-metad | nebula-graphd | nebula-storaged>
```

参数	说明
`start`	启动服务。
`stop`	停止服务。
`restart`	重启服务。
`status`	查看服务状态。
`nebula`	管理所有服务。
`nebula-metad`	管理 Meta 服务。
`nebula-graphd`	管理 Graph 服务。
`nebula-storaged`	管理 Storage 服务。

6.5.3 启动悦数图数据库服务

执行如下命令启动服务：

```
$ sudo /usr/local/nebula/scripts/nebula.service start all
[INFO] Starting nebula-metad...
[INFO] Done
[INFO] Starting nebula-graphd...
[INFO] Done
[INFO] Starting nebula-storaged...
[INFO] Done
```

或者：

```
$ systemctl start nebula
```

如果需要设置开机自动启动，命令如下：

```
$ systemctl enable nebula
```

6.5.4 停止悦数图数据库服务

Danger

请勿使用 `kill -9` 命令强制终止进程，否则可能较小概率出现数据丢失。

执行如下命令停止悦数图数据库服务：

```
$ sudo /usr/local/nebula/scripts/nebula.service stop all
[INFO] Stopping nebula-metad...
[INFO] Done
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
[INFO] Stopping nebula-graphd...
[INFO] Done
[INFO] Stopping nebula-storaged...
[INFO] Done
```

或者：

```
$ systemctl stop nebula
```

6.5.5 查看悦数图数据库服务

执行如下命令查看悦数图数据库服务状态：

```
$ sudo /usr/local/nebula/scripts/nebula.service status all
```

- 如果返回如下结果，表示悦数图数据库服务正常运行。

```
[INFO] nebula-metad(33fd35e): Running as 29020, Listening on 9559
[INFO] nebula-graphd(33fd35e): Running as 29095, Listening on 9669
[WARN] nebula-storaged after v3.0.0 will not start service until it is added to cluster.
[WARN] See Manage Storage hosts:ADD HOSTS in https://docs.nebula-graph.io/
[INFO] nebula-storaged(33fd35e): Running as 29147, Listening on 9779
```

> **Note**
>
> 正常启动悦数图数据库后，`nebula-storaged` 进程的端口显示红色。这是因为 `nebula-storaged` 在启动流程中会等待 `nebula-metad` 添加当前 Storage 服务，当前 Storage 服务收到 Ready 信号后才会正式启动服务。从 3.0.0 版本开始，在配置文件中添加的 Storage 节点无法直接读写，配置文件的作用仅仅是将 Storage 节点注册至 Meta 服务中。必须使用 `ADD HOSTS` 命令后，才能正常读写 Storage 节点。更多信息，参见管理 Storage 主机。

- 如果返回类似如下结果，表示悦数图数据库服务异常，可以根据异常服务信息进一步排查，或者在 悦数图数据库社区寻求帮助。

```
[INFO] nebula-metad: Running as 25600, Listening on 9559
[INFO] nebula-graphd: Exited
[INFO] nebula-storaged: Running as 25646, Listening on 9779
```

也可以使用 `systemctl` 命令查看悦数图数据库服务状态：

```
$ systemctl status nebula
● nebula.service
   Loaded: loaded (/usr/lib/systemd/system/nebula.service; disabled; vendor preset: disabled)
   Active: active (exited) since — 2022-03-28 04:13:24 UTC; 1h 47min ago
  Process: 21772 ExecStart=/usr/local/ent-nightly/scripts/nebula.service start all (code=exited, status=0/SUCCESS)
 Main PID: 21772 (code=exited, status=0/SUCCESS)
    Tasks: 325
   Memory: 424.5M
   CGroup: /system.slice/nebula.service
           ├─21789 /usr/local/ent-nightly/bin/nebula-metad --flagfile /usr/local/ent-nightly/etc/nebula-metad.conf
           ├─21827 /usr/local/ent-nightly/bin/nebula-graphd --flagfile /usr/local/ent-nightly/etc/nebula-graphd.conf
           └─21900 /usr/local/ent-nightly/bin/nebula-storaged --flagfile /usr/local/ent-nightly/etc/nebula-storaged.conf

3月 28 04:13:24 xxxxxx systemd[1]: Started nebula.service.
...
```

悦数图数据库服务由 Meta 服务、Graph 服务和 Storage 服务共同提供，这三种服务的配置文件都保存在安装目录的 `etc` 目录内，默认路径为 `/usr/local/nebula/etc/`，用户可以检查相应的配置文件排查问题。

6.5.6 下一步

- 连接悦数图数据库

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.6 连接悦数图数据库服务

本文介绍如何使用原生命令行客户端 Nebula Console 连接悦数图数据库。

Caution

首次连接到悦数图数据库后，必须先注册 Storage 服务，才能正常查询数据。

悦数图数据库支持多种类型的客户端，包括命令行客户端、可视化界面客户端和流行编程语言客户端。详情参见客户端列表。

6.6.1 前提条件

- 悦数图数据库服务已启动。
- 运行 Nebula Console 的机器和运行悦数图数据库的服务器网络互通。
- Nebula Console 的版本兼容悦数图数据库的版本。

Note

版本相同的 Nebula Console 和悦数图数据库兼容程度最高，版本不同的 Nebula Console 连接悦数图数据库时，可能会有兼容问题，或者无法连接并报错 `incompatible version between client and server` 。

6.6.2 操作步骤

1. 在 Nebula Console 下载页面，确认需要的版本，单击 **Assets**。

Note

建议选择最新版本。

2. 在 **Assets** 区域找到机器运行所需的二进制文件，下载文件到机器上。
3. （可选）为方便使用，重命名文件为 `nebula-console` 。

Note

在 Windows 系统中，请重命名为 `nebula-console.exe` 。

4. 在运行 Nebula Console 的机器上执行如下命令，为用户授予 nebula-console 文件的执行权限。

Note

Windows 系统请跳过此步骤。

```
$ chmod 111 nebula-console
```

5. 在命令行界面中，切换工作目录至 nebula-console 文件所在目录。

Copyright © Yueshu Graph Technology - All rights reserved.

6. 执行如下命令连接悦数图数据库。

- Linux 或 macOS

```
$ ./nebula-console -addr <ip> -port <port> -u <username> -p <password>
  [-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

- Windows

```
> nebula-console.exe -addr <ip> -port <port> -u <username> -p <password>
  [-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

参数说明如下。

参数	说明
`-h/-help`	显示帮助菜单。
`-addr/-address`	设置要连接的 Graph 服务的 IP 地址。默认地址为 127.0.0.1。
`-P/-port`	设置要连接的 Graph 服务的端口。默认端口为 9669。
`-u/-user`	设置悦数图数据库账号的用户名。未启用身份认证时，可以使用任意已存在的用户名（默认为 `root`）。
`-p/-password`	设置用户名对应的密码。未启用身份认证时，密码可以填写任意字符。
`-t/-timeout`	设置整数类型的连接超时时间。单位为毫秒，默认值为 120。
`-e/-eval`	设置字符串类型的 nGQL 语句。连接成功后会执行一次该语句并返回结果，然后自动断开连接。
`-f/-file`	设置存储 nGQL 语句的文件的路径。连接成功后会执行该文件内的 nGQL 语句并返回结果，执行完毕后自动断开连接。
`-enable_ssl`	连接悦数图数据库时使用 SSL 加密。
`-ssl_root_ca_path`	指定 CA 证书的存储路径。
`-ssl_cert_path`	指定 CRT 证书的存储路径。
`-ssl_private_key_path`	指定私钥文件的存储路径。

更多参数参见项目仓库。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.7 管理 Storage 主机

从 3.0.0 版本开始，在配置文件中添加的 Storage 主机无法直接读写，配置文件的作用仅仅是将 Storage 主机注册至 Meta 服务中。必须使用 `ADD HOSTS` 命令后，才能正常读写 Storage 主机。

6.7.1 前提条件

- 已经连接服务

6.7.2 增加 Storage 主机

向集群中增加 Storage 主机。

```
nebula> ADD HOSTS <ip>:<port> [,<ip>:<port> ...];
nebula> ADD HOSTS "<hostname>":<port> [,"<hostname>":<port> ...];
```

Note

- 增加 Storage 主机在下一个心跳周期之后才能生效，为确保数据同步，请等待 2 个心跳周期（20 秒），然后执行 `SHOW HOSTS` 查看是否在线。
- IP地址和端口请和配置文件中的设置保持一致，例如单机部署的默认为 `127.0.0.1:9779` 。
- 使用域名时，需要用引号包裹，例如 `ADD HOSTS "foo-bar":9779` 。
- 确保新增的 Storage 主机没有被其他集群使用过，否则会导致添加 Storage 节点失败。

6.7.3 删除 Storage 主机

从集群中删除 Storage 主机。

Note

无法直接删除正在使用的 Storage 主机，需要先删除关联的图空间，才能删除 Storage 主机。

```
nebula> DROP HOSTS <ip>:<port> [,<ip>:<port> ...];
nebula> DROP HOSTS "<hostname>":<port> [,"<hostname>":<port> ...];
```

6.7.4 查看 Storage 主机

查看集群中的 Storage 主机。

```
nebula> SHOW HOSTS STORAGE;
+-------------+------+----------+-----------+--------------+----------------------+
| Host        | Port | Status   | Role      | Git Info Sha | Version              |
+-------------+------+----------+-----------+--------------+----------------------+
| "storaged0" | 9779 | "ONLINE" | "STORAGE" | "3ba41bd"    | "3.5.1" |
| "storaged1" | 9779 | "ONLINE" | "STORAGE" | "3ba41bd"    | "3.5.1" |
| "storaged2" | 9779 | "ONLINE" | "STORAGE" | "3ba41bd"    | "3.5.1" |
+-------------+------+----------+-----------+--------------+--------------- ------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.8 升级悦数图数据库 3.x 至 3.5.1

本文以悦数图数据库 3.1.0 为例，介绍如何升级 v3.x 至 v3.5.1。

6.8.1 升级说明

- 此升级操作仅适用于升级悦数图数据库 3.x（x < 4）至 3.5.1。对于升级 3.4.0 及以上版本至 3.5.1，可以直接替换二进制文件进行升级。具体操作请参见升级悦数图数据库至 3.5.1。

  Note

  如果您的悦数图数据库版本低于 3.0.0，请先升级到 3.1.0 再升级到 3.5.1。具体操作请参见升级悦数图数据库 2.x 至 3.1.0。

- 执行升级操作的集群 IP 地址必须与原集群相同。
- 机器硬盘剩余空间至少需为原数据目录的 1.5 倍。
- 在升级部署了全文索引的悦数图数据库前，需要手动删除 Elasticsearch (ES) 中的全文索引。在升级后需要重新使用 `SIGN IN` 语句登录 ES 并重新创建全文索引。

  Note

  用户可通过 cURL 命令手动删除 ES 中全文索引。命令为 `curl -XDELETE -u <es_username>:<es_password> '<es_access_ip>:<port>/<fullindex_name>'`，例如 `curl -XDELETE -u elastic:elastic 'http://192.168.8.223:9200/nebula_index_2534'`。如果 ES 没有设置用户名及密码，则无需指定 `-u` 选项。

6.8.2 升级步骤

1. 联系我们获取悦数图数据库 v3.5.1 的安装包并安装。

   Note

   不同安装包的升级步骤相同。本文以 RPM 包且安装目录为 `/usr/local/nebulagraph-ent-3.5.1` 为例。具体操作请参见安装 RPM 包。

   Caution

   请确保 3.5.1 集群的 Meta 服务和 Storage 服务的配置文件中的 `--data_path` 参数设置的存储路径数量与 3.x 集群的配置文件中的 `--data_path` 参数配置的路径数量相同。否则，升级后的集群无法启动。

2. 停止悦数图数据库 v3.x 服务。详情请参见管理悦数图数据库服务。 运行命令后可继续运行 `nebula.service status all` 命令以确认所有服务都已停止。

Copyright © Yueshu Graph Technology - All rights reserved.

3. 在悦数图数据库 v3.5.1 的安装目录下，分别执行以下命令以升级 Storage 和 Meta 服务。

- 升级 Storage 服务：

  命令：

```
sudo ./bin/db_upgrader  --max_concurrent_parts=<num> --src_db_path=<source_storage_data_path> --dst_db_path=<destination_storage_data_path>
```

参数	说明
`--max_concurrent_parts`	指定同时升级的分片数量，默认值为 `1` 。 建议根据磁盘性能适当调大。
`--src_db_path`	指定源数据目录的绝对路径。下述示例源数据的目录为 `/usr/local/nebula-ent-3.1.0/data/storage` 。
`--dst_db_path`	指定目标数据目录的绝对路径。本文示例的目标数据目录为 `/usr/local/nebula-ent-3.5.1/data/storage` 。

  示例：

```
sudo ./bin/db_upgrader --max_concurrent_parts=20 --src_db_path=/usr/local/nebula-ent-3.1.0/data/storage --dst_db_path=/usr/local/nebula-ent-3.5.1/data/storage
```

  如果有多个源数据目录，请分别指定不同的源数据目录和目标数据目录并执行命令。例如，有两个源数据目录 `/usr/local/nebula-ent-3.1.0/data/storage` 和 `/usr/local/nebula-ent-3.1.0/data2/storage` ，则执行以下命令：

```
sudo ./bin/db_upgrader --src_db_path=/usr/local/nebula-ent-3.1.0/data/storage --dst_db_path=/usr/local/nebula-ent-3.5.1/data/storage

sudo ./bin/db_upgrader --src_db_path=/usr/local/nebula-ent-3.1.0/data2/storage --dst_db_path=/usr/local/nebula-ent-3.5.1/data2/storage
```

- 升级 Meta 服务：

  命令：

```
sudo ./bin/meta_upgrader --src_meta_path=<source_meta_data_path> --dst_meta_path=<destination_meta_data_path>
```

参数	说明
`--src_meta_path`	指定源 Meta 数据目录的绝对路径。下述示例源数据的目录为 `/usr/local/nebula-ent-3.1.0/data/meta` 。
`--dst_meta_path`	指定目标 Meta 数据目录的绝对路径。本文示例的目标数据目录为 `/usr/local/nebula-ent-3.5.1/data/meta` 。

  示例：

```
sudo ./bin/meta_upgrader --src_meta_path=/usr/local/nebula-ent-3.1.0/data/meta --dst_meta_path=/usr/local/nebula-ent-3.5.1/data/meta
```

  如果有多个源 Meta 数据目录，请指定不同的源 Meta 数据目录和目标 Meta 数据目录并分别执行命令。

  服务升级完成后，会在 v3.5.1 的安装目录下生成 `data` 目录，其中包含升级后的数据文件。

4. 启动和连接悦数图数据库 v3.5.1 服务后，验证数据是否正确。参考命令如下：

```
nebula> SHOW HOSTS;
nebula> SHOW HOSTS storage;
nebula> SHOW SPACES;
nebula> USE <space_name>
nebula> SHOW PARTS;
nebula> SUBMIT JOB STATS;
nebula> SHOW STATS;
nebula> MATCH (v) RETURN v LIMIT 5;
```

6.8.3 Docker Compose 部署

Caution

Docker Compose 部署的悦数图数据库建议重新部署新版本后导入数据。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

6.9 卸载悦数图数据库

本文介绍如何卸载悦数图数据库。

Caution

如果需要重新部署悦数图数据库，请务必完全卸载后再重新部署，否则可能会出现问题，包括 Meta 不一致等。

6.9.1 前提条件

停止悦数图数据库服务。详情参见管理悦数图数据库服务。

6.9.2 步骤 1：删除数据和元数据文件

如果在配置文件内修改了数据文件的路径，可能会导致安装路径和数据文件保存路径不一致，因此需要查看配置文件，确认数据文件保存路径，然后手动删除数据文件目录。

Note

如果是集群架构，需要删除所有 Storage 和 Meta 服务节点的数据文件。

1. 检查 Storage 服务的 disk 配置。例如：

```
########## Disk ##########
# Root data path. Split by comma. e.g. --data_path=/disk1/path1/,/disk2/path2/
# One path per Rocksdb instance.
--data_path=/nebula/data/storage
```

2. 检查 metad 服务的配置文件，找到对应元数据目录。
3. 删除以上数据和元数据目录。

6.9.3 步骤 2：卸载安装目录

Note

删除整个安装目录，包括 `cluster.id` 文件。

安装路径为参数 `--prefix` 指定的路径。默认路径为 `/usr/local/nebula` 。

卸载编译安装的悦数图数据库

找到悦数图数据库的安装目录，删除整个安装目录。

卸载 **RPM** 包安装的悦数图数据库

1. 使用如下命令查看悦数图数据库版本。

```
$ rpm -qa | grep "nebula"
```

返回类似如下结果。

```
nebula-graph-3.5.1-1.x86_64
```


Copyright © Yueshu Graph Technology - All rights reserved.

2. 使用如下命令卸载悦数图数据库。

```
sudo rpm -e <nebula_version>
```

例如：

```
sudo rpm -e nebula-graph-3.5.1-1.x86_64
```

3. 删除安装目录。

卸载 **DEB** 包安装的悦数图数据库

1. 使用如下命令查看悦数图数据库版本。

```
$ dpkg -l | grep "nebula"
```

返回类似如下结果。

```
ii  nebula-graph  3.5.1  amd64    NebulaGraph Package built using CMake
```

2. 使用如下命令卸载悦数图数据库。

```
sudo dpkg -r <nebula_version>
```

例如：

```
sudo dpkg -r nebula-graph
```

3. 删除安装目录。

卸载 **Docker Compose** 部署的悦数图数据库

1. 在目录 `nebula-docker-compose` 内执行如下命令停止悦数图数据库服务。

```
docker-compose down -v
```

2. 删除目录 `nebula-docker-compose` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7. 配置与日志

7.1 配置

7.1.1 配置管理

悦数图数据库基于 gflags 库打造了系统配置，多数配置项都是其中的 flags。悦数图数据库服务启动时，默认会从配置文件中获取配置信息。对于文件中没有的配置项，系统使用默认值。

Enterpriseonly

性能、参数、查询语句的调优方式及服务仅在企业版提供。

Note

- 由于配置项多且可能随着悦数图数据库的开发发生变化，文档不会介绍所有配置项。按下文说明可在命令行获取配置项的详细说明。
- 不建议修改文档未介绍的配置项，除非已经熟悉源代码并完全了解配置项的作用。

版本兼容性

1.x 版本的文档提供了使用 CONFIGS 命令修改缓存中配置的方法，但在生产环境中使用该方法容易导致集群配置与本地配置文件不一致。因此，自 2.x 版本开始文档中将不再介绍 CONFIGS 命令的使用方法。

查看配置项列表与说明

使用以下命令获取二进制文件对应服务的所有配置项信息：

```
<binary> --help
```

例如：

```
# 获取 Meta 配置项的帮助信息
$ /usr/local/nebula/bin/nebula-metad  --help

# 获取 Graph 配置项的帮助信息
$ /usr/local/nebula/bin/nebula-graphd --help

# 获取 Storage 配置项的帮助信息
$ /usr/local/nebula/bin/nebula-storaged --help
```

以上示例使用了二进制文件的默认存储路径 `/usr/local/nebula/bin/` 。如果修改了悦数图数据库安装路径，使用实际路径查询配置项信息。

查看运行配置

使用 `curl` 命令获取运行中的配置项的值，即悦数图数据库的运行配置。

例如：

```
# 获取 Meta 服务的运行配置
curl 127.0.0.1:19559/flags

# 获取 Graph 服务的运行配置
curl 127.0.0.1:19669/flags

# 获取 Storage 服务的运行配置
curl 127.0.0.1:19779/flags
```


Copyright © Yueshu Graph Technology - All rights reserved.

Note

实际环境中需使用真实的主机 IP 地址取代以上示例中的 `127.0.0.1`。

配置文件简介

源码、RPM/DEB、TAR 包集群的配置文件

悦数图数据库为每个服务都提供了两份初始配置文件 `<service_name>.conf.default` 和 `<service_name>.conf.production`，方便用户在不同场景中使用。使用源码和 RPM/DEB 安装集群的配置文件的默认路径为 `/usr/local/nebula/etc/`；使用 TAR 包安装集群的配置文件路径为 `<install_path>/<tar_package_directory>/etc` TAR 包的安装路径。

初始配置文件中的配置值仅供参考，使用时可根据实际需求调整。如需使用初始配置文件，从上述两个文件选择其一，删除后缀 `.default` 或 `.production` 使其生效。

Note

为确保服务的可用性，同类服务的配置建议保持一致，本机 IP 地址 `local_ip` 除外。例如，一个悦数图数据库集群中部署了 3 个 Storage 服务器，3 者除 IP 地址外的其它配置建议都相同。

下表列出了各服务对应的初始配置文件。

悦数图数据库服务	初始配置文件	配置说明
Meta	`nebula-metad.conf.default` 和 `nebula-metad.conf.production`	Meta 服务配置
Graph	`nebula-graphd.conf.default` 和 `nebula-graphd.conf.production`	Graph 服务配置
Storage	`nebula-storaged.conf.default` 和 `nebula-storaged.conf.production`	Storage 服务配置

所有服务的初始配置文件中都包含 `local_config` 参数，预设值为 `true`，表示悦数图数据库服务会从其配置文件获取配置并启动。

Caution

不建议修改 `local_config` 的值为 `false`。修改后悦数图数据库服务启动后会先尝试从 Meta 服务获取缓存的配置信息，可能导致集群配置不一致，造成未知风险。

DOCKER COMPOSE 集群的配置文件

对于使用 Docker Compose 创建的集群，集群的配置文件的默认路径为 `<install_path>/nebula-docker-compose/docker-compose.yaml`。配置文件中的 `command` 下面的参数为各服务的启动参数。

NEBULAGRAPH OPERATOR 集群的配置文件

对于通过 NebulaGraph Operator 使用 Kubectl 方式创建的集群，集群的配置文件的路径为用户创建集群 YAML 文件的路径。用户可通过配置文件中的 `spec.{graphd|storaged|metad}.config` 参数修改集群各个服务的相关配置。

Note

通过 Helm 部署的集群，暂不支持修改集群服务的相关配置。

关于悦数图数据库的配置相关的更多信息，参见 悦数图数据库 Config。

修改配置

用户可以在配置文件中修改悦数图数据库的配置，或使用命令动态修改配置。

Copyright © Yueshu Graph Technology - All rights reserved.

Caution

同时使用两种方式修改配置会导致配置信息失去统一的管理方式，可能造成配置混乱。建议仅使用配置文件管理配置，或在通过命令动态更新配置后对配置文件做相同的修改，以保证一致性。

在配置文件中修改配置

默认情况下，所有悦数图数据库服务从配置文件获取配置。用户可以按照以下步骤修改配置并使其生效。

- 针对使用源码、RPM/DEB、TAR 包安装的集群：

a. 使用文本编辑器修改目标服务的配置文件并保存。

b. 选择合适的时间重启所有悦数图数据库服务使修改生效。

- 针对使用 Docker Compose 安装的集群：

a. 在文件 `<install_path>/nebula-docker-compose/docker-compose.yaml` 中，修改服务配置。

b. 在目录 `nebula-docker-compose` 内执行命令 `docker-compose up -d` 重启涉及配置变化的服务。

- 针对使用 Kubectl 方式创建的集群：

  具体操作，参见自定义集群的配置参数。

使用命令动态修改配置

用户可以通过 `curl` 命令动态修改悦数图数据库服务的配置。例如，修改 Storage 服务的 `wal_ttl` 参数为 `600`，命令如下：

```
curl -X PUT -H "Content-Type: application/json" -d'{"wal_ttl":"600"}' -s "http://192.168.15.6:19779/flags"
```

其中，`{"wal_ttl":"600"}` 为待修改的配置参数及其值；`192.168.15.6:19779` 为 Storage 服务的 IP 地址和 HTTP 端口号。

Caution

- 动态修改配置功能仅适用于原型验证和测试环境，不建议在生产环境中使用。因为当 `local_config` 值设置为 `true` 时，动态修改的配置不会持久化，重启服务后配置会恢复为初始配置。
- 仅支持动态修改部分配置参数，具体支持的参数列表，参见各服务配置中是否支持运行时动态修改的描述。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7.1.2 Meta 服务配置

Meta 服务提供了两份初始配置文件 `nebula-metad.conf.default` 和 `nebula-metad.conf.production`，方便在不同场景中使用。文件的默认路径为 `/usr/local/nebula/etc/`。

Caution

- 不建议修改 `local_config` 的值为 `false`。修改后悦数图数库服务启动后会先尝试从 Meta 服务获取缓存的配置信息，可能导致集群配置不一致，造成未知风险。
- 不建议修改文档未介绍的配置项，除非已经熟悉源代码并清楚了解配置项作用。

配置文件使用方式

如需使用初始配置文件，从上述两个文件选择其一，删除后缀 `.default` 或 `.production`，Meta 服务才能将其识别为配置文件并从中获取配置信息。

配置文件参数值说明

配置文件内没有设置某个参数表示参数使用的是默认值。文件内只预设了部分参数的值，而且两份初始配置文件内的参数值也略有不同，本文的预设值以 `nebula-metad.conf.default` 为准。

Caution

配置文件中的部分参数值可以在运行时动态修改。本文将这些参数标记为支持运行时动态修改，并当 `local_config` 值设置为 `true` 时重启服务后配置会恢复为初始配置。详情参见修改配置。

如需查看所有的参数及其当前值，参见配置管理。

basics 配置

名称	预设值	说明	是否支持运行时动态修改
`daemonize`	`true`	是否启动守护进程。	不支持
`pid_file`	`pids/nebula-metad.pid`	记录进程 ID 的文件。	不支持
`timezone_name`	-	指定悦数图数库的时区。初始配置文件中未设置该参数，如需使用请手动添加。系统默认值为 `UTC+00:00:00`。格式请参见 Specifying the Time Zone with TZ。例如，东八区的设置方式为 `--timezone_name=UTC+08:00`。	不支持

Note

- 在插入时间类型的属性值时，悦数图数库会根据 `timezone_name` 设置的时区将该时间值（TIMESTAMP 类型例外）转换成相应的 UTC 时间，因此在查询中返回的时间类型属性值为 UTC 时间。
- `timezone_name` 参数只用于转换悦数图数库中存储的数据，悦数图数库进程中其它时区相关数据，例如日志打印的时间等，仍然使用主机系统默认的时区。

Copyright © Yueshu Graph Technology - All rights reserved.

enterprise license 配置

名称	预设值	说明	是否支持运行时动态修改
`license_manager_url`	-	License Manager 地址。设置为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。更多信息，参见 License Manager。	不支持

logging 配置

名称	预设值	说明	是否支持运行时动态修改
`log_dir`	`logs`	存放 Meta 服务日志的目录，建议和数据保存在不同硬盘。	不支持
`minloglevel`	`0`	最小日志级别，即记录此级别或更高级别的日志。可选值为 `0`（INFO）、`1`（WARNING）、`2`（ERROR）、`3`（FATAL）。建议在调试时设置为 `0`，生产环境中设置为 `1`。如果设置为 `4`，悦数图数据库不会记录任何日志。	支持
`v`	`0`	VLOG 日志详细级别，即记录小于或等于此级别的所有 VLOG 消息。可选值为 `0`、`1`、`2`、`3`、`4`、`5`。glog 提供的 VLOG 宏允许用户定义自己的数字日志记录级别，并用参数 `v` 控制记录哪些详细消息。详情参见 Verbose Logging。	支持
`logbufsecs`	`0`	缓冲日志的最大时间，超时后输出到日志文件。`0` 表示实时输出。单位：秒。	不支持
`redirect_stdout`	`true`	是否将标准输出和标准错误重定向到单独的输出文件。	不支持
`stdout_log_file`	`metad-stdout.log`	标准输出日志文件名称。	不支持
`stderr_log_file`	`metad-stderr.log`	标准错误日志文件名称。	不支持
`stderrthreshold`	`3`	要复制到标准错误中的最小日志级别（`minloglevel`）。	不支持
`timestamp_in_logfile_name`	`true`	日志文件名称中是否包含时间戳。`true` 表示包含，`false` 表示不包含。	不支持

Copyright © Yueshu Graph Technology - All rights reserved.

networking 配置

名称	预设值	说明	是否支持运行时动态修改
`meta_server_addrs`	`127.0.0.1:9559`	全部 Meta 服务的 IP 地址和端口。多个 Meta 服务用英文逗号（`,`）分隔。	不支持
`local_ip`	`127.0.0.1`	Meta 服务的本地 IP 地址。本地 IP 地址用于识别 nebula-metad 进程，如果是分布式集群或需要远程访问，请修改为对应地址。	不支持
`port`	`9559`	Meta 服务的 RPC 守护进程监听端口。Meta 服务对外端口为 `9559`，对内端口为 `对外端口+1`，即 `9560`，悦数图数据库使用内部端口进行多副本间的交互。	不支持
`ws_ip`	`0.0.0.0`	HTTP 服务的 IP 地址。	不支持
`ws_http_port`	`19559`	HTTP 服务的端口。	不支持
`ws_storage_http_port`	`19779`	HTTP 协议监听 Storage 服务的端口，需要和 Storage 服务配置文件中的 `ws_http_port` 保持一致。仅存算合并版悦数图数据库需要设置本参数。	不支持
`heartbeat_interval_secs`	`10`	默认心跳间隔。请确保所有服务的 `heartbeat_interval_secs` 取值相同，否则会导致系统无法正常工作。单位：秒。	支持

Caution

必须在配置文件中使用真实的 IP 地址。否则某些情况下 `127.0.0.1/0.0.0.0` 无法正确解析。

storage 配置

名称	预设值	说明	是否支持运行时动态修改
`data_path`	`data/meta`	meta 数据存储路径。	不支持

misc 配置

名称	预设值	说明	是否支持运行时动态修改
`default_parts_num`	`100`	创建图空间时的默认分片数量。	不支持
`default_replica_factor`	`1`	创建图空间时的默认副本数量。	不支持

rocksdb options 配置

名称	预设值	说明	是否支持运行时动态修改
`rocksdb_wal_sync`	`true`	是否同步 RocksDB 的 WAL 日志。	不支持

Copyright © Yueshu Graph Technology - All rights reserved.

Black box 配置

Enterpriseonly

仅企业版支持黑匣子功能配置。

名称	预设值	说明	是否支持运行时动态修改
`ng_black_box_switch`	`true`	是否开启黑匣子功能。	不支持
`ng_black_box_home`	`black_box`	存储黑匣子数据的目录名称。	不支持
`ng_black_box_dump_period_seconds`	`5`	黑匣子采集指标的时间间隔。单位：秒。	不支持
`ng_black_box_file_lifetime_seconds`	`1800`	采集指标后生成文件的保存时间。单位：秒	支持

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7.1.3 Graph 服务配置

Graph 服务提供了两份初始配置文件 `nebula-graphd.conf.default` 和 `nebula-graphd.conf.production`，方便在不同场景中使用。文件的默认路径为 `/usr/local/nebula/etc/` 。

Caution

- 不建议修改 `local_config` 的值为 `false` 。修改后悦数图数库服务启动后会先尝试从 Meta 服务获取缓存的配置信息，可能导致集群配置不一致，造成未知风险。
- 不建议修改文档未介绍的配置项，除非已经熟悉源代码并完全了解配置项的作用。

配置文件使用方式

如需使用初始配置文件，从上述两个文件选择其一，删除后缀 `.default` 或 `.production` ，Graph 服务才能将其识别为配置文件并从中获取配置信息。

配置文件参数值说明

配置文件内没有设置某个参数表示参数使用的是默认值。文件内只预设了部分参数的值，而且两份初始配置文件内的参数值也略有不同，本文的预设值以 `nebula-graphd.conf.default` 为准。

Caution

配置文件中的部分参数值可以在运行时动态修改。本文将这些参数标记为支持运行时动态修改，并当 `local_config` 值设置为 `true` 时重启服务后配置会恢复为初始配置。详情参见修改配置。

如需查看所有的参数及其当前值，参见配置管理。

basics 配置

名称	预设值	说明	是否支持运行时动态修改
`daemonize`	`true`	是否启动守护进程。	不支持
`pid_file`	`pids/nebula-graphd.pid`	记录进程 ID 的文件。	不支持
`enable_optimizer`	`true`	是否启用优化器。	不支持
`timezone_name`	-	指定悦数图数库的时区。初始配置文件中未设置该参数，使用需手动添加。系统默认值为 `UTC+00:00:00` 。格式请参见 Specifying the Time Zone with TZ。例如，东八区的设置方式为 `--timezone_name=UTC+08:00` 。	不支持
`local_config`	`true`	是否从配置文件获取配置信息。	不支持

Note

- 在插入时间类型 的属性值时，悦数图数库会根据 `timezone_name` 设置的时区将该时间值（TIMESTAMP 类型例外）转换成相应的 UTC 时间，因此在查询中返回的时间类型属性值为 UTC 时间。
- `timezone_name` 参数只用于转换悦数图数库中存储的数据，悦数图数库进程中其它时区相关数据，例如日志打印的时间等，仍然使用主机系统默认的时区。

Copyright © Yueshu Graph Technology - All rights reserved.

logging 配置

名称	预设值	说明	是否支持运行时动态修改
`log_dir`	`logs`	存放 Graph 服务日志的目录，建议和数据保存在不同硬盘。	不支持
`minloglevel`	`0`	最小日志级别，即记录此级别或更高级别的日志。可选值为 `0`（INFO）、`1`（WARNING）、`2`（ERROR）、`3`（FATAL）。建议在调试时设置为 `0`，生产环境中设置为 `1`。如果设置为 `4`，悦数图数据库不会记录任何日志。	支持
`v`	`0`	VLOG 日志详细级别，即记录小于或等于此级别的所有 VLOG 消息。可选值为 `0`、`1`、`2`、`3`、`4`、`5`。glog 提供的 VLOG 宏允许用户定义自己的数字日志记录级别，并用参数 `v` 控制记录哪些详细消息。详情参见 Verbose Logging。	支持
`logbufsecs`	`0`	缓冲日志的最大时间，超时后输出到日志文件。`0` 表示实时输出。单位：秒。	不支持
`redirect_stdout`	`true`	是否将标准输出和标准错误重定向到单独的输出文件。	不支持
`stdout_log_file`	`graphd-stdout.log`	标准输出日志文件名称。	不支持
`stderr_log_file`	`graphd-stderr.log`	标准错误日志文件名称。	不支持
`stderrthreshold`	`3`	要复制到标准错误中的最小日志级别（`minloglevel`）。	不支持
`timestamp_in_logfile_name`	`true`	日志文件名称中是否包含时间戳。`true` 表示包含，`false` 表示不包含。	不支持

query 配置

名称	预设值	说明	是否支持运行时动态修改
`accept_partial_success`	`false`	是否将部分成功视为错误。此配置仅适用于只读请求，写请求总是将部分成功视为错误。	支持
`session_reclaim_interval_secs`	`60`	将 Session 信息发送给 Meta 服务的间隔。单位：秒。	支持
`max_allowed_query_size`	`4194304`	最大查询语句长度。单位：字节。默认为 `4194304`，即 4MB。	支持

Copyright © Yueshu Graph Technology - All rights reserved.

networking 配置

名称	预设值	说明	是否支持运行时动态修改
`meta_server_addrs`	`127.0.0.1:9559`	全部 Meta 服务的 IP 地址和端口。多个 Meta 服务用英文逗号（,）分隔。	不支持
`local_ip`	`127.0.0.1`	Graph 服务的本地 IP 地址。本地 IP 地址用于识别 nebula-graphd 进程，如果是分布式集群或需要远程访问，请修改为对应地址。	不支持
`listen_netdev`	`any`	监听的网络设备。	不支持
`port`	`9669`	Graph 服务的 RPC 守护进程监听端口。	不支持
`reuse_port`	`false`	是否启用 SO_REUSEPORT。	不支持
`listen_backlog`	`1024`	socket 监听的连接队列最大长度，调整本参数需要同时调整 `net.core.somaxconn`。	不支持
`client_idle_timeout_secs`	`28800`	空闲连接的超时时间。取值范围为 1~604800，单位：秒。默认 8 小时。	不支持
`session_idle_timeout_secs`	`28800`	空闲会话的超时时间。取值范围为 1~604800。默认 8 小时。单位：秒。	不支持
`num_accept_threads`	`1`	接受传入连接的线程数。	不支持
`num_netio_threads`	`0`	网络 IO 线程数。`0` 表示 CPU 核数。	不支持
`num_max_connections`	`0`	所有网络线程的最大活动连接数，`0` 表示没有限制。 每个网络线程的最大连接数= `num_max_connections` / `num_netio_threads`。	不支持
`num_worker_threads`	`0`	执行用户查询的线程数。`0` 表示 CPU 核数。	不支持
`ws_ip`	`0.0.0.0`	HTTP 服务的 IP 地址。	不支持
`ws_http_port`	`19669`	HTTP 服务的端口。	不支持
`heartbeat_interval_secs`	`10`	默认心跳间隔。请确保所有服务的 `heartbeat_interval_secs` 取值相同，否则会导致系统无法正常工作。单位：秒。	支持
`storage_client_timeout_ms`	-	Graph 服务与 Storage 服务的 RPC 连接超时时间。初始配置文件中未设置该参数，使用需手动添加。默认值为 `60000` 毫秒。	不支持
`enable_record_slow_query`	`true`	是否启用慢查询记录。注：企业版功能。	不支持
`slow_query_limit`	`100`	慢查询记录的最大条数。注：企业版功能。	不支持
`slow_query_threshold_us`	`200000`	定义超过多长时间的查询为慢查询。单位：微秒。	不支持
`ws_meta_http_port`	`19559`	HTTP 协议监听 Meta 服务的端口，需要和 Meta 服务配置文件中的 `ws_http_port` 保持一致。	不支持

Caution

必须在配置文件中使用真实的 IP 地址。否则某些情况下 `127.0.0.1/0.0.0.0` 无法正确解析。

Copyright © Yueshu Graph Technology - All rights reserved.

charset and collate 配置

名称	预设值	说明	是否支持运行时动态修改
`default_charset`	`utf8`	创建图空间时的默认字符集。	不支持
`default_collate`	`utf8_bin`	创建图空间时的默认排序规则。	不支持

authorization 配置

名称	预设值	说明	是否支持运行时动态修改
`enable_authorize`	`false`	用户登录时是否进行身份验证。身份验证详情请参见身份验证。	不支持
`auth_type`	`password`	用户登录的身份验证方式。取值为 `password` 、 `ldap` 、 `cloud` 。	不支持

memory 配置

名称	预设值	说明	是否支持运行时动态修改
`system_memory_high_watermark_ratio`	`0.8`	内存高水位报警机制的触发阈值。系统内存占用率高于该值会触发报警机制，悦数图数据库会停止接受查询。	支持

audit 配置

详细参数说明参见审计日志。

metrics 配置

名称	预设值	说明	是否支持运行时动态修改
`enable_space_level_metrics`	`false`	开启后可打开图空间级别的监控，对应的监控指标名称中包含图空间的名称，例如 `query_latency_us{space=basketballplayer}.avg.3600` 。支持的监控指标可用 `curl` 命令查看，详细说明参见查询监控指标。	不支持

session 配置

名称	预设值	说明	是否支持运行时动态修改
`max_sessions_per_ip_per_user`	`300`	相同用户和 IP 地址可以创建的最大活跃会话数。	不支持

Copyright © Yueshu Graph Technology - All rights reserved.

Black box 配置

Enterpriseonly

仅企业版支持黑匣子功能配置。

名称	预设值	说明	是否支持运行时动态修改
`ng_black_box_switch`	`true`	是否开启黑匣子功能。	不支持
`ng_black_box_home`	`black_box`	存储黑匣子数据的目录名称。	不支持
`ng_black_box_dump_period_seconds`	`5`	黑匣子采集指标的时间间隔。单位：秒。	不支持
`ng_black_box_file_lifetime_seconds`	`1800`	采集指标后生成文件的保存时间。单位：秒	支持

memory tracker 配置

名称	预设值	说明	是否支持运行时动态修改
`memory_tracker_limit_ratio`	`0.8`	可用内存的百分比，当可用内存低于该值时，悦数图数据库会停止接受查询。 计算公式：可用内存/(总内存 - 保留内存)。 注意：对于混合部署的集群，需要根据实际情况调小该参数。例如，当预期 Graphd 只占用 50% 的内存时，该参数的值可设置为小于 `0.5` 。	支持
`memory_tracker_untracked_reserved_memory_mb`	`50`	保留内存的大小，单位：MB。	支持
`memory_tracker_detail_log`	`false`	是否定期生成较详细的内存跟踪日志。当值为 `true` 时，会定期生成内存跟踪日志。	支持
`memory_tracker_detail_log_interval_ms`	`60000`	内存跟踪日志的生成时间间隔，单位：毫秒。仅当 `memory_tracker_detail_log` 为 `true` 时，该参数生效。	支持
`memory_purge_enabled`	`true`	是否定期开启内存清理功能。当值为 `true` 时，会定期清理内存。	支持
`memory_purge_interval_seconds`	`10`	内存清理的时间间隔，单位：秒。 `memory_purge_enabled` 为 `true` 时，该参数生效。	支持

performance optimization 配置

名称	预设值	说明	是否支持运行时动态修改
`max_job_size`	`1`	最大作业并发数，即查询执行时在可以并发执行的阶段所采用的最大线程数。建议为物理 CPU 核数的一半。	支持
`min_batch_size`	`8192`	处理数据集的最小批处理大小。仅在 `max_job_size` 大于 1 时生效。	支持
`optimize_appendvertices`	`false`	启用后，执行 `MATCH` 语句时不过滤悬挂边。	支持
`path_batch_size`	`10000`	每个线程构建的路径数。	支持

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7.1.4 Storage 服务配置

Storage 服务提供了两份初始配置文件 `nebula-storaged.conf.default` 和 `nebula-storaged.conf.production`，方便在不同场景中使用。文件的默认路径为 `/usr/local/nebula/etc/`。

Caution

- 不建议修改 `local_config` 的值为 `false`。修改配置并重启 Storage 服务，会先尝试从 Meta 服务获取缓存的配置信息，可能导致集群配置不一致，造成未知风险。
- 不建议修改文档未介绍的配置项，除非已经熟悉源代码并完全了解配置项的作用。

配置文件使用方式

如需使用初始配置文件，从上述两个文件选择其一，删除后缀 `.default` 或 `.production`，Storage 服务才能将其识别为配置文件并从中获取配置信息。

配置文件参数值说明

配置文件内没有设置某个参数表示参数使用的是默认值。文件内只预设了部分参数的值，而且两份初始配置文件内的参数值也略有不同，本文的预设值以 `nebula-storaged.conf.default` 文件为准，其中没有的参数则以 `nebula-storaged.conf.production` 文件为准。

Caution

配置文件中的部分参数值可以在运行时动态修改。本文将这些参数标记为支持运行时动态修改，并当 `local_config` 值设置为 `true` 时重启服务后配置会恢复为初始配置。详情参见修改配置。

Note

Raft Listener 的配置和 Storage 服务配置不同，详情请参见部署 Raft listener。

如需查看所有的参数及其当前值，参见配置管理。

basics 配置

名称	预设值	说明	是否支持运行时动态修改
`daemonize`	`true`	是否启动守护进程。	不支持
`pid_file`	`pids/nebula-storaged.pid`	记录进程 ID 的文件。	不支持
`timezone_name`	`UTC+00:00:00`	指定悦数图数据库的时区。初始配置文件中未设置该参数，如需使用请手动添加。格式请参见 Specifying the Time Zone with TZ。例如，东八区的设置方式为 `--timezone_name=UTC+08:00`。	不支持
`local_config`	`true`	是否从配置文件获取配置信息。	不支持

Copyright © Yueshu Graph Technology - All rights reserved.

Note

- 在插入时间类型的属性值时，悦数图数据库会根据 `timezone_name` 设置的时区将该时间值（TIMESTAMP 类型例外）转换成相应的 UTC 时间，因此在查询中返回的时间类型属性值为 UTC 时间。
- `timezone_name` 参数只用于转换悦数图数据库中存储的数据，悦数图数据库进程中其它时区相关数据，例如日志打印的时间等，仍然使用主机系统默认的时区。

logging 配置

名称	预设值	说明	是否支持运行时动态修改
`log_dir`	`logs`	存放 Storage 服务日志的目录，建议和数据保存在不同硬盘。	不支持
`minloglevel`	`0`	最小日志级别，即记录此级别或更高级别的日志。可选值：`0`（INFO）、`1`（WARNING）、`2`（ERROR）、`3`（FATAL）。建议在调试时设置为 `0`，生产环境中设置为 `1`。如果设置为 `4`，悦数图数据库不会记录任何日志。	支持
`v`	`0`	VLOG 日志详细级别，即记录小于或等于此级别的所有 VLOG 消息。可选值为 `0`、`1`、`2`、`3`、`4`、`5`。glog 提供的 VLOG 宏允许用户定义自己的数字日志记录级别，并用参数 `v` 控制记录哪些详细消息。详情参见 Verbose Logging。	支持
`logbufsecs`	`0`	缓冲日志的最大时间，超时后输出到日志文件。`0` 表示实时输出。单位：秒。	不支持
`redirect_stdout`	`true`	是否将标准输出和标准错误重定向到单独的输出文件。	不支持
`stdout_log_file`	`storaged-stdout.log`	标准输出日志文件名称。	不支持
`stderr_log_file`	`storaged-stderr.log`	标准错误日志文件名称。	不支持
`stderrthreshold`	`3`	要复制到标准错误中的最小日志级别（`minloglevel`）。	不支持
`timestamp_in_logfile_name`	`true`	日志文件名称中是否包含时间戳。`true` 表示包含，`false` 表示不包含。	不支持

Copyright © Yueshu Graph Technology - All rights reserved.

networking 配置

名称	预设值	说明	是否支持运行时动态修改
`meta_server_addrs`	`127.0.0.1:9559`	全部 Meta 服务的 IP 地址和端口。多个 Meta 服务用英文逗号（`,`）分隔。	不支持
`local_ip`	`127.0.0.1`	Storage 服务的本地 IP 地址。本地 IP 地址用于识别 nebula-storaged 进程，如果是分布式集群或需要远程访问，请修改为对应地址。	不支持
`port`	`9779`	Storage 服务的 RPC 守护进程监听端口。Storage 服务对外端口为 `9779`，对内端口为 `9777`、`9778` 和 `9780`，悦数图数据库使用内部端口进行多副本间的交互。 `9777`：drainer 服务占用端口，仅在企业版集群中暴露；`9778`：Admin 服务（Storage 接收 Meta 命令的服务）占用的端口；`9780`：Raft 通信端口。	不支持
`ws_ip`	`0.0.0.0`	HTTP 服务的 IP 地址。	不支持
`ws_http_port`	`19779`	HTTP 服务的端口。	不支持
`heartbeat_interval_secs`	`10`	默认心跳间隔。请确保所有服务的 `heartbeat_interval_secs` 取值相同，否则会导致系统无法正常工作。单位：秒。	支持

Caution

必须在配置文件中使用真实的 IP 地址。否则某些情况下 `127.0.0.1/0.0.0.0` 无法正确解析。

raft 配置

名称	预设值	说明	是否支持运行时动态修改
`raft_heartbeat_interval_secs`	`30`	Raft 选举超时时间。单位：秒。	支持
`raft_rpc_timeout_ms`	`500`	Raft 客户端的远程过程调用（RPC）超时时间。单位：毫秒。	支持
`wal_ttl`	`14400`	Raft WAL 的有效时间。单位：秒。	支持

Copyright © Yueshu Graph Technology - All rights reserved.

disk 配置

Copyright © Yueshu Graph Technology - All rights reserved.

名称	预设值	说明	是否支持运行时动态修改
`data_path`	`data/storage`	数据存储路径，多个路径用英文逗号（`,`）分隔。一个 RocksDB 实例对应一个分片。	不支持
`minimum_reserved_bytes`	`268435456`	每个数据存储路径的剩余空间最小值，低于该值时，可能会导致集群数据写入失败。单位：字节。	不支持
`rocksdb_batch_size`	`4096`	批量操作的缓存大小。单位：字节。	不支持
`rocksdb_block_cache`	`4`	BlockBasedTable 的默认块缓存大小。单位：兆字节。	不支持
`disable_page_cache`	`false`	允许或禁止悦数图数据库使用操作系统的页缓存。默认值为 `false`，表示允许使用 page cache。当值为 `true` 时，禁止悦数图数据库使用 page cache，此时须设置充足的块缓存（block cache）空间。	不支持
`engine_type`	`rocksdb`	存储引擎类型。	不支持
`rocksdb_compression`	`lz4`	压缩算法，可选值：`no`、`snappy`、`lz4`、`lz4hc`、`zlib`、`bzip2`、`zstd`。 该参数会修改每一层的压缩算法，如需为不同层级设置不同的压缩算法，请使用 `rocksdb_compression_per_level` 参数。	不支持
`rocksdb_compression_per_level`	-	为不同层级设置不同的压缩算法。优先级高于 `rocksdb_compression`。例如 `no:no:lz4:lz4:snappy:zstd:snappy`。 也可以不设置某个层级的压缩算法，例如 `no:no:lz4:lz4::zstd`，此时 L4、L6 层使用 `rocksdb_compression` 参数的压缩算法。	不支持
`enable_rocksdb_statistics`	`false`	是否启用 RocksDB 的数据统计。	不支持
`rocksdb_stats_level`	`kExceptHistogramOrTimers`	RocksDB 的数据统计级别。可选值：`kExceptHistogramOrTimers`（禁用计时器统计，跳过柱状图统计）、`kExceptTimers`（跳过计时器统计）、`kExceptDetailedTimers`（收集除互斥锁和压缩花费时间之外的所有统计数据）、`kExceptTimeForMutex` 收集除互斥锁花费时间之外的所有统计数据）、`kAll`（收集所有统计数据）。	不支持
`enable_rocksdb_prefix_filtering`	`true`	是否启用 prefix bloom filter，启用时可以提升图遍历速度，但是会增加内存消耗。	不支持

Copyright © Yueshu Graph Technology - All rights reserved.

名称	预设值	说明	是否支持运行时动态修改
`enable_rocksdb_whole_key_filtering`	`false`	是否启用 whole key bloom filter。	不支持
`rocksdb_filtering_prefix_length`	`12`	每个 key 的 prefix 长度。可选值：`12`（分片 ID+点 ID）、`16`（分片 ID+点 ID+TagID/Edge typeID）。单位：字节。	不支持
`enable_partitioned_index_filter`	`false`	设置为 `true` 可以降低 bloom 过滤器占用的内存大小，但是在某些随机寻道（random-seek）的情况下，可能会降低读取性能。初始配置文件中未设置该参数，如需使用请手动添加。	不支持

misc 配置

Caution

下表中的 `snapshot` 与悦数图数据库快照是不同的概念，这里的 `snapshot` 指 Raft 同步过程中 leader 上的存量数据。

名称	预设值	说明	是否支持运行时动态修改
`query_concurrently`	`true`	是否开启多线程查询。开启后可以提高单个查询的时延性能，但大压力下会降低整体的吞吐量。	支持
`auto_remove_invalid_space`	`true`	在执行 `DROP SPACE` 后，会删除指定图空间，该参数设置是否同时删除指定图空间内的所有数据。当值为 `true` 时，同时删除指定图空间内的所有数据。	支持
`num_io_threads`	`16`	网络 I/O 线程的数量，用于发送 RPC 请求和接收响应。	不支持
`num_max_connections`	`0`	所有网络线程的最大活动连接数，`0` 表示没有限制。 每个网络线程的最大连接数= `num_max_connections` / `num_netio_threads` 。	不支持
`num_worker_threads`	`32`	Storage 的 RPC 服务的工作线程数量。	不支持
`max_concurrent_subtasks`	`10`	TaskManager 并发执行子任务的最大个数。	不支持
`snapshot_part_rate_limit`	`10485760`	Raft leader 向 Raft group 中其它成员同步存量数据时的限速。单位：字节/秒。	支持
`snapshot_batch_size`	`1048576`	Raft leader 向 Raft group 中其它成员同步存量数据时每批发送的数据量。单位：字节。	支持
`rebuild_index_part_rate_limit`	`4194304`	重建索引过程中，Raft leader 向 Raft group 中其它成员同步索引数据时的限速。单位：字节/秒。	支持
`rebuild_index_batch_size`	`1048576`	重建索引过程中，Raft leader 向 Raft group 中其它成员同步索引数据时每批发送的数据量。单位：字节。	支持

Copyright © Yueshu Graph Technology - All rights reserved.

rocksdb options 配置

名称	预设值	说明	是否支持运行时动态修改
`rocksdb_db_options`	`{}`	RocksDB database 选项。	不支持
`rocksdb_column_family_options`	`{"write_buffer_size":"67108864", "max_write_buffer_number":"4", "max_bytes_for_level_base":"268435456"}`	RocksDB column family 选项。	不支持
`rocksdb_block_based_table_options`	`{"block_size":"8192"}`	RocksDB block based table 选项。	不支持

rocksdb options 配置的格式为 `{"<option_name>":"<option_value>"}`，多个选项用英文逗号（,）隔开。

`rocksdb_db_options` 和 `rocksdb_column_family_options` 支持的选项如下：

- `rocksdb_db_options`

```
max_total_wal_size
delete_obsolete_files_period_micros
max_background_jobs
stats_dump_period_sec
compaction_readahead_size
writable_file_max_buffer_size
bytes_per_sync
wal_bytes_per_sync
delayed_write_rate
avoid_flush_during_shutdown
max_open_files
stats_persist_period_sec
stats_history_buffer_size
strict_bytes_per_sync
enable_rocksdb_prefix_filtering
enable_rocksdb_whole_key_filtering
rocksdb_filtering_prefix_length
num_compaction_threads
rate_limit
```

- `rocksdb_column_family_options`

```
write_buffer_size
max_write_buffer_number
level0_file_num_compaction_trigger
level0_slowdown_writes_trigger
level0_stop_writes_trigger
target_file_size_base
target_file_size_multiplier
max_bytes_for_level_base
max_bytes_for_level_multiplier
disable_auto_compactions
```

参数的详细说明请参见 RocksDB 官方文档。

black box 配置

Enterpriseonly

仅企业版支持黑匣子功能配置。

名称	预设值	说明	是否支持运行时动态修改
`ng_black_box_switch`	`true`	是否开启黑匣子功能。	不支持
`ng_black_box_home`	`black_box`	存储黑匣子数据的目录名称。	不支持
`ng_black_box_dump_period_seconds`	`5`	黑匣子采集指标的时间间隔。单位：秒。	不支持
`ng_black_box_file_lifetime_seconds`	`1800`	采集指标后生成文件的保存时间。单位：秒	支持

Copyright © Yueshu Graph Technology - All rights reserved.

memory tracker 配置

有关 Memory Tracker 的详细信息，请参见图数据库 NebulaGraph 的内存管理实践之 Memory Tracker。

名称	预设值	说明	是否支持运行时动态修改
`memory_tracker_limit_ratio`	`0.8`	取值可设置为：`(0, 1]` 、`2` 、`3` 。 `(0, 1]`：可用内存的百分比，当可用内存低于该值时，悦数图数据库会停止接受查询。 计算公式：可用内存/(总内存 - 保留内存)。 注意：对于混合部署的集群，需要根据实际情况调小该参数。例如，当预期 Graphd 只占用 50% 的内存时，该参数的值可设置为小于 `0.5` 。 `2`：动态自适应模式（Dynamic Self Adaptive），Memory Tracker 会根据系统当前的可用内存，动态调整可用内存。 注意：此功能为实验性功能，由于动态自适应不能做到实时监控操作系统内存使用情况，在一些大内存分配的场景，还是会存在 OOM 可能。 `3`：关掉 Memory Tracker，Memory Tracker 将只记录内存使用情况，即使超过限额，也不会干预执行。	支持
`memory_tracker_untracked_reserved_memory_mb`	`50`	保留内存的大小，单位：MB。	支持
`memory_tracker_detail_log`	`false`	是否定期生成较详细的内存跟踪日志。当值为 `true` 时，会定期生成内存跟踪日志。	支持
`memory_tracker_detail_log_interval_ms`	`60000`	内存跟踪日志的生成时间间隔，单位：毫秒。仅当 `memory_tracker_detail_log` 为 `true` 时，该参数生效。	支持
`memory_purge_enabled`	`true`	是否定期开启内存清理功能。当值为 `true` 时，会定期清理内存。	支持
`memory_purge_interval_seconds`	`10`	内存清理的时间间隔，单位：秒。`memory_purge_enabled` 为 `true` 时，该参数生效。	支持

超级节点处理（出入边数量极多的点）

在每个点出发的查询获取到边时，直接截断。目的是避免超级节点的邻边过多，单个查询占用过多的硬盘和内存。截取前 `max_edge_returned_per_vertex` 个边，多余的边不返回。该参数作用于全局，不用于单个 space。

属性名	默认值	说明	是否支持运行时动态修改
max_edge_returned_per_vertex	`2147483647`	每个稠密点，最多返回多少条边，多余的边截断不返回。初始配置文件中未设置该参数，如需使用请手动添加。	不支持

数据量大而内存不够时

> **Warning**
>
> 一个图空间至少占用大约 300 MB 的内存。

如果数据量很大但内存不够，则可以尝试把 storage 配置中的 `enable_partitioned_index_filter` 设置为 `true` ，减少 `rocksdb_block_cache` 。但由于缓存了较少的 RocksDB 索引，性能可能会受影响。此外，当通过语句拉取大量数据时，仍存在 OOM 的可能。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7.1.5 Linux 内核配置

本文介绍与悦数图数据库相关的 Linux 内核配置，并介绍如何修改配置。

资源控制

ULIMIT 注意事项

命令 `ulimit` 用于为当前 shell 会话设置资源阈值，注意事项如下：

- `ulimit` 所做的更改仅对当前会话或子进程生效。
- 资源的阈值（软阈值）不能超过硬阈值。
- 普通用户不能使用命令调整硬阈值，即使使用 `sudo` 也不能调整。
- 修改系统级别或调整硬性阈值，请编辑文件 `/etc/security/limits.conf` 。这种方式需要重新登录才生效。

ULIMIT -C

`ulimit -c` 用于限制 core 文件的大小，建议设置为 `unlimited` ，命令如下：

```
ulimit -c unlimited
```

ULIMIT -N

`ulimit -n` 用于限制打开文件的数量，建议设置为超过 10 万，例如：

```
ulimit -n 130000
```

内存

VM.SWAPPINESS

`vm.swappiness` 是触发虚拟内存（swap）的空闲内存百分比。值越大，使用 swap 的可能性就越大，建议设置为 0，表示首先删除页缓存。需要注意的是，0 表示尽量不使用 swap。

VM.MIN_FREE_KBYTES

`vm.min_free_kbytes` 用于设置 Linux 内核保留的最小空闲千字节数。如果系统内存足够，建议设置较大值。例如物理内存为 128 GB，可以将 `vm.min_free_kbytes` 设置为 5 GB。如果值太小，会导致系统无法申请足够的连续物理内存。

VM.MAX_MAP_COUNT

`vm.max_map_count` 用于限制单个进程的 VMA（虚拟内存区域）数量。默认值为 `65530` ，对于绝大多数应用程序来说已经足够。如果应用程序因为内存消耗过大而报错，请增大本参数的值。

VM.DIRTY_*

`vm.dirty_*` 是一系列控制系统脏数据缓存的参数。对于写密集型场景，用户可以根据需要进行调整（吞吐量优先或延迟优先），建议使用系统默认值。

TRANSPARENT HUGE PAGE

为了降低延迟，用户必须关闭 THP（transparent huge page）。命令如下：

```
root# echo never > /sys/kernel/mm/transparent_hugepage/enabled
root# echo never > /sys/kernel/mm/transparent_hugepage/defrag
root# swapoff -a && swapon -a
```

为了防止系统重启后该配置失效，可以在GRUB配置文件或 `/etc/rc.local` 中添加相关配置，使系统启动时自动关闭 THP。

Copyright © Yueshu Graph Technology - All rights reserved.

网络

NET.IPV4.TCP_SLOW_START_AFTER_IDLE

`net.ipv4.tcp_slow_start_after_idle` 默认值为 1，会导致闲置一段时间后拥塞窗口超时，建议设置为 `0`，尤其适合大带宽高延迟场景。

NET.CORE.SOMAXCONN

`net.core.somaxconn` 用于限制 socket 监听的连接队列数量。默认值为 `128`。对于有大量突发连接的场景，建议设置为不低于 `1024`。

NET.IPV4.TCP_MAX_SYN_BACKLOG

`net.ipv4.tcp_max_syn_backlog` 用于限制处于 SYN_RECV（半连接）状态的 TCP 连接数量。默认值为 `128`。对于有大量突发连接的场景，建议设置为不低于 `1024`。

NET.CORE.NETDEV_MAX_BACKLOG

`net.core.netdev_max_backlog` 用于限制队列中数据包的数量。默认值为 `1000`，建议设置为 `10000` 以上，尤其是万兆网卡。

NET.IPV4.TCP_KEEPALIVE_*

`net.ipv4.tcp_keepalive_*` 是一系列保持 TCP 连接存活的参数。对于使用四层透明负载均衡的应用程序，如果空闲连接异常断开，请增大 `tcp_keepalive_time` 和 `tcp_keepalive_intvl` 的值。

NET.IPV4.TCP_WMEM/RMEM

TCP 套接字发送/接收缓冲池的最小、最大、默认空间。对于大连接，建议设置为 `带宽 (GB) *往返时延 (ms)`。

SCHEDULER

对于 SSD 设备，建议将 `scheduler` 设置为 `noop` 或者 `none`，路径为 `/sys/block/DEV_NAME/queue/scheduler`。

其他参数

KERNEL.CORE_PATTERN

建议设置为 `core`，并且将 `kernel.core_uses_pid` 设置为 `1`。

修改参数

SYSCTL 命令

- `sysctl <conf_name>`

  查看当前参数值。

- `sysctl -w <conf_name>=<value>`

  临时修改参数值，立即生效，重启后恢复原值。

- `sysctl -p [<file_path>]`

  从指定配置文件里加载 Linux 系统参数，默认从 `/etc/sysctl.conf` 加载。

PRLIMIT

命令 `prlimit` 可以获取和设置进程资源的限制，结合 `sudo` 可以修改硬阈值，例如，`prlimit --nofile=140000 --pid=$$` 调整当前进程允许的打开文件的最大数量为 `140000`，立即生效，此命令仅支持 RedHat 7u 或更高版本。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7.2 日志

7.2.1 运行日志

运行日志通常提供给 DBA 或开发人员查看，当系统出现故障，DBA 或开发人员可以根据运行日志定位问题。

悦数图数据库默认使用 glog 打印运行日志，使用 gflags 控制日志级别，并在运行时通过 HTTP 接口动态修改日志级别，方便跟踪问题。

运行日志目录

运行日志的默认目录为 `/usr/local/nebula/logs/` 。

如果在悦数图数据库运行过程中删除运行日志目录，日志不会继续打印，但是不会影响业务。重启服务可以恢复正常。

配置说明

- `minloglevel`：最小日志级别，即不会记录低于这个级别的日志。可选值为 `0` （INFO）、`1` （WARNING）、`2` （ERROR）、`3` （FATAL）。建议在调试时设置为 `0` ，生产环境中设置为 `1` 。如果设置为 `4` ，悦数图数据库不会记录任何日志。
- `v`：日志详细级别，值越大，日志记录越详细。可选值为 `0` 、`1` 、`2` 、`3` 。

Meta 服务、Graph 服务和 Storage 服务的日志级别可以在各自的配置文件中查看，默认路径为 `/usr/local/nebula/etc/` 。

查看运行日志级别

使用如下命令查看当前所有的 gflags 参数（包括日志参数）：

```
$ curl <ws_ip>:<ws_port>/flags
```

参数	说明
`ws_ip`	HTTP 服务的 IP 地址，可以在配置文件中查看。默认值为 `127.0.0.1` 。
`ws_port`	HTTP 服务的端口，可以在配置文件中查看。默认值分别为 `19559` （Meta）、`19669` （Graph） `19779` （Storage）。

示例如下：

- 查看 Meta 服务当前的最小日志级别：

```
$ curl 127.0.0.1:19559/flags | grep 'minloglevel'
```

- 查看 Storage 服务当前的日志详细级别：

```
$ curl 127.0.0.1:19779/flags | grep -w 'v'
```

修改运行日志级别

使用如下命令修改运行日志级别：

Copyright © Yueshu Graph Technology - All rights reserved.

```
$ curl -X PUT -H "Content-Type: application/json" -d '{"<key>":<value>[,"<key>":<value>]}' "<ws_ip>:<ws_port>/flags"
```

参数	说明
`key`	待修改的运行日志类型，可选值请参见配置说明。
`value`	运行日志级别，可选值请参见配置说明。
`ws_ip`	HTTP 服务的 IP 地址，可以在配置文件中查看。默认值为 `127.0.0.1` 。
`ws_port`	HTTP 服务的端口，可以在配置文件中查看。默认值分别为 `19559` （Meta） 、 `19669` （Graph） `19779` （Storage） 。

示例如下：

```
$ curl -X PUT -H "Content-Type: application/json" -d '{"minloglevel":0,"v":3}' "127.0.0.1:19779/flags" # storaged
$ curl -X PUT -H "Content-Type: application/json" -d '{"minloglevel":0,"v":3}' "127.0.0.1:19669/flags" # graphd
$ curl -X PUT -H "Content-Type: application/json" -d '{"minloglevel":0,"v":3}' "127.0.0.1:19559/flags" # metad
```

如果在悦数图数据库运行时修改了运行日志级别，重启服务后会恢复为配置文件中设置的级别，如果需要永久修改，请修改配置文件。

RocksDB 运行日志

RocksDB 的运行日志通常在 `/usr/local/nebula/data/storage/nebula/$id/data/LOG` , 其中 `$id` 为实例号。该日志通常用于调试 RocksDB 参数。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

7.2.2 审计日志

悦数图数据库的审计日志功能可以将 Graph 服务接受到的所有操作进行分类存储，然后提供给终端用户查看，终端用户可以根据需要，追踪指定类型的操作。

Enterpriseonly

仅企业版支持本功能。

日志类别

类别	语句	说明
`login`	-	客户端尝试连接 Graph 服务时，记录相关信息。
`exit`	-	断开与 Graph 服务的连接时，记录相关信息。
`ddl`	`CREATE SPACE` 、 `DROP SPACE` 、 `CREATE TAG` 、 `DROP TAG` 、 `ALTER TAG` 、 `DELETE TAG` 、 `CREATE EDGE` 、 `DROP EDGE` 、 `ALTER EDGE` 、 `CREATE INDEX` 、 `DROP INDEX` 、 `CREATE FULLTEXT INDEX` 、 `DROP FULLTEXT INDEX`	记录 DDL 语句的信息。
`dql`	`MATCH` 、 `LOOKUP` 、 `GO` 、 `FETCH` 、 `GET SUBGRAPH` 、 `FIND PATH` 、 `UNWIND` 、 `GROUP BY` 、 `ORDER BY` 、 `YIELD` 、 `LIMIT` 、 `RETURN` 、 `REBUILD INDEX` 、 `REBUILD FULLTEXT INDEX`	记录 DQL 语句的信息。
`dml`	`INSERT VERTEX` 、 `DELETE VERTEX` 、 `UPDATE VERTEX` 、 `UPSERT VERTEX` 、 `INSERT EDGE` 、 `DELETE EDGE` 、 `UPDATE EDGE` 、 `UPSERT EDGE`	记录 DML 语句的信息。
`dcl`	`CREATE USER` 、 `GRANT ROLE` 、 `REVOKE ROLE` 、 `CHANGE PASSWORD` 、 `ALTER USER` 、 `DROP USER` 、 `CREATE SNAPSHOT` 、 `DROP SNAPSHOT` 、 `ADD LISTENER` 、 `REMOVE LISTENER` 、 `BALANCE` 、 `SUBMIT JOB` 、 `STOP JOB` 、 `RECOVER JOB` 、 `ADD DRAINER` 、 `REMOVE DRAINER` 、 `SIGN IN DRAINER SERVICE` 、 `SIGN OUT DRAINER SERVICE` 、 `DOWNLOAD HDFS` 、 `INGEST`	记录 DCL 语句的信息。
`util`	`SHOW HOSTS` 、 `SHOW USERS` 、 `SHOW ROLES` 、 `SHOW SNAPSHOTS` 、 `SHOW SPACES` 、 `SHOW PARTS` 、 `SHOW TAGS` 、 `SHOW EDGES` 、 `SHOW INDEXES` 、 `SHOW CREATE SPACE` 、 `SHOW CREATE TAG/EDGE` 、 `SHOW CREATE INDEX` 、 `SHOW INDEX STATUS` 、 `SHOW LISTENER` 、 `SHOW TEXT SEARCH CLIENTS` 、 `SHOW DRAINER CLIENTS` 、 `SHOW FULLTEXT INDEXES` 、 `SHOW CONFIGS` 、 `SHOW CHARSET` 、 `SHOW COLLATION` 、 `SHOW STATS` 、 `SHOW SESSIONS` 、 `SHOW META LEADER` 、 `SHOW DRAINERS` 、 `SHOW QUERIES` 、 `SHOW JOB` 、 `SHOW JOBS` 、 `DESCRIBE INDEX` 、 `DESCRIBE EDGE` 、 `DESCRIBE TAG` 、 `DESCRIBE SPACE` 、 `DESCRIBE USER` 、 `USE SPACE` 、 `SIGN IN TEXT SERVICE` 、 `SIGN OUT TEXT SERVICE` 、 `EXPLAIN` 、 `PROFILE` 、 `KILL QUERY`	记录工具类语句的信息。
`unknown`	-	记录未能识别的语句。

设置审计日志

使用审计日志需要修改集群内的所有 Graph 服务的配置（ `nebula-graphd.conf` ），默认路径为 `/usr/local/nebula/etc/nebula-graphd.conf` 。

Note

修改配置后，需要重启 Graph 服务才能生效。

Copyright © Yueshu Graph Technology - All rights reserved.

与审计日志相关的参数说明如下。

参数	预设值	说明
`enable_audit`	`false`	是否开启审计日志。
`audit_log_handler`	`file`	审计日志的存储方案。可选值为 `file`（本地文件）和 `es`（Elasticsearch），支持的 Elasticsearch 版本为 7.x 和 8.x。
`audit_log_file`	`./logs/audit/audit.log`	仅在 `audit_log_handler=file` 时生效。审计日志的存储路径，支持相对路径或绝对路径。
`audit_log_strategy`	`synchronous`	仅在 `audit_log_handler=file` 时生效。审计日志的同步方案。可选值为 `asynchronous` 和 `synchronous` 。设置为 `asynchronous` 时，日志事件使用内存缓冲，不会阻塞主线程，但是可能会因为缓存不够而导致日志缺失；设置为 `synchronous` 时，日志事件每次都刷新并同步到文件中。
`audit_log_max_buffer_size`	`1048576`	仅在 `audit_log_handler=file` 、 `audit_log_strategy=asynchronous` 时生效。审计日志的缓存大小。单位：字节。
`audit_log_format`	`xml`	仅在 `audit_log_handler=file` 时生效。审计日志的格式。可选值为 `xml` 、 `json` 和 `csv` 。
`audit_log_es_address`	-	仅在 `audit_log_handler=es` 时生效。Elasticsearch 服务器的地址。格式为 `IP1:port1, IP2:port2, ...` 。
`audit_log_es_user`	-	仅在 `audit_log_handler=es` 时生效。登录 Elasticsearch 服务器的用户名。
`audit_log_es_password`	-	仅在 `audit_log_handler=es` 时生效。Elasticsearch 用户名对应的密码。
`audit_log_es_batch_size`	`1000`	仅在 `audit_log_handler=es` 时生效。每次发送至 Elasticsearch 服务器的日志条数。
`audit_log_exclude_spaces`	-	不需要记录日志的图空间列表。多个图空间用英文逗号（,）分隔。
`audit_log_categories`	`login,exit`	需要记录日志的分类列表。多个类别用英文逗号（,）分隔。

审计日志格式

以默认路径（`/usr/local/nebula/logs/audit/audit.log`）和默认 XML 格式为例说明各个字段的含义。

Note

如果在悦数图数据库运行过程中删除审计日志目录，日志不会继续打印，但是不会影响程序运行。重启服务审计日志打印可以恢复正常。

```
<AUDIT_RECORD
  CATEGORY="util"
  TIMESTAMP="2022-04-07 02:31:38"
  TERMINAL=""
  CONNECTION_ID="1649298693144580"
  CONNECTION_STATUS="0"
  CONNECTION_MESSAGE=""
  USER="root"
  CLIENT_HOST="127.0.0.1"
  HOST="192.168.8.111"
  SPACE=""
  QUERY="use basketballplayer1"
  QUERY_STATUS="-1005"
  QUERY_MESSAGE="SpaceNotFound: "
/>
<AUDIT_RECORD
  CATEGORY="util"
  TIMESTAMP="2022-04-07 02:31:39"
  TERMINAL=""
  CONNECTION_ID="1649298693144580"
  CONNECTION_STATUS="0"
  CONNECTION_MESSAGE=""
  USER="root"
  CLIENT_HOST="127.0.0.1"
  HOST="192.168.8.111"
  SPACE=""
  QUERY="use basketballplayer"
  QUERY_STATUS="0"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  QUERY_MESSAGE=""
/>
```

字段	说明
`CATEGORY`	日志类别。
`TIMESTAMP`	日志生成时间。
`TERMINAL`	保留字段，暂不支持。
`CONNECTION_ID`	连接的会话ID。
`CONNECTION_STATUS`	连接的状态码。`0` 表示成功，其他数字代表不同的错误信息。
`CONNECTION_MESSAGE`	如果连接出错，会显示报错信息。
`USER`	连接的用户名。
`CLIENT_HOST`	客户端的 IP 地址。
`HOST`	连接的机器的 IP 地址。
`SPACE`	执行查询的图空间。
`QUERY`	查询语句。
`QUERY_STATUS`	查询状态。`0` 表示成功，其他数字代表不同的错误信息。
`QUERY_MESSAGE`	如果查询出错，会显示报错信息。

视频

- 悦数图数据库的审计日志（3 分 53 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

8. 监控

8.1 查询悦数图数据库监控指标

悦数图数据库支持多种方式查询服务的监控指标，本文将介绍最基础的方式，即通过 HTTP 端口查询。

8.1.1 监控指标结构说明

悦数图数据库的每个监控指标都由三个部分组成，中间用英文句号（.）隔开，例如 `num_queries.sum.600` 。不同的悦数图数据库服务支持查询的监控指标也不同。指标结构的说明如下。

类别	示例	说明
指标名称	`num_queries`	简单描述指标的含义。
统计类型	`sum`	指标统计的方法。当前支持 SUM、AVG、RATE 和 P 分位数（P75、P95、P99、P999）。
统计时间	`600`	指标统计的时间范围，当前支持 5 秒、60 秒、600 秒和 3600 秒，分别表示最近 5 秒、最近 1 分钟、最近 10 分钟和最近 1 小时。

8.1.2 通过 HTTP 端口查询监控指标

语法

```
curl -G "http://<ip>:<port>/stats?stats=<metric_name_list> [&format=json]"
```

选项	说明
`ip`	服务器的 IP 地址，可以在安装目录内查看配置文件获取。
`port`	服务器的 HTTP 端口，可以在安装目录内查看配置文件获取。默认情况下，Meta 服务端口为 19559，Graph 服务端口为 19669，Storage 服务端口为 19779。
`metric_name_list`	监控指标名称，多个监控指标用英文逗号（,）隔开。
`&format=json`	将结果以 JSON 格式返回。

Note

如果悦数图数据库服务部署在容器中，需要执行 `docker-compose ps` 命令查看映射到容器外部的端口，然后通过该端口查询。

查询单个监控指标

查询 Graph 服务中，最近 10 分钟的请求总数。

```
$ curl -G "http://192.168.8.40:19669/stats?stats=num_queries.sum.600"
num_queries.sum.600=400
```

查询多个监控指标

查询 Meta 服务中，最近 1 分钟的心跳平均延迟和最近 10 分钟 P99 心跳（1%最慢的心跳）的平均延迟。

```
$ curl -G "http://192.168.8.40:19559/stats?stats=heartbeat_latency_us.avg.60,heartbeat_latency_us.p99.600"
heartbeat_latency_us.avg.60=281
heartbeat_latency_us.p99.600=985
```


Copyright © Yueshu Graph Technology - All rights reserved.

查询监控指标并以 **JSON** 格式返回

查询 Storage 服务中，最近 10 分钟新增的点数量，并以 JSON 格式返回结果。

```
$ curl -G "http://192.168.8.40:19779/stats?stats=num_add_vertices.sum.600&format=json"
[{"value":1,"name":"num_add_vertices.sum.600"}]
```

查询服务器的所有监控指标

不指定查询某个监控指标时，会返回该服务器上所有的监控指标。

```
$ curl -G "http://192.168.8.40:19559/stats"
heartbeat_latency_us.avg.5=304
heartbeat_latency_us.avg.60=308
heartbeat_latency_us.avg.600=299
heartbeat_latency_us.avg.3600=285
heartbeat_latency_us.p75.5=652
heartbeat_latency_us.p75.60=669
heartbeat_latency_us.p75.600=651
heartbeat_latency_us.p75.3600=642
heartbeat_latency_us.p95.5=930
heartbeat_latency_us.p95.60=963
heartbeat_latency_us.p95.600=933
heartbeat_latency_us.p95.3600=929
heartbeat_latency_us.p99.5=986
heartbeat_latency_us.p99.60=1409
heartbeat_latency_us.p99.600=989
heartbeat_latency_us.p99.3600=986
num_heartbeats.rate.5=0
num_heartbeats.rate.60=0
num_heartbeats.rate.600=0
num_heartbeats.rate.3600=0
num_heartbeats.sum.5=2
num_heartbeats.sum.60=40
num_heartbeats.sum.600=394
num_heartbeats.sum.3600=2364
...
```

查询图空间监控指标

Graph 服务支持一系列基于图空间的监控指标，对不同图空间的数据分别记录。

图空间指标只能通过查询所有监控指标的形式查询到，例如 `curl -G "http://192.168.8.40:19559/stats"` ，返回结果中以 `{space=space_name}` 的形式包含图空间名称，例如 `num_active_queries{space=basketballplayer}.sum.5=0` 。

Caution

如需开启图空间监控指标，先在 Graph 服务的配置文件中将 `enable_space_level_metrics` 参数的值修改为 `true` ，再启动悦数图数据库。修改配置的详细方式参见配置管理。

Copyright © Yueshu Graph Technology - All rights reserved.

8.1.3 监控指标说明

Graph

参数	说明
`num_active_queries`	活跃的查询语句数的变化数。 计算公式：时间范围内开始执行的语句数减去执行完毕的语句数。
`num_active_sessions`	活跃的会话数的变化数。 计算公式：时间范围内登录的会话数减去登出的会话数。 例如查询num_active_sessions.sum.5，过去 5 秒中登录了 10 个会话数，登出了 30 个会话数，那么该指标值就是 `-20` （10-30）。
`num_aggregate_executors`	聚合（Aggregate）算子执行时间。
`num_auth_failed_sessions_bad_username_password`	因用户名密码错误导验证失败的会话数量。
`num_auth_failed_sessions_out_of_max_allowed`	因为超过 `FLAG_OUT_OF_MAX_ALLOWED_CONNECTIONS` 参数导致的验证登录的失败的会话数量。
`num_auth_failed_sessions`	登录验证失败的会话数量。
`num_indexscan_executors`	索引扫描（IndexScan）算子执行时间。
`num_killed_queries`	被终止的查询数量。
`num_opened_sessions`	服务端建立过的会话数量。
`num_queries`	查询次数。
`num_query_errors_leader_changes`	因查询错误而导致的 Leader 变更的次数。
`num_query_errors`	查询错误次数。
`num_reclaimed_expired_sessions`	服务端主动回收的过期的会话数量。
`num_rpc_sent_to_metad_failed`	Graphd 服务发给 Metad 的 RPC 请求失败的数量。
`num_rpc_sent_to_metad`	Graphd 服务发给 Metad 服务的 RPC 请求数量。
`num_rpc_sent_to_storaged_failed`	Graphd 服务发给 Storaged 服务的 RPC 请求失败的数量。
`num_rpc_sent_to_storaged`	Graphd 服务发给 Storaged 服务的 RPC 请求数量。
`num_sentences`	Graphd 服务接收的语句数。
`num_slow_queries`	慢查询次数。
`num_sort_executors`	排序（Sort）算子执行时间。
`optimizer_latency_us`	优化器阶段延迟时间。
`query_latency_us`	查询延迟时间。
`slow_query_latency_us`	慢查询延迟时间。
`num_queries_hit_memory_watermark`	达到内存水位线的语句的数量。

Copyright © Yueshu Graph Technology - All rights reserved.

Meta

参数	说明
`commit_log_latency_us`	Raft 协议中 Commit 日志的延迟时间。
`commit_snapshot_latency_us`	Raft 协议中 Commit 快照的延迟时间。
`heartbeat_latency_us`	心跳延迟时间。
`num_heartbeats`	心跳次数。
`num_raft_votes`	Raft 协议中投票的次数。
`transfer_leader_latency_us`	Raft 协议中转移 Leader 的延迟时间。
`num_agent_heartbeats`	AgentHBProcessor 心跳次数。
`agent_heartbeat_latency_us`	AgentHBProcessor 延迟时间。
`replicate_log_latency_us`	Raft 复制日志至大多数节点的延迟。
`num_send_snapshot`	Raft 发送快照至其他节点的次数。
`append_log_latency_us`	Raft 复制日志到单个节点的延迟。
`append_wal_latency_us`	Raft 写入单条 WAL 的延迟。
`num_grant_votes`	Raft 投票给其他节点的次数。
`num_start_elect`	Raft 发起投票的次数。

Copyright © Yueshu Graph Technology - All rights reserved.

Storage

Copyright © Yueshu Graph Technology - All rights reserved.

参数	说明
`add_edges_latency_us`	添加边的延迟时间。
`add_vertices_latency_us`	添加点的延迟时间。
`commit_log_latency_us`	Raft 协议中 Commit 日志的延迟时间。
`commit_snapshot_latency_us`	Raft 协议中 Commit 快照的延迟时间。
`delete_edges_latency_us`	删除边的延迟时间。
`delete_vertices_latency_us`	删除点的延迟时间。
`get_neighbors_latency_us`	查询邻居延迟时间。
`get_dst_by_src_latency_us`	通过起始点获取终点的延迟时间。
`num_get_prop`	GetPropProcessor 执行的次数。
`num_get_neighbors_errors`	GetNeighborsProcessor 执行出错的次数。
`num_get_dst_by_src_errors`	GetDstBySrcProcessor 执行出错的次数。
`get_prop_latency_us`	GetPropProcessor 执行的延迟时间。
`num_edges_deleted`	删除的边数量。
`num_edges_inserted`	插入的边数量。
`num_raft_votes`	Raft 协议中投票的次数。
`num_rpc_sent_to_metad_failed`	Storaged 服务发给 Metad 服务的 RPC 请求失败的数量。
`num_rpc_sent_to_metad`	Storaged 服务发给 Metad 服务的 RPC 请求数量。
`num_tags_deleted`	删除的 Tag 数量。
`num_vertices_deleted`	删除的点数量。
`num_vertices_inserted`	插入的点数量。
`transfer_leader_latency_us`	Raft 协议中转移 Leader 的延迟时间。
`lookup_latency_us`	LookupProcessor 执行的延迟时间。
`num_lookup_errors`	LookupProcessor 执行时出错的次数。
`num_scan_vertex`	ScanVertexProcessor 执行的次数。
`num_scan_vertex_errors`	ScanVertexProcessor 执行时出错的次数。
`update_edge_latency_us`	UpdateEdgeProcessor 执行的延迟时间。
`num_update_vertex`	UpdateVertexProcessor 执行的次数。
`num_update_vertex_errors`	UpdateVertexProcessor 执行时出错的次数。
`kv_get_latency_us`	Getprocessor 的延迟时间。
`kv_put_latency_us`	PutProcessor 的延迟时间。
`kv_remove_latency_us`	RemoveProcessor 的延迟时间。
`num_kv_get_errors`	GetProcessor 执行出错次数。
`num_kv_get`	GetProcessor 执行次数。
`num_kv_put_errors`	PutProcessor 执行出错次数。
`num_kv_put`	PutProcessor 执行次数。

Copyright © Yueshu Graph Technology - All rights reserved.

参数	说明
`num_kv_remove_errors`	RemoveProcessor 执行出错次数。
`num_kv_remove`	RemoveProcessor 执行次数。
`forward_tranx_latency_us`	传输延迟时间。
`scan_edge_latency_us`	ScanEdgeProcessor 执行的延迟时间。
`num_scan_edge_errors`	ScanEdgeProcessor 执行时出错的次数。
`num_scan_edge`	ScanEdgeProcessor 执行的次数。
`scan_vertex_latency_us`	ScanVertexProcessor 执行的延迟时间。
`num_add_edges`	添加边的次数。
`num_add_edges_errors`	添加边时出错的次数。
`num_add_vertices`	添加点的次数。
`num_start_elect`	Raft 发起投票的次数
`num_add_vertices_errors`	添加点时出错的次数。
`num_delete_vertices_errors`	删除点时出错的次数。
`append_log_latency_us`	Raft 复制日志到单个节点的延迟。
`num_grant_votes`	Raft 投票给其他节点的次数。
`replicate_log_latency_us`	Raft 复制日志到大多数节点的延迟。
`num_delete_tags`	删除 Tag 的次数。
`num_delete_tags_errors`	删除 Tag 时出错的次数。
`num_delete_edges`	删除边的次数。
`num_delete_edges_errors`	删除边时出错的次数。
`num_send_snapshot`	发送快照的次数。
`update_vertex_latency_us`	UpdateVertexProcessor 执行的延迟时间。
`append_wal_latency_us`	Raft 写入单条 WAL 的延迟。
`num_update_edge`	UpdateEdgeProcessor 执行的次数。
`delete_tags_latency_us`	删除 Tag 的延迟时间。
`num_update_edge_errors`	UpdateEdgeProcessor 执行时出错的次数。
`num_get_neighbors`	GetNeighborsProcessor 执行的次数。
`num_get_dst_by_src`	GetDstBySrcProcessor 执行的次数。
`num_get_prop_errors`	GetPropProcessor 执行时出错的次数。
`num_delete_vertices`	删除点的次数。
`num_lookup`	LookupProcessor 执行的次数。
`num_sync_data`	Storage 同步 Drainer 数据的次数。
`num_sync_data_errors`	Storage 同步 Drainer 数据出错的次数。
`sync_data_latency_us`	Storage 同步 Drainer 数据的延迟时间。

Copyright © Yueshu Graph Technology - All rights reserved.

图空间级别监控指标

Note

图空间级别监控指标是动态创建的，只有当图空间内触发该行为时，对应的指标才会创建，用户才能查询到。

参数	说明
`num_active_queries`	当前正在执行的查询数。
`num_queries`	查询次数。
`num_sentences`	Graphd 服务接收的语句数。
`optimizer_latency_us`	优化器阶段延迟时间。
`query_latency_us`	查询延迟时间。
`num_slow_queries`	慢查询次数。
`num_query_errors`	查询报错语句数量。
`num_query_errors_leader_changes`	因查询错误而导致的 Leader 变更的次数。
`num_killed_queries`	被终止的查询数量。
`num_aggregate_executors`	聚合（Aggregate）算子执行时间。
`num_sort_executors`	排序（Sort）算子执行时间。
`num_indexscan_executors`	索引扫描（IndexScan）算子执行时间。
`num_auth_failed_sessions_bad_username_password`	因用户名密码错误导验证失败的会话数量。
`num_auth_failed_sessions`	登录验证失败的会话数量。
`num_opened_sessions`	服务端建立过的会话数量。
`num_queries_hit_memory_watermark`	达到内存水位线的语句的数量。
`num_reclaimed_expired_sessions`	服务端主动回收的过期的会话数量。
`num_rpc_sent_to_metad_failed`	Graphd 服务发给 Metad 的 RPC 请求失败的数量。
`num_rpc_sent_to_metad`	Graphd 服务发给 Metad 服务的 RPC 请求数量。
`num_rpc_sent_to_storaged_failed`	Graphd 服务发给 Storaged 服务的 RPC 请求失败的数量。
`num_rpc_sent_to_storaged`	Graphd 服务发给 Storaged 服务的 RPC 请求数量。
`slow_query_latency_us`	慢查询延迟时间。

Copyright © Yueshu Graph Technology - All rights reserved.

单进程指标

Graph、Meta 和 Storage 服务都有自身的单进程指标。

参数	说明
`context_switches_total`	上下文切换数量。
`cpu_seconds_total`	基于用户时间和系统时间的 CPU 使用量。
`memory_bytes_gauge`	使用的内存字节数。
`open_filedesc_gauge`	文件描述符的数量。
`read_bytes_total`	读取的字节数量。
`write_bytes_total`	写入的字节数量。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

8.2 RocksDB 统计数据

悦数图数据库使用 RocksDB 作为底层存储，本文介绍如何收集和展示悦数图数据库的 RocksDB 统计信息。

8.2.1 启用 RocksDB

RocksDB 统计功能默认关闭，启动 RocksDB 统计功能，你需要：

1. 修改 `nebula-storaged.conf` 文件中 `--enable_rocksdb_statistics` 参数为 `true`。配置默认文件目录为 `/use/local/nebula/etc`。
2. 重启服务使修改生效。

8.2.2 获取 RocksDB 统计信息

用户可以使用存储服务中的内置 HTTP 服务来获取以下类型的统计信息，且支持返回 JSON 格式的结果：

- 所有统计信息。
- 指定条目的信息。

8.2.3 示例

使用以下命令获取所有 RocksDB 统计信息：

```
curl -L "http://${storage_ip}:${port}/rocksdb_stats"
```

例如：

```
curl -L "http://172.28.2.1:19779/rocksdb_stats"

rocksdb.blobdb.blob.file.bytes.read=0
rocksdb.blobdb.blob.file.bytes.written=0
rocksdb.blobdb.blob.file.bytes.synced=0
...
```

使用以下命令获取部分 RocksDB 统计信息：

```
curl -L "http://${storage_ip}:${port}/rocksdb_stats?stats=${stats_name}"
```

例如使用以下语句获取 `rocksdb.bytes.read` 和 `rocksdb.block.cache.add` 的信息。

```
curl -L "http://172.28.2.1:19779/rocksdb_stats?stats=rocksdb.bytes.read,rocksdb.block.cache.add"

rocksdb.block.cache.add=14
rocksdb.bytes.read=1632
```

使用以下命令获取部分 JSON 格式的 RocksDB 统计信息：

```
curl -L "http://${storage_ip}:${port}/rocksdb_stats?stats=${stats_name}&format=json"
```

例如使用以下语句获取 `rocksdb.bytes.read` 和 `rocksdb.block.cache.add` 的统计信息并返回 JSON 的格式数据。

```
curl -L "http://172.28.2.1:19779/rocksdb_stats?stats=rocksdb.bytes.read,rocksdb.block.cache.add&format=json"

[
  {
    "rocksdb.block.cache.add": 1
  },
  {
    "rocksdb.bytes.read": 160
  }
]
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

8.3 悦数黑匣子

8.3.1 黑匣子介绍

悦数图数据库自带黑匣子功能，可以定期收集和归档操作系统和服务指标的数据。当服务出现故障时，可以帮助用户在无直连网络的情况下快速定位问题和分析原因。

功能说明

黑匣子在服务器后台以进程运行并收集指标数据，目前仅收集操作系统运行时的一些性能指标（例如，CPU/Memory/Network IO 等相关指标），在未来，我们将支持收集服务指标。关于指标说明，参见 PROC。

开启黑匣子

黑匣子功能默认开启，系统会在悦数图数据库首次启动时在其安装目录下自动创建一个名为 `black_box` 的目录，用于存储收集的指标数据。

用户可关闭黑匣子功能，参见悦数图数据库各个服务的配置文件中的 **Black box** 配置。有关配置文件详情，参见配置管理。

黑匣子文件

目录结构

在 `black_box` 目录下，系统根据当前机器上运行的悦数图数据库的各个服务的进程号，自动创建以相应进程号命名的目录。在每个目录下，默认每 5 秒生成一个二进制文件，记录这段时间内的操作系统性能指标数据。文件名格式为 `black_box.{timestamp_id}.log`，其中 `timestamp_id` 为文件生成时的时间戳。服务重启后，进程号变化会导致日志写到新的目录下。

```
├── 5776 # 服务进程号
│   ├── black_box.1665472900.log # 黑匣子文件
│   ├── black_box.1665472905.log
│   ...
├── 5787
│   ├── black_box.1665728450.log
│   ├── black_box.1665728455.log
│   ...
...
```

黑匣子文件的保留时长默认为 30 分钟（1800 秒），超过 30 分钟的文件会被自动删除。

黑匣子文件的生成间隔和保留时长可在悦数图数据库各个服务的配置文件的 **Black box** 配置中进行配置。有关配置文件详情，参见配置管理。

查看黑匣子文件

查看黑匣子二进制文件需要使用悦数黑匣子工具，还可以使用该工具将二进制文件转换成 CSV 文件并导出查看。关于悦数黑匣子的使用，参见使用悦数黑匣子。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

8.3.2 使用悦数黑匣子

悦数黑匣子是一个帮助用户查看和分析黑匣子数据的工具。本文介绍如何在 Linux 系统中使用悦数黑匣子。

悦数黑匣子的功能

悦数黑匣子提供了以下功能:

- 通过终端用户界面（Terminal User Interface, TUI）查看黑匣子数据。
- 将黑匣子数据导出为 CSV 文件。
- 不同维度查看数据
- 查看一个或多个指标数据。
- 查看某个时间段的数据。
- 查看一个或多个目录/文件的数据，或者同时指定目录和文件查看数据。
- 支持 Linux、macOS 和 Windows 系统。

版本兼容性

悦数图数据库与悦数黑匣子的版本兼容性如下:

悦数图数据库	悦数黑匣子
3.5.1	3.5.0

部署悦数黑匣子

支持多种部署方式，包括 RPM、DEB、TAR、Docker 等。本文以 RPM 部署为例介绍如何部署悦数黑匣子。

1. 获取 RPM 安装包。

> **Enterpriseonly**

联系我们以获取 Nebula-BBox 的安装包。

2. 使用 `sudo rpm -i <rpm>` 命令安装 RPM 包。

例如， `sudo rpm -i nebula-bbox-<version>.x86_64.rpm`

悦数黑匣子以二进制文件（文件名为 `nebula-bbox` ）的形式安装在默认路径 `/usr/bin/` 下。

使用悦数黑匣子

用户可以执行 `nebula-bbox -h/--help` 查看悦数黑匣子的帮助信息。

> **Caution**

对于安装在非默认路径（默认为 `/usr/bin/` ）下的悦数黑匣子，执行 `nebula-bbox` 相关命令时，需指定悦数黑匣子的安装路径。例如，悦数黑匣子安装在 `/usr/bbox` 下，则需执行 `/usr/bbox/nebula-bbox -h` 。

查看悦数黑匣子版本

执行 `nebula-bbox version` 查看悦数黑匣子的版本信息。

Copyright © Yueshu Graph Technology - All rights reserved.

查看采集的所有指标

执行 `nebula-bbox metrics` 命令查看悦数黑匣子采集的所有指标。关于指标的详细信息，参见 PROC(5)。

查看黑匣子文件数据

通过悦数黑匣子，用户可以查看黑匣子文件的指标数据。语法如下：

```
nebula-bbox view [(-o|--output=)tui|csv] [--metrics name[,name ...]] [flags] (FILE | DIRECTORY ...)
```

参数说明

参数	说明
`-o`，`--output`	指定悦数黑匣子显示数据的格式。可选值为 `tui` 和 `csv`。未指定时默认值为 `tui`。 `tui`：输出格式为终端用户界面（Terminal User Interface, TUI）。 `csv`：输出格式为 CSV 文件。
`--metrics`	指定悦数黑匣子输出的指标。可选值为 `nebula-bbox metrics` 命令输出的指标。未指定时默认值为所有指标。 指定多个指标时，用逗号分隔，如 `--metrics <name>,<name>...`。
`flags`	可定义其他的参数项： `--output-file`：当 `-o` 或 `--output` 值为 `csv` 时需指定该参数，用于定义 CSV 文件的存储路径及文件名。 `--start-time`：当根据时间段过滤数据时，定义起始时间。 `--end-time`：当根据时间段过滤数据时，定义结束时间，和 `--start-time` 同时用。 `--duration`：当根据时间段过滤数据时，定义数据采集的时间周期，和 `--start-time` 同时用。不能和 `--end-time` 同时使用。

示例

以下示例均假设悦数图数据库的安装路径为 `/usr/local/nebula`，黑匣子文件的存储路径为 `/usr/local/nebula/black_box`。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

指定指标前，需先执行 `nebula-bbox metrics` 命令查看悦数黑匣子支持的指标。详情参见上文查看采集的所有指标。

场景	示例
查看单个文件的所有指标数据	`nebula-bbox view /usr/local/nebula/black_box/<pid>/black_box.<timestamp>.log`
查看多个文件的所有指标数据	`nebula-bbox view /usr/local/nebula/black_box/<pid>/black_box.<timestamp1>.log /usr/local/nebula/black_box/<pid>/black_box.<timestamp2>.log`
查看所有文件的所有指标数据	`nebula-bbox view /usr/local/nebula/black_box`
查看多个子目录下的所有指标数据	`nebula-bbox view /usr/local/nebula/black_box/<pid1> /usr/local/nebula/black_box/<pid2>`
查看子目录和单个文件的所有指标数据	`nebula-bbox view /usr/local/nebula/black_box/<pid1> /usr/local/nebula/black_box/<pid2> /usr/local/nebula/black_box/<pid3>/black_box.<timestamp>.log`
查看单个文件中指定的指标数据	`nebula-bbox view --metrics <name> /usr/local/nebula/black_box/<pid>/black_box.<timestamp>.log`
在所有文件中查看指定的指标数据	`nebula-bbox view --metrics <name> /usr/local/nebula/black_box`
将所有文件中指定的指标数据导出为 CSV 文件	`nebula-bbox view --metrics <name1>[,<name2> ...] --output csv --output-file <csv_filename>.csv /usr/local/nebula/black_box`
在所有文件中查看多个指定的指标数据	`nebula-bbox view --metrics <name1>[,<name2> ...] /usr/local/nebula/black_box`
查看北京时间 2022 年 9 月 6 日 12：00 起至现在的所有文件数据	`nebula-bbox view --start-time "Tue, 06 Sep 2022 12:00:00 +0800" /usr/local/nebula/black_box` 其中 `Tue, 06 Sep 2022 12:00:00 +0800` 的格式可替换成 `2022-09-06T12:00:00+08:00` 、 `2022-09-06 04:00:00 +0800` 。
查看北京时间 2022 年 9 月 6 日 12：00 起 1 小时内的所有文件数据	`nebula-bbox view --start-time "Tue, 06 Sep 2022 12:00:00 +0800" --duration 1h /usr/local/nebula/black_box` ，时间段单位支持 `h` 、 `m` 、 `s` 。
查看北京时间 2022 年 9 月 6 日 12：00 起至 2022 年 9 月 6 日 13：00 内的所有文件数据	`nebula-bbox view --start-time "2022-09-06 04:00:00 +0800" --end-time "2022-09-06 05:00:00 +0800" /usr/local/nebula/black_box`

TUI 界面及快捷键

TUI 界面以表格形式显示指标数据。页面首行显示指标采集的时间（Time）、悦数图数据库服务的进程号（Pid）、服务类型（ServerName）和各指标名称。

NebulaGraph Black Box View

Time	Pid	ServerName	CPUModeName	CPUCores	CPUThreads	CPUSockets	LoadAvg5m	LoadAvg10m	LoadAvg15m	CurrentKernelScheduleEntries	TotalKernelScheduleEntries	MemInfo:MemTotal	…
Tue Oct 18 09:51:43 CST 2022	1910021	Graph	Intel(R) Xeon(R) Platinum 8352Y CPU @ 2.20GHz	32	128	2	28	22.98	15.05	3	5721	1	1
Tue Oct 18 09:51:43 CST 2022	1910018	Storage	Intel(R) Xeon(R) Platinum 8352Y CPU @ 2.20GHz	32	128	2	28	22.98	15.05	2	5721	1	1
Tue Oct 18 09:51:43 CST 2022	1910065	Graph	Intel(R) Xeon(R) Platinum 8352Y CPU @ 2.20GHz	32	128	2	28	22.98	15.05	1	5721	1	1
Tue Oct 18 09:51:44 CST 2022	1909956	Meta	Intel(R) Xeon(R) Platinum 8352Y CPU @ 2.20GHz	32	128	2	25.76	22.6	14.97	1	5721	1	1
Tue Oct 18 09:52:13 CST 2022	1910018	Storage	Intel(R) Xeon(R) Platinum 8352Y CPU @ 2.20GHz	32	128	2	28	22.98	15.05	2	5721	1	1

Copyright © Yueshu Graph Technology - All rights reserved.

当用户使用 TUI 查看指标数据时，可以使用以下快捷键。

快捷键	说明
`F1`	显示帮助。
`Left`	向左移动。
`Right`	向右移动。
`Down`	向下移动。
`Up`	向上移动。
`Ctrl-A` 或 `Home`	跳转至当前行的首列。
`Ctrl-E` 或 `End`	跳转至当前行的末尾列。
`Ctrl-T`	跳转至首行。
`Ctrl-B`	跳转至末尾行。
`Enter`	显示选中单元格的详细信息。
`Escape`	退出显示单元格详细信息。

FAQ

问：为什么我的 TUI 界面如下显示?

答：遇如上图所示的情况，是因为 Linux 系统字符集不匹配。在命令行中执行 `export LC_CTYPE="en_US.UTF-8"` 可解决问题。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

9. 数据安全

9.1 验证和授权

9.1.1 身份验证

身份验证用于将会话映射到特定用户，从而实现访问控制。

当客户端连接到悦数图数据库时，悦数图数据库会创建一个会话，会话中存储连接的各种信息，如果开启了身份验证，就会将会话映射到对应的用户。

悦数图数据库支持两种身份验证方式：本地身份验证和 LDAP 验证。

本地身份验证

本地身份验证是指在服务器本地存储用户名、加密密码，当用户尝试访问悦数图数据库时，将进行身份验证。

启用本地身份验证

1. 编辑配置文件 `nebula-graphd.conf`（默认目录为 `/usr/local/nebula/etc/`），设置如下参数：

- `--enable_authorize`：是否启用身份验证，可选值：`true`、`false`。

> **Note**
>
> - 默认情况下，身份验证功能是关闭的，用户可以使用 `root` 用户名和任意密码连接到悦数图数据库。
> - 开启身份验证后，默认的 God 角色账号为 `root`，密码为 `nebula`。角色详情请参见内置角色权限。

- `--failed_login_attempts`：可选项，需要手动添加该参数。单个 Graph 节点允许连续输入错误密码的次数。超过该次数时，账户会被锁定。如果有多个 Graph 节点，允许的次数为 `节点数 * 次数`。
- `--password_lock_time_in_secs`：可选项，需要手动添加该参数。多次输入错误密码后，账户被锁定的时间。单位：秒。

2. 重启悦数图数据库服务。

OpenLDAP 验证

OpenLDAP 是轻型目录访问协议（LDAP）的开源实现，可以实现账号集中管理。

启用 OPENLDAP 验证

> **Enterpriseonly**

当前仅企业版支持集成 OpenLDAP 进行身份验证，详情请参见使用 OpenLDAP 进行身份验证。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

9.1.2 用户管理

用户管理是悦数图数据库访问控制中不可或缺的组成部分，本文将介绍用户管理的相关语法。

开启身份验证后，用户需要使用已创建的用户才能连接悦数图数据库，而且连接后可以进行的操作也取决于该用户拥有的角色权限。

Note

- 默认情况下，身份验证功能是关闭的，用户可以使用 root 用户名和任意密码连接到悦数图数据库。
- 修改权限后，对应的用户需要重新登录才能生效。

创建用户（`CREATE USER`）

执行 `CREATE USER` 语句可以创建新的悦数图数据库用户。当前仅 **God** 角色用户（即 `root` 用户）能够执行 `CREATE USER` 语句。

- 语法

```
CREATE USER [IF NOT EXISTS] <user_name> [WITH PASSWORD '<password>'];
```

- `IF NOT EXISTS`： 检测待创建的用户名是否存在，只有不存在时，才会创建新用户。
- `user_name`： 待创建的用户名。长度最多为 16 个字符。
- `password`： 用户名对应的密码。默认密码为空字符串（`''`）。长度最多为 24 个字符。

- 企业版语法

```
CREATE USER [IF NOT EXISTS] <user_name> [WITH PASSWORD '<password>'][WITH IP WHITELIST <ip_list>];
```

- `ip_list` （企业版）： IP 白名单列表。该用户只有通过列表内的 IP 才能连接悦数图数据库。多个 IP 之间用英文逗号（,）分隔。

- 示例

```
nebula> CREATE USER user1 WITH PASSWORD 'nebula';
nebula> SHOW USERS;
+---------+------------------------------+
| Account | IP Whitelist                 |
+---------+------------------------------+
| "root"  | ""                           |
| "user1" | ""                           |
+---------+------------------------------+
```

- 企业版示例

```
nebula> CREATE USER user2 WITH PASSWORD 'nebula' WITH IP WHITELIST 192.168.10.10,192.168.10.12;
nebula> SHOW USERS;
+---------+------------------------------+
| Account | IP Whitelist                 |
+---------+------------------------------+
| "user2" | "192.168.10.10,192.168.10.12" |
+---------+------------------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

授权用户（**`GRANT ROLE`**）

执行 `GRANT ROLE` 语句可以将指定图空间的内置角色权限授予用户。当前仅 **God** 角色用户和 **Admin** 角色用户能够执行 `GRANT ROLE` 语句。角色权限的说明，请参见内置角色权限。

- 语法

```
GRANT ROLE <role_type> ON <space_name> TO <user_name>;
```

- 示例

```
nebula> GRANT ROLE USER ON basketballplayer TO user1;
```

撤销用户权限（**`REVOKE ROLE`**）

执行 `REVOKE ROLE` 语句可以撤销用户的指定图空间的内置角色权限。当前仅 **God** 角色用户和 **Admin** 角色用户能够执行 `REVOKE ROLE` 语句。角色权限的说明，请参见内置角色权限。

- 语法

```
REVOKE ROLE <role_type> ON <space_name> FROM <user_name>;
```

- 示例

```
nebula> REVOKE ROLE USER ON basketballplayer FROM user1;
```

查看指定用户权限（**`DESCRIBE USER`**）

执行 `DESCRIBE USER` 语句可以查看指定用户的角色权限信息。

- 语法

```
DESCRIBE USER <user_name>;
DESC USER <user_name>;
```

- 示例

```
nebula> DESCRIBE USER user1;
+---------+--------------------+
| role    | space              |
+---------+--------------------+
| "ADMIN" | "basketballplayer" |
+---------+--------------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

查看指定空间内用户权限（**SHOW ROLES**）

执行 `SHOW ROLES` 语句可以查看指定空间内的所有用户（除 `root` 以外）和对应角色权限的信息。

- 语法

```
SHOW ROLES IN <space_name>;
```

- 示例

```
nebula> SHOW ROLES IN basketballplayer;
+---------+-----------+
| Account | Role Type |
+---------+-----------+
| "user1" | "ADMIN"   |
+---------+-----------+
```

修改用户密码（**CHANGE PASSWORD**）

执行 `CHANGE PASSWORD` 语句可以修改用户密码，修改时需要提供旧密码和新密码。

- 语法

```
CHANGE PASSWORD <user_name> FROM '<old_password>' TO '<new_password>';
```

- 示例

```
nebula> CHANGE PASSWORD user1 FROM 'nebula' TO 'nebula123';
```


Copyright © Yueshu Graph Technology - All rights reserved.

修改用户密码（`ALTER USER`）

执行 `ALTER USER` 语句可以修改用户密码，修改时不需要提供旧密码。当前仅 **God** 角色用户（即 `root` 用户）能够执行 `ALTER USER` 语句。

- 语法

```
ALTER USER <user_name> WITH PASSWORD '<password>';
```

- 企业版语法

```
ALTER USER <user_name> WITH PASSWORD '<password>' [WITH IP WHITELIST <ip_list>];
```

- 示例

```
nebula> ALTER USER user2 WITH PASSWORD 'nebula';
nebula> SHOW USERS;
+---------+--------------+
| Account | IP Whitelist |
+---------+--------------+
| "root"  | ""           |
| "user1" | ""           |
| "user2" | ""           |
+---------+--------------+
```

- 企业版示例

Enterpriseonly

没有使用 `WITH IP WHITELIST` 时，表示取消 IP 白名单，用户使用任何 IP 都可以连接悦数图数据库。

```
nebula> ALTER USER user2 WITH PASSWORD 'nebula' WITH IP WHITELIST 192.168.10.10;
```

删除用户（`DROP USER`）

执行 `DROP USER` 语句可以删除用户。当前仅 **God** 角色用户能够执行 `DROP USER` 语句。

Note

删除用户不会自动断开该用户当前会话，而且权限仍在当前会话中生效。

- 语法

```
DROP USER [IF EXISTS] <user_name>;
```

- 示例

```
nebula> DROP USER user1;
```


Copyright © Yueshu Graph Technology - All rights reserved.

查看用户列表（**`SHOW USERS`**）

执行 `SHOW USERS` 语句可以查看用户列表。当前仅 **God** 角色用户能够执行 `SHOW USERS` 语句。

- 语法

```
SHOW USERS;
```

- 示例

```
nebula> SHOW USERS;
+---------+-----------------+
| Account | IP Whitelist    |
+---------+-----------------+
| "root"  | ""              |
| "user1" | ""              |
| "user2" | "192.168.10.10" |
+---------+-----------------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

9.1.3 内置角色权限

所谓角色，就是一组相关权限的集合。用户可以把角色分配给创建的用户，从而实现访问控制。

内置角色

悦数图数据库内置了多种角色，说明如下：

- God
- 初始最高权限角色，拥有所有操作的权限。类似于 Linux 中的 `root` 和 Windows 中的 `administrator` 。
- Meta 服务初始化时，会自动创建 God 角色用户 `root` ，密码为 `nebula` 。

> 注意
>
> 请及时修改 `root` 用户的密码，保证数据安全。

- 在 `nebula-graphd.conf` 文件中（默认目录为 `/usr/local/nebula/etc/` ），当 `--enable_authorize` 为 `true` 时：
- 一个集群只能有一个 God 角色用户，该用户可以管理集群内所有图空间。
- 不支持手动授权 God 角色，只能使用默认 God 角色用户 `root` 。

- Admin
- 对权限内的图空间拥有 Schema 和 data 的读写权限。
- 可以将权限内的图空间授权给其他用户。

> 注意
>
> 只能授权低于 ADMIN 级别的角色给其他用户。

- DBA
- 对权限内的图空间拥有 Schema 和 data 的读写权限。
- 无法将权限内的图空间授权给其他用户。

- User
- 对权限内的图空间拥有 Schema 的只读权限。
- 对权限内的图空间拥有 data 的读写权限。

- Guest
- 对权限内的图空间拥有 Schema 和 data 的只读权限。

- Basic
- 对权限内的图空间拥有 Schema 的只读权限。
- （需额外授权）对权限内的图空间拥有 Tag 和 Edge Type 的读写权限。

Copyright © Yueshu Graph Technology - All rights reserved.

Enterpriseonly

Basic 角色仅在企业版中提供。

Note

- 不支持自行创建角色，只能使用默认的内置角色。
- 一个用户在一个图空间内只能拥有一个角色权限。授权用户请参见用户管理。

角色权限

各角色的执行权限如下。

权限	God	Admin	DBA	User	Guest	Basic（企业版）	相关语句
Read space	Y	Y	Y	Y	Y	Y	`USE` 、 `DESCRIBE SPACE`
Read schema	Y	Y	Y	Y	Y	Y	`DESCRIBE TAG` 、 `DESCRIBE EDGE` 、 `DESCRIBE TAG INDEX` 、 `D` `EDGE INDEX`
Write schema	Y	Y	Y				`CREATE TAG` 、 `ALTER TAG` 、 `CREATE EDGE` 、 `ALTER EDGE` 、 `D` `DELETE TAG` 、 `DROP EDGE` 、 `CREATE TAG INDEX` 、 `CREATE ED` `DROP TAG INDEX` 、 `DROP EDGE INDEX`
Write user	Y						`CREATE USER` 、 `DROP USER` 、 `ALTER USER`
Write role	Y	Y					`GRANT` 、 `REVOKE`
Read data	Y	Y	Y	Y	Y	C	`GO` 、 `SET` 、 `PIPE` 、 `MATCH` 、 `ASSIGNMENT` 、 `LOOKUP` 、 `YIE` `ORDER BY` 、 `FETCH VERTICES` 、 `Find` 、 `FETCH EDGES` 、 `FIND` `LIMIT` 、 `GROUP BY` 、 `RETURN`
Write data	Y	Y	Y	Y		C	`INSERT VERTEX` 、 `UPDATE VERTEX` 、 `INSERT EDGE` 、 `UPDATE` `DELETE VERTEX` 、 `DELETE EDGES` 、 `DELETE TAG`
Show operations	Y	Y	Y	Y	Y	Y	`SHOW` 、 `CHANGE PASSWORD`
Job	Y	Y	Y	Y			`SUBMIT JOB COMPACT` 、 `SUBMIT JOB FLUSH` 、 `SUBMIT JOB STA` `STOP JOB` 、 `RECOVER JOB` 、 `BUILD TAG INDEX` 、 `BUILD EDGE` `INGEST` 、 `DOWNLOAD`
Write space	Y						`CREATE SPACE` 、 `DROP SPACE` 、 `CREATE SNAPSHOT` 、 `DROP SN` `BALANCE` 、 `CONFIG`

Enterpriseonly

当前仅企业版支持基于 Basic 角色的细粒度(Tag/Edge type 级别)权限管理。

Caution

Show operations 为特殊操作，只会在自身权限内执行。例如 `SHOW SPACES` ，每个角色都可以执行，但是只会返回自身权限内的图空间。 只有 God 角色可以执行 `SHOW USERS` 和 `SHOW SNAPSHOTS` 语句。

Copyright © Yueshu Graph Technology - All rights reserved.

Basic 角色(企业版)

语法

Caution

进入图空间后才可执行以下命令。

- 授予 Basic 用户 Tag/Edge 权限

```
GRANT { OPTION[,OPTION] } [ TAG {  * | <tag>[,...] } | EDGE {  * | <edge_type>[, ...] }] TO <user_name>;
OPTION = { READ | WRITE }
```

- 取消 Basic 用户 Tag/Edge 权限

```
REVOKE { OPTION[,OPTION] } [ TAG {  * | <tag>[,...] } | EDGE {  * | <edge_type>[, ...] }] TO <user_name>;
OPTION = { READ | WRITE }
```

- 查询用户权限

```
SHOW GRANTS [<user_name>]
```

注意事项

- 默认 Basic 角色无任何 Tag/Edge 读写权限。
- 只有 GOD 和 ADMIN 角色用户能执行授权（GRANT）和撤销权限（REVOKE）操作。
- 只允许用户在指定图空间对 Basic 角色进行授权（GRANT）和撤销权限（REVOKE）操作，不允许授权给其他角色用户。
- Basic 角色不允许插入无 Tag 的点。
- 执行 UPDATE 或 UPSERT 操作必须同时拥有读权限和写权限。

示例

```
# 创建 test 用户
nebula> CREATE USER test WITH PASSWORD 'nebula';

# 为 test 用户创建 Basic 角色权限
nebula> GRANT ROLE BASIC ON basketballplayer TO test;

# 选择图空间 basketballplayer
nebula> use basketballplayer;

# 授予 test 用户 Tag `player` 和 Edge Type `follow` 和 `serve` 的读写权限
# 授予用户指定 Tag/Edge 读写权限必须在指定图空间后
nebula> GRANT READ, WRITE TAG player EDGE follow, serve TO test;

# 查询 test 用户权限
nebula> > SHOW GRANTS test;
+--------+------------+---------------------+------------+---------------------+
| user   | READ(TAG)  | READ(EDGE)          | WRITE(TAG) | WRITE(EDGE)         |
+--------+------------+---------------------+------------+---------------------+
| "test" | ["player"] | ["follow", "serve"] | ["player"] | ["follow", "serve"] |
+--------+------------+---------------------+------------+---------------------+

# 取消 test 用户 所有 Edge Type 的读、写权限
nebula> REVOKE READ,WRITE EDGE * FROM test;

# 查询 test 用户权限
nebula> SHOW GRANTS test;
+--------+------------+------------+------------+-------------+
| user   | READ(TAG)  | READ(EDGE) | WRITE(TAG) | WRITE(EDGE) |
+--------+------------+------------+------------+-------------+
| "test" | ["player"] | []         | ["player"] | []          |
+--------+------------+------------+------------+-------------+

# Basic 角色用户读取无权限的数据时，将会出现以下报错
nebula>  MATCH (v:player)-[:likex]-() RETURN v;
[ERROR (-1008)]: PermissionError: Edge `likex' does not exist or is not readable.
```


Copyright © Yueshu Graph Technology - All rights reserved.

Caution

对于 Basic 角色用户，显式指定无读权限的 Tag/Edge Type 将报错，未显式指定无读权限的 Tag/Edge Type 不会报错。在遍历过程中，所有查询无法读取无权限的 Tag/Edge Type 及其上的属性。Edge Type 的读权限可控制边的扩展行为，在遍历过程中，如果 Edge Type 无权限，将不会进行拓展；Tag 的读权限不控制点的拓展行为，拓展过程中即使点上 Tag 无权限，仍可进行拓展。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

9.1.4 使用 OpenLDAP 进行身份验证

本文介绍如何将悦数图数据库连接到 OpenLDAP 服务器，使用 OpenLDAP 中定义的 DN（Distinguished Name）和密码进行身份验证。

> **Enterpriseonly**
>
> 仅企业版支持本功能。

认证方式

启用 OpenLDAP 身份验证后，输入用户的账号和密码登录悦数图数据库时，悦数图数据库会在 Meta 服务中查找登录账号是否存在，如果账号存在，再根据认证方式去 OpenLDAP 中找到对应的 DN，验证密码。

OpenLDAP 支持的认证方式有两种：简单绑定认证和搜索绑定认证。

简单绑定认证（**SIMPLEBINDAUTH**）

简单绑定认证会根据登录账号和 Graph 服务配置信息，拼接成 OpenLDAP 可以识别的 DN，然后根据 DN 和密码，在 OpenLDAP 上进行验证。

搜索绑定认证（**SEARCHBINDAUTH**）

搜索绑定认证会读取 Graph 服务配置信息，查询配置信息中的 `uid` 和登录账号是否匹配，如果匹配，就读取这个 DN，然后用 DN 和密码，在 OpenLDAP 上进行验证。

> **Caution**
>
> 悦数图数据库仅支持使用 OpenLDAP 中的 `uid` 属性指定用户名进行搜索绑定认证。

前提条件

- 已安装 OpenLDAP。
- 已在 OpenLDAP 上导入用户的账号和密码信息。
- OpenLDAP 所在服务器已开放相应认证端口。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

以 OpenLDAP 上已存在的账号 `test2` 、密码 `passwdtest2` 为例进行演示。

1. 连接悦数图数据库，创建与 OpenLDAP 中对应的影子账号 `test2` 并授权。

```
nebula> CREATE USER test2 WITH PASSWORD '';
nebula> GRANT ROLE ADMIN ON basketballplayer TO test2;
```

Note

悦数图数据库内创建用户时，密码可以任意设置。

2. 编辑配置文件 `nebula-graphd.conf` （默认目录为 `/usr/local/nebula/etc/` ）：

- 简单绑定认证（推荐）

```
# 是否从配置文件获取配置信息。
--local_config=true
# 是否开启身份验证
--enable_authorize=true
# 身份验证方式：password、ldap、cloud
--auth_type=ldap
# OpenLDAP 服务器地址
--ldap_server=192.168.8.211
# OpenLDAP 服务器端口
--ldap_port=389
# OpenLDAP 中的 Schema 名称
--ldap_scheme=ldap
# DN 前缀
--ldap_prefix=uid=
# DN 后缀
--ldap_suffix=,ou=it,dc=sys,dc=com
```

- 搜索绑定认证

```
# 是否从配置文件获取配置信息。
--local_config=true
# 是否开启身份验证
--enable_authorize=true
# 身份验证方式：password、ldap、cloud
--auth_type=ldap
# OpenLDAP 服务器地址
--ldap_server=192.168.8.211
# OpenLDAP 服务器端口
--ldap_port=389
# OpenLDAP 中的 Schema 名称
--ldap_scheme=ldap
# 绑定目标对象的 DN
--ldap_basedn=ou=it,dc=sys,dc=com
# OpenLDAP 服务器登录用户名。支持匿名访问时，该项为选填，否则为必填。
--ldap_binddn=cn=admin,dc=example,dc=org
# OpenLDAP 服务器登录密码。支持匿名访问时，该项为选填，否则为必填。
--ldap_bindpasswd=admin
```

3. 重启悦数图数据库服务，让新配置生效。
4. 进行登录测试。

```
$ ./nebula-console --addr 127.0.0.1 --port 9669 -u test2 -p passwdtest2
2021/09/08 03:49:39 [INFO] connection pool is initialized successfully

Welcome to NebulaGraph!
```

Note

开启 OpenLDAP 进行身份验证后，可以使用 OpenLDAP 里设置的账密进行登录。但是需要确保悦数图数据库的本地账号内有相同的用户名。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

9.2 SSL 加密

悦数图数据库支持在客户端、Graph 服务、Meta 服务和 Storage 服务之间进行 SSL 加密传输，本文介绍如何设置 SSL 加密。

9.2.1 注意事项

开启 SSL 加密会轻微影响性能。

9.2.2 参数介绍

参数	默认值	说明
`cert_path`	-	PEM 证书的路径。
`key_path`	-	密钥证书的路径。
`password_path`	-	密码文件证书的路径。
`ca_path`	-	受信任 CA 证书文件的路径。
`enable_ssl`	`false`	是否开启 SSL 加密。
`enable_graph_ssl`	`false`	是否仅在 Graph 服务上开启 SSL 加密。
`enable_meta_ssl`	`false`	是否仅在 Meta 服务上开启 SSL 加密。

9.2.3 证书模式

为了使用 SSL 加密，必须有 SSL 证书。悦数图数据库支持两种证书模式：

- 自签名证书模式

  需要自行制作签名证书。需要根据加密策略，在对应的配置文件内设置 `cert_path` 、 `key_path` 和 `password_path` 。

- CA 签名证书模式

  需要在认证机构（Certificate Authority）申请签名证书。需要根据加密策略，在对应的配置文件内设置 `cert_path` 、 `key_path` 和 `ca_path` 。

9.2.4 加密策略

悦数图数据库支持三种加密策略。加密涉及的具体进程请参见详细说明。

- 对客户端、Graph 服务、Meta 服务和 Storage 服务之间的传输数据加密。

  需要修改 `nebula-graphd.conf` 、 `nebula-metad.conf` 和 `nebula-storaged.conf` 配置文件，设置 `enable_ssl = true` 。

- 对客户端和 Graph 服务之间的传输数据加密。

  适用于集群设置在同一个机房内，仅对外开放 Graph 服务的端口。因为其他服务可以通过内部网络通信，无需加密。需要修改 `nebula-graphd.conf` 配置文件，设置 `enable_graph_ssl = true` 。

- 对集群中 Meta 服务相关的传输数据加密。

  适用于向 meta 服务传输需保密的信息。需要修改 `nebula-graphd.conf` 、 `nebula-metad.conf` 和 `nebula-storaged.conf` 配置文件，设置 `enable_meta_ssl = true` 。

Copyright © Yueshu Graph Technology - All rights reserved.

9.2.5 使用方式

1. 确认证书模式和加密策略。
2. 在对应的配置文件内增加证书配置和策略配置。

   例如使用自签名证书，并对客户端、Graph 服务、Meta 服务和 Storage 服务之间的数据传输进行加密。需要对三个配置文件都进行如下设置:

```
--cert_path=xxxxxx
--key_path=xxxxx
--password_path=xxxxxx
--enable_ssl=true
```

3. 客户端设置安全套接字并添加受信任的 CA。示例代码请参见 nebula-test-run.py。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

10. 备份与恢复

10.1 悦数备份&恢复

10.1.1 什么是Backup&Restore

Backup&Restore企业版是一款命令行界面（CLI）工具。通过Backup&Restore，用户可以备份悦数图数据库数据，并基于备份目录进行数据恢复。对于在 K8s Operator 中 BR 的部署，参见 K8s 中的 br。

> **Enterpriseonly**
>
> 该工具仅适用于悦数图数据库企业版。

功能

- 一键操作备份和恢复数据。
- 支持全量和增量备份。
- 支持全量恢复。
- 支持跨集群恢复。
- 支持基于以下存储环境备份和恢复数据:
- 本地磁盘（SSD 或 HDD），建议搭配共享存储服务（NFS）。
- 兼容亚马逊对象存储（Amazon S3）云存储服务接口，例如：阿里云对象存储（Alibaba Cloud OSS）、MinIO、Ceph RGW 等。
- 支持查看备份或恢复的进度。

限制

- 悦数图数据库的版本需要为 v3.5.1及以上。
- 不支持在容器部署的悦数图数据库集群中使用Backup&Restore。

使用流程

可以按照以下步骤使用备份和恢复功能:

1. 安装
2. 备份数据
3. 恢复数据

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

10.1.2 安装Backup&Restore

备份和恢复悦数图数据库数据需要安装Backup&Restore。本文介绍如何安装Backup&Restore。

安装说明

使用Backup&Restore时，需要安装 NebulaGraph Agent 插件。Agent 是集群中每台机器的一个守护进程，用于启停悦数图数据库服务和上传/下载备份文件。Backup&Restore和 Agent 插件的安装方式如下文。

版本兼容性

悦数图数据库	**Backup&Restore**	**Agent**
3.5.x	3.5.1	3.4.0
3.4.1	3.4.1、3.4.0	3.4.0

安装**Backup&Restore**

Backup&Restore是一款命令行界面（CLI）工具，可以帮助备份悦数图数据库的图空间数据，或者通过备份目录恢复数据。

操作步骤如下：

1. 获取安装包。

Enterpriseonly

联系我们以获取Backup&Restore的安装包。

2. 使用 `sudo rpm -i <rpm>` 命令安装 RPM 包。

例如，安装Backup&Restore需要运行以下命令，默认安装路径为 `/usr/local/br-ent/`：

`sudo rpm -i nebula-br-ent-<version>.x86_64.rpm`

在Backup&Restore的安装目录下，用户可以通过命令 `./br version` 查看版本信息。返回以下信息：

```
[br-ent]$ ./br version
Nebula Backup And Restore Utility Tool,V-3.5.1
```

安装 **Agent**

NebulaGraph Agent 以二进制形式存在各个机器的安装目录中，通过 RPC 协议对Backup&Restore提供服务。

Copyright © Yueshu Graph Technology - All rights reserved.

在每台机器中，进行以下操作：

1. 下载 Agent。

```
wget https://github.com/vesoft-inc/nebula-agent/releases/download/v3.4.0/agent-3.4.0-linux-amd64
```

2. 修改 Agent 的名称为 `agent`。

```
sudo mv agent-3.4.0-linux-amd64 agent
```

3. 授予 Agent 可执行权限。

```
sudo chmod +x agent
```

4. 执行以下命令启动 Agent。

> **Note**
>
> 启动 Agent 前，确保已经启动集群中的 Meta 服务，并且 Agent 有对应悦数图数据库服务目录和备份目录的读写权限。

```
sudo nohup ./agent --agent="<agent_node_ip>:8888" --meta="<metad_node_ip>:9559" > nebula_agent.log 2>&1 &
```

- `--agent`： Agent 所在机器的 IP 地址和访问端口。
- `--meta`： 集群中任一 Meta 服务所在机器的 IP 地址和访问端口。
- `--ratelimit`： 可选项，限制文件上传和下载的速度，防止带宽被占满导致其他服务不可用。单位：Bytes。

例如：

```
sudo nohup ./agent --agent="192.168.8.129:8888" --meta="192.168.8.129:9559" --ratelimit=1048576 > nebula_agent.log 2>&1 &
```

5. 连接服务并查看 Agent 的运行状态。

```
nebula> SHOW HOSTS AGENT;
+----------------+------+----------+---------+--------------+---------+
| Host           | Port | Status   | Role    | Git Info Sha | Version |
+----------------+------+----------+---------+--------------+---------+
| "192.168.8.129" | 8888 | "ONLINE" | "AGENT" | "96646b8"    |         |
+----------------+------+----------+---------+--------------+---------+
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

10.1.3 备份数据

用户可以通过Backup&Restore工具备份悦数图数据库数据。支持增量和全量备份，支持备份数据至本地及兼容亚马逊对象存储（Amazon S3）服务接口的云存储服务中。本文介绍如何备份数据。

背景信息

- 全量备份是指将数据库中的所有数据进行备份。
- 增量备份是指备份上一次备份后变化或者修改过的数据。上一次备份可以是全量备份或者增量备份。
- 关于悦数图数据库的数据目录结构，参见（默认）路径 `usr/local/nebula-ent/data` 。

注意事项

- 数据备份过程中，所有图空间中的 DDL 和 DML 语句将会阻塞，我们建议在业务低峰期进行操作，例如凌晨 2 点至 5 点。
- 如果备份数据到本地磁盘，备份的文件将会放置在每个服务器的本地路径中。也可以在本地挂载 NFS 文件系统，以便将备份数据还原到不同的主机上。
- 全量备份时，如果数据库中的数据量较大，可能会耗费较长时间。
- 执行增量备份的集群和指定的上一次备份的集群需为同一个，并且和指定的上一次备份的（存储桶）路径需相同。
- 确保每次增量备份距离上一次备份的时间小于一个 `wal_ttl` 的时间。
- 确保 Agent 有对应悦数图数据库安装目录和备份目录的读写权限。
- 不支持备份 Listener。
- 不支持备份全文索引。
- 不支持备份指定图空间数据。

准备工作

- 已启动悦数图数据库服务。
- 已安装 Backup&Restore工具和 Agent 插件并在集群中的每个机器上运行 Agent。
- 如果在本地保存备份目录，需要在 Meta 服务器、Storage 服务器和Backup&Restore机器上创建绝对路径相同的目录，并记录绝对路径，同时需要保证账号对该目录有写权限。

全量备份

全量备份数据至云存储服务

> **Note**
>
> 只支持备份数据至兼容亚马逊 S3 协议的云存储服务中。

在Backup&Restore工具安装目录下执行以下命令备份整个集群数据至云存储服务：

```
./br backup full --meta <ip_address:port> --s3.access_key <access_key> --s3.secret_key <secret_key>  --s3.region <region_name> --storage s3://<storage_path> --s3.endpoint <endpoint_url>
```

例如，对 Meta 服务的地址为 `192.168.8.129:9559` 的整个集群进行全量备份操作，并将备份目录保存至 Amazon S3 服务 `nebula-br-test` 存储桶的 `/` 路径下。

```
./br backup full --meta 192.168.8.129:9559 --s3.access_key QImbbGDjfQExxx --s3.secret_key dVSJZfl7tnoFq7Z5zt6sfxxxx  --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint
http://192.168.8.xxx:9000/
```


Copyright © Yueshu Graph Technology - All rights reserved.

全量备份数据至本地

Caution

在生产环境中，建议用户将 NFS（Network File System）共享存储设备挂载到 Meta 服务器、Storage 服务器和Backup&Restore机器上进行本地备份，或者使用 Alibaba Cloud OSS、Amazon S3 进行远程备份。否则当需要通过本地文件恢复数据前，必须手动将这些备份目录移动到指定目录。

备份至本地时，只备份 leader metad 和 leader partition 的数据，因此当未使用共享存储且有多个 metad 或 partition 的 replica 数大于 1 时，需要手动将备份后的 leader metad 的目录（路径为 `<storage_path>/meta`）拷贝并覆盖其他 follower metad 节点的对应目录，并将 leader partition 的目录中对应的分片数据（路径为 `<storage_path>/<partition_id>`）拷贝至其他 follower partition 对应的目录中。不建议用户进行手动拷贝操作。

在Backup&Restore工具安装目录下执行以下命令备份整个集群数据至本地：

Note

确保备份目录的路径存在。

```
./br backup full --meta <ip_address:port> --storage local://<storage_path>
```

例如：对 Meta 服务的地址为 `192.168.8.129:9559` 的整个集群进行全量备份操作，并将备份目录保存到本地的 `/backup/` 路径下。

```
./br backup full --meta "192.168.8.129:9559" --storage "local:///backup/"
```

目录结构

全量备份会备份全量的 leader metad 和 leader partition 的数据。备份目录中，结构如下：

```
├── BACKUP_2022_08_12_08_41_19.meta（备份元信息）
├── data（storage 数据的备份目录）
│   ├── 1（图空间 ID）
│   │   ├── 1（分片 ID）
│   │   │   ├── data
│   │   │   │   ├── 000033.sst
│   │   │   │   ├── 000035.sst
│   │   │   │   ├── 000037.sst
│   │   │   │   ├── 000039.sst
│   │   │   │   ├── CURRENT
│   │   │   │   ├── MANIFEST-000004
│   │   │   │   └── OPTIONS-000007
│   │   │   └── wal
│   │   │       ├── 0000000000000000001.wal
│   │   │       ├── 0000000000000000267.wal
│   │   │       ├── 0000000000000000324.wal
│   │   │       └── commitlog.id
│   │   ├── 10（分片 ID）
│       ...
└── meta（meta 数据的备份目录）
    ...
```

增量备份

Caution

进行增量备份前，确保上一次的全量备份或者增量备份目录中的数据无修改，否则可能造成增量备份失败。

增量备份数据至云存储服务

Note

只支持备份数据至兼容亚马逊 S3 协议的云存储服务中。

在Backup&Restore工具安装目录下执行以下命令备份增量数据至云存储服务：

Copyright © Yueshu Graph Technology - All rights reserved.

```
./br backup incr --meta <ip_address:port> --s3.access_key <access_key> --s3.secret_key <secret_key>  --s3.region <region_name> --storage s3://<storage_path> --s3.endpoint <endpoint_url> --
base <backup_file_name>
```

例如，对 Meta 服务的地址为 `192.168.8.129:9559` 的整个集群基于备份目录 `BACKUP_2022_08_11_09_11_07` 进行增量备份操作，并将备份目录保存至 Amazon S3 服务 `nebula-br-test` 存储桶的 `/` 路径下。

```
./br backup incr --meta 192.168.8.129:9559 --s3.access_key QImbbGDjfQExxx --s3.secret_key dVSJZfl7tnoFq7Z5zt6sfxxxx  --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint
http://192.168.8.xxx:9000/ --base BACKUP_2022_08_11_09_11_07
```

增量备份数据至本地

在Backup&Restore工具安装目录下执行以下命令备份增量数据至本地：

Note

确保备份目录的路径存在。

```
./br backup incr --meta <ip_address:port> --storage local://<storage_path> --base <backup_file_name>
```

例如：对 Meta 服务的地址为 `192.168.8.129:9559` 的整个集群基于备份目录 `BACKUP_2022_08_11_09_11_07` 进行增量备份操作，并将备份目录保存到本地的 `/backup/` 路径下。

```
./br backup incr --meta "192.168.8.129:9559" --storage "local:///backup/" --base BACKUP_2022_08_11_09_11_07
```

目录结构

增量备份除了备份 leader meta 外，对于已有的图空间（以下代码中的图空间 ID `1`）的 leader partition 数据，只备份 `wal` 目录；而对新增加的图空间（以下代码中的图空间 ID `4`）中 leader partition 数据, 备份其全量的 `data` 和 `wal` 目录数据。因此相较全量备份的数据结构，增量备份目录结构可能会有所不同。

增量备份目录结构示例如下：

```
├── BACKUP_2022_08_12_08_58_23.meta
├── data (storage 数据的备份目录)
│   ├── 1 (图空间 ID)
│   │   ├── 1 (分片 ID)
│   │   │   └── wal  (wal 目录)
│   │   │       ├── 0000000000000000671.wal
│   │   │       ├── 0000000000000000700.wal
│   │   │       └── commitlog.id
...
│   └── 4 (图空间 ID)
│       ├── 1
│       │   ├── data (data 目录)
│       │   │   ├── 000009.sst
│       │   │   ├── CURRENT
│       │   │   ├── MANIFEST-000004
│       │   │   └── OPTIONS-000007
│       │   └── wal
│       │       ├── 00000000000000000001.wal
│       │       └── commitlog.id
└── meta (meta 数据的备份目录)
    ...
```


Copyright © Yueshu Graph Technology - All rights reserved.

参数说明

参数	数据类型	是否必需	默认值	说明
`-h`，`--help`	-	否	无	查看帮助。
`--debug`	-	否	无	查看更多日志信息。
`--log`	string	否	`"br.log"`	日志路径。
`--concurrency`	int	否	无	用来控制数据备份时，上传文件的并发数量。默认值 `5` 。
`--meta`	string	是	无	集群中任一 Meta 服务的地址和端口号。
`--storage`	string	是	无	备份数据存储位置，格式为：`<schema>://<storage_path>` 。 `schema`： 可选值 `s3` 和 `local` 。 选择 `s3` 时，需要填写 `s3.access_key` 、 `s3.endpoint` 、 `s3.region` 和 `s3.secret_key` 。 `<storage_path>`： 存储位置的路径。
`--s3.access_key`	string	否	无	用于标识用户。
`--s3.endpoint`	string	否	无	S3 对外服务的访问域名的 URL，指定 http 或 https。
`--s3.region`	string	否	无	数据中心所在物理位置。
`--s3.secret_key`	string	否	无	用户用于加密签名字符串和用来验证签名字符串的密钥，必须保密。
`--base`	string	是	无	之前任一备份的目录名。基于此备份，进行增量备份。

查看备份进度

在安装目录内可以查看Backup&Restore的日志文件 `br.log` 。日志文件会记录备份进度，内容类似：

```
{"level":"info","msg":"full upload storaged partition finished, progress: 1/20","time":"2023-03-15T02:13:20.946Z"}
{"level":"info","msg":"full upload storaged partition finished, progress: 2/20","time":"2023-03-15T02:13:21.154Z"}
{"level":"info","msg":"full upload storaged partition finished, progress: 3/20","time":"2023-03-15T02:13:21.537Z"}
```

下一步

备份目录生成后，可以将备份目录的数据恢复到悦数图数据库中。具体操作，参见恢复数据。

Caution

请勿修改备份目录的名称及路径，否则会导致数据恢复失败。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

10.1.4 恢复数据

如果使用备份工具备份了悦数图数据库的数据，可以通过备份目录进行数据恢复。本文介绍如何通过备份目录恢复数据。

注意事项

- 数据恢复会有一段时间的服务不可用，建议在业务低峰期进行操作。
- 数据恢复需要原集群和目标集群的 Storage 主机数量一致。
- 数据恢复执行成功后，目标集群上已有的数据会被删除，然后替换为备份目录中的数据。建议提前备份目标集群上的数据。

准备工作

- 已安装 Backup&Restore工具和 Agent 插件并在集群中的每个机器上运行 Agent。
- 确认没有应用程序连接到待恢复数据的悦数图数据库集群。
- 确认原集群和目标集群的 Storage 主机数量一致。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

在Backup&Restore工具安装目录下执行以下操作:

Copyright © Yueshu Graph Technology - All rights reserved.

1. 查看现有备份信息。

- 列出在本地的备份目录。

```
./br show --storage local://<storage_path>
```

例如，可以使用以下命令列出在本地 `/backup/` 路径中的备份目录的信息。

```
./br show --storage "local:///backup/"
+---------------------------+---------------------+-----------------------+-------------+------------+---------------------------+
|           NAME            |     CREATE TIME     |         SPACES        | FULL BACKUP | ALL SPACES |      BASE BACKUP NAME     |
+---------------------------+---------------------+-----------------------+-------------+------------+---------------------------+
| BACKUP_2022_08_11_06_12_43 | 2022-08-11 06:12:43 | basketballplayer      | true        | true       |                           |
| BACKUP_2022_08_11_08_27_14 | 2022-08-11 08:27:15 | basketballplayer,br   | false       | true       | BACKUP_2022_08_11_06_12_43 |
```

- 列出在兼容 S3 协议的对象存储服务 `nebula-br-test` 桶下的 `/` 中的备份的信息。

```
./br show --s3.access_key QImbbGDjfQEYxxxx --s3.secret_key dVSJZfl7tnoFq7Z5zt6sfYnvi63bxxxx  --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint http://192.168.8.xxx:9000/
```

2. 使用以下命令恢复数据。

- 基于本地备份恢复数据。

```
./br restore full --meta <ip_address> --storage <storage_path> --name <backup_name>
```

例如，可以使用以下命令，将本地 `/backup/` 路径中的 `BACKUP_2022_08_11_09_11_07` 文件数据恢复到为 Meta 地址为 `192.168.8.129:9559` 集群中：

```
./br restore full --meta "192.168.8.129:9559" --storage "local:///backup/" --name BACKUP_2022_08_11_09_11_07
```

- 基于云存储备份恢复数据。

将兼容 S3 协议的对象存储服务 `nebula-br-test` 桶下的 `/` 路径中的 `BACKUP_2022_08_12_07_37_02` 备份数据，恢复到 Meta 服务的地址为 `192.168.8.129:9559` 的集群中。

```
./br restore full --meta 192.168.8.129:9559 --s3.accesskey QImbbGDjfQEYxxxx --s3.secretkey dVSJZfl7tnoFq7Z5zt6sfYnvi63bxxxx  --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint
http://192.168.8.xxx:9000/ --log "3.log" --name BACKUP_2022_08_12_07_37_02
```

如果返回以下信息，表示数据已经恢复成功。

```
Restore succeed.
```

Note

如果数据恢复失败，Backup&Restore会自动执行回滚流程，将集群的数据会自动还原回恢复前的数据。

3. 执行以下命令清理临时文件。该命令将清理集群和外部存储中的临时文件，同时也可以使用该命令清理旧的备份目录。示例如下：

Note

默认情况下，数据恢复执行发生错误时，Backup&Restore会自动清理临时文件。如果自动清理失败，需要手动执行命令清理临时文件。

- 清除本地的备份目录。

```
./br cleanup --meta 192.168.8.129:9559 --storage "local:///backup/" --name BACKUP_2022_08_11_09_11_07
```

- 清除云存储服务中的备份目录。

```
./br cleanup --meta 192.168.8.129:9559 --s3.accesskey QImbbGDjfQEYxxxx --s3.secretkey dVSJZfl7tnoFq7Z5zt6sfYnvi63bxxxx  --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint
http://192.168.8.xxx:9000/ --name BACKUP_2022_08_12_07_37_02
```


Copyright © Yueshu Graph Technology - All rights reserved.

参数说明

以下列出备份命令的相关参数。

参数	数据类型	是否必需	默认值	说明
`-h,-help`	-	否	-	查看帮助。
`--debug`	-	否	无	查看更多日志信息。
`--log`	string	否	"br.log"	日志路径。
`--concurrency`	int	否	无	用来控制数据恢复时，下载文件的并发数量。默认值 `5` 。
`--meta`	string	是	无	meta 服务的地址和端口号。
`--name`	string	是	无	备份名字。
`--storage`	string	是	无	备份数据存储位置，格式为：`<schema>://<storage_path>` 。 `schema`： 可选值 `s3` 和 `local` 。 选择 `s3` 时，需要填写 `s3.access_key` 、 `s3.endpoint` 、 `s3.region` 和 `s3.secret_key` 。 `<storage_path>`： 备份文件的存储路径。
`--s3.access_key`	string	否	无	用于标识用户。
`--s3.endpoint`	string	否	无	S3 对外服务的访问域名的 URL，指定 http 或 https。
`--s3.region`	string	否	无	数据中心所在物理位置。
`--s3.secret_key`	string	否	无	用户用于加密签名字符串和用来验证签名字符串的密钥，必须保密。

查看恢复进度

在安装目录内可以查看Backup&Restore的日志文件 `br.log` 。日志文件会记录恢复进度，内容类似：

```
{"level":"info","msg":"download storaged partition finished, progress: 1/20","time":"2023-03-15T02:16:43.430Z"}
{"level":"info","msg":"download storaged partition finished, progress: 2/20","time":"2023-03-15T02:16:43.431Z"}
{"level":"info","msg":"download storaged partition finished, progress: 3/20","time":"2023-03-15T02:16:43.763Z"}
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

10.2 管理快照

悦数图数据库提供快照（snapshot）功能，用于保存集群当前时间点的数据状态，当出现数据丢失或误操作时，可以通过快照恢复数据。

10.2.1 前提条件

悦数图数据库的身份认证功能默认是关闭的，此时任何用户都能使用快照功能。

如果身份认证开启，仅 God 角色用户可以使用快照功能。关于角色说明，请参见内置角色权限。

10.2.2 注意事项

- 系统结构发生变化后，建议立刻创建快照，例如在 `add host` 、 `drop host` 、 `create space` 、 `drop space` 、 `balance` 等操作之后。
- 不支持自动回收创建失败的快照垃圾文件，需要手动删除。
- 不支持指定快照保存路径，默认路径为 `/usr/local/nebula/data` 。

10.2.3 快照路径

悦数图数据库创建的快照以目录的形式存储，例如 `SNAPSHOT_2021_03_09_08_43_12` ，后缀 `2021_03_09_08_43_12` 根据创建时间（UTC）自动生成。

创建快照时，快照目录会自动在 leader Meta 服务器和所有 Storage 服务器的目录 `checkpoints` 内创建。

为了快速定位快照所在路径，可以使用 Linux 命令 `find` 。例如：

```
$ find |grep 'SNAPSHOT_2021_03_09_08_43_12'
./data/meta2/nebula/0/checkpoints/SNAPSHOT_2021_03_09_08_43_12
./data/meta2/nebula/0/checkpoints/SNAPSHOT_2021_03_09_08_43_12/data
./data/meta2/nebula/0/checkpoints/SNAPSHOT_2021_03_09_08_43_12/data/000081.sst
...
```

10.2.4 创建快照

命令 `CREATE SNAPSHOT` 可以创建集群当前时间点的快照。只支持创建所有图空间的快照，不支持创建指定图空间的快照。

> **Note**
>
> 如果快照创建失败，请删除快照重新创建。

```
nebula> CREATE SNAPSHOT;
```

10.2.5 查看快照

命令 `SHOW SNAPSHOTS` 可以查看集群中的所有快照。

```
nebula> SHOW SNAPSHOTS;
+-------------------------------+---------+-----------------+
| Name                          | Status  | Hosts           |
+-------------------------------+---------+-----------------+
| "SNAPSHOT_2021_03_09_08_43_12" | "VALID" | "127.0.0.1:9779" |
| "SNAPSHOT_2021_03_09_09_10_52" | "VALID" | "127.0.0.1:9779" |
+-------------------------------+---------+-----------------+
```


Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下：

参数	说明
`Name`	快照名称，前缀为 `SNAPSHOT`，表示该文件为快照文件，后缀为快照创建的时间点（UTC 时间）。
`Status`	快照状态。`VALID` 表示快照有效，`INVALID` 表示快照无效。
`Hosts`	创建快照时所有 Storage 服务器的 IP 地址和端口。

10.2.6 删除快照

命令 `DROP SNAPSHOT` 可以删除指定的快照，语法为：

```
DROP SNAPSHOT <snapshot_name>;
```

示例如下：

```
nebula> DROP SNAPSHOT SNAPSHOT_2021_03_09_08_43_12;
nebula> SHOW SNAPSHOTS;
+-------------------------------+---------+------------------+
| Name                          | Status  | Hosts            |
+-------------------------------+---------+------------------+
| "SNAPSHOT_2021_03_09_09_10_52" | "VALID" | "127.0.0.1:9779" |
+-------------------------------+---------+------------------+
```

10.2.7 恢复快照

Warning

恢复快照数据时，如果快照中备份的图空间被删除（执行了 `DROP SPACE` 操作），那么该图空间数据将无法恢复。

当前未提供恢复快照命令，需要手动拷贝快照文件到对应的文件夹内，也可以通过 shell 脚本进行操作。实现逻辑如下：

1. 创建快照后，会在 leader Meta 服务器和所有 Storage 服务器的安装目录内生成 `checkpoints` 目录，保存创建的快照。以本文为例，当存在 2 个图空间时，创建的快照分别保存在 `/usr/local/nebula/data/meta/nebula/0/checkpoints`、`/usr/local/nebula/data/storage/nebula/3/checkpoints` 和 `/usr/local/nebula/data/storage/nebula/4/checkpoints` 中。

```
$ ls /usr/local/nebula/data/meta/nebula/0/checkpoints/
SNAPSHOT_2021_03_09_09_10_52
$ ls /usr/local/nebula/data/storage/nebula/3/checkpoints/
SNAPSHOT_2021_03_09_09_10_52
$ ls /usr/local/nebula/data/storage/nebula/4/checkpoints/
SNAPSHOT_2021_03_09_09_10_52
```

2. 当数据丢失需要通过快照恢复时，用户可以找到合适的时间点快照，将内部的文件夹 `data` 和 `wal` 分别拷贝到各自的上级目录（和 `checkpoints` 平级），覆盖之前的 `data` 和 `wal`，然后重启集群即可。

Caution

需要同时覆盖所有 Meta 节点的 data 和 wal 目录，因为存在重启集群后发生 Meta 重新选举 leader 的情况，如果不覆盖所有 Meta 节点，新的 leader 使用的还是最新的 Meta 数据，导致恢复失败。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

11. 同步与迁移

11.1 负载均衡

我们可以提交任务让悦数图数据库的 Storage 服务实现负载均衡。详细示例请参见 Storage 负载均衡。

Note

其他作业管理命令请参见作业管理。

负载均衡相关的语法说明如下。

语法	说明
`SUBMIT JOB BALANCE LEADER`	启动任务均衡分布所有图空间中的 leader。该命令会返回任务 ID（`job_id`）。
`SUBMIT JOB BALANCE DATA`	启动任务均衡分布当前图空间中的所有分片。该命令会返回任务 ID（`job_id`）。
`SUBMIT JOB BALANCE DATA REMOVE <ip:port> [,<ip>:<port> ...]`	启动任务迁空当前图空间指定的 Storage 服务中的分片。端口默认为 `9779`。

Note

`REMOVE` 只能清空当前图空间的分片，如果 Storage 节点的图空间较多，需要切换到不同图空间执行 `REMOVE` 操作。

查看、停止、重启任务，请参见作业管理。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

11.2 集群间数据同步

悦数图数据库支持在集群间进行数据同步，即主集群 A 的数据可以近实时地复制到从集群 B 中，方便用户进行异地灾备或分流，降低数据丢失的风险，保证数据安全。

11.2.1 背景

在集群间数据同步方案中，如果主集群 A 的图空间 a 和从集群 B 的图空间 b 建立了同步关系，任何向图空间 a 写入的数据，都会被发送到 Meta listener 或 Storage listener。listener 再将数据发送到 drainer。drainer 接收并存储数据，然后通过从集群的 Meta client 或 Storage client 发送数据至从集群的对应分片。

通过以上流程，最终实现集群间数据同步。

11.2.2 适用场景

- 异地灾备：通过数据同步可以实现跨机房或者跨城市的异地灾备。
- 数据迁移：通过切换主从集群的身份，可以实现不停止服务而完成迁移。
- 读写分离：通过设置主集群只写，从集群只读，实现读写分离，降低集群负载，提高稳定性和可用性。

Copyright © Yueshu Graph Technology - All rights reserved.

11.2.3 注意事项

- 主从集群的数据库版本必须相同，否则数据同步会失败。
- 数据同步的基本单位是图空间，即只可以设置从一个图空间到另一个图空间的数据同步。
- 主从集群的数据同步是异步的（近实时）。
- 主从集群之间只支持1 对 1，不支持多个主集群同步到 1 个从集群，也不支持 1 个主集群同步到多个从集群，但是可以通过链式方式同步到多个从集群，例如 `主集群->从集群1->从集群2->从集群3` 。
- Meta listener 监听 Meta 服务，Storage listener 监听 Storage 服务，不可以混用。
- 1 个图空间只有 1 个 Meta listener 和 1 个或多个 Storage listener，这些 listener 可以对应 1 个或多个 drainer。
- listener 服务记录来自主集群的 WAL 或快照，drainer 服务记录来自 listener 的 WAL 和写入从集群的 WAL。这些文件都保存在对应服务的本地。
- 从集群中数据如果不为空，数据同步时可能会导致数据冲突或者数据不一致。建议保持从集群数据为空。
- 建议使用具备 God 权限的 `root` 用户进行集群数据同步操作。集群同步操作中各命令需要的用户角色权限不同，详情参见文末的权限说明。
- 在数据同步期间，请勿同时在主集群进行数据恢复（备份恢复和快照恢复）操作，否则数据同步将失败。

11.2.4 操作步骤

准备工作

- 准备至少 2 台部署服务的机器。主从集群需要分开部署，listener 和 drainer 可以单独部署，也可以分别部署在主从集群所在机器上，但是会增加集群负载。

示例环境

主集群A：机器 IP 地址为 `192.168.10.101` ，只启动 Graph、Meta、Storage 服务。

从集群B：机器 IP 地址为 `192.168.10.102` ，只启动 Graph、Meta、Storage 服务。

listener：机器 IP 地址为 `192.168.10.103` ，只启动 Meta listener、Storage listener 服务。

drainer：机器 IP 地址为 `192.168.10.104` ，只启动 drainer 服务。

1.搭建主从集群、**listener** 和 **drainer** 服务

1. 在所有机器上安装悦数图数据库，修改配置文件：

- 主、从集群修改： `nebula-graphd.conf` 、 `nebula-metad.conf` 、 `nebula-storaged.conf` 。
- listener 修改： `nebula-metad-listener.conf` 、 `nebula-storaged-listener.conf` 。
- drainer 修改： `nebula-drainerd.conf` 。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

修改配置文件时需要注意：

- 将配置文件的后缀 `.default` 或 `.production` 删除。
- Meta 服务的配置文件（`nebula-metad.conf`）中，设置 `license_manager_url` 的值为许可证管理工具所在的主机 IP 和端口号 `9119`，例如 `192.168.8.xxx:9119`。
- 所有配置文件里都需要用真实的机器 IP 地址替换 `local_ip` 的 `127.0.0.1`。
- 所有 `nebula-graphd.conf` 配置文件里设置 `enable_authorize=true`。
- 在主集群的 `nebula-metad.conf` 和 `nebula-storaged.conf` 文件中，配置 `--snapshot_send_files=false`。
- 主从集群填写各自集群的 `meta_server_addrs`，注意不要错填其他集群的地址。
- listener 的配置文件里 `meta_server_addrs` 填写主集群的机器 IP，`meta_sync_listener` 填写 listener 机器的 IP。
- drainer 的配置文件里 `meta_server_addrs` 填写从集群的机器 IP。

更多配置说明，请参见配置管理。

2. 在所有机器的悦数图数据库安装目录内启动对应的服务：

- 主、从集群启动命令：`sudo scripts/nebula.service start all`。
- listener 启动命令：
- Meta listener：`sudo bin/nebula-metad --flagfile etc/nebula-metad-listener.conf`。
- Storage listener：`sudo bin/nebula-storaged --flagfile etc/nebula-storaged-listener.conf`。
- drainer 启动命令：`sudo scripts/nebula-drainerd.service start`。

3. 登录主集群增加 Storage 主机，检查 listener 服务状态。

```
nebula> ADD HOSTS 192.168.10.101:9779;
nebula> SHOW HOSTS STORAGE;
+-----------------+------+----------+-----------+-------------+---------------------+
| Host            | Port | Status   | Role      | Git Info Sha | Version             |
+-----------------+------+----------+-----------+-------------+---------------------+
| "192.168.10.101" | 9779 | "ONLINE" | "STORAGE" | "xxxxxxx"    | "3.5.1"          |
+-----------------+------+----------+-----------+-------------+---------------------+

nebula> SHOW HOSTS STORAGE LISTENER;
+-----------------+------+----------+-------------------+-------------+---------------------+
| Host            | Port | Status   | Role              | Git Info Sha | Version             |
+-----------------+------+----------+-------------------+-------------+---------------------+
| "192.168.10.103" | 9789 | "ONLINE" | "STORAGE_LISTENER" | "xxxxxxx"    | "3.5.1"          |
+-----------------+------+----------+-------------------+-------------+---------------------+

nebula> SHOW HOSTS META LISTENER;
+-----------------+------+----------+----------------+-------------+---------------------+
| Host            | Port | Status   | Role           | Git Info Sha | Version             |
+-----------------+------+----------+----------------+-------------+---------------------+
| "192.168.10.103" | 9569 | "ONLINE" | "META_LISTENER" | "xxxxxxx"    |  "3.5.1"         |
+-----------------+------+----------+----------------+-------------+---------------------+
```

4. 登录从集群增加 Storage 主机，检查 drainer 服务状态。

```
nebula> ADD HOSTS 192.168.10.102:9779;
nebula> SHOW HOSTS STORAGE;
+-----------------+------+----------+-----------+-------------+---------------------+
| Host            | Port | Status   | Role      | Git Info Sha | Version             |
+-----------------+------+----------+-----------+-------------+---------------------+
| "192.168.10.102" | 9779 | "ONLINE" | "STORAGE" | "xxxxxxx"    | "3.5.1"          |
+-----------------+------+----------+-----------+-------------+---------------------+

nebula> SHOW HOSTS DRAINER;
+-----------------+------+----------+-----------+-------------+---------------------+
| Host            | Port | Status   | Role      | Git Info Sha | Version             |
+-----------------+------+----------+-----------+-------------+---------------------+
| "192.168.10.104" | 9889 | "ONLINE" | "DRAINER" | "xxxxxxx"    | "3.5.1"          |
+-----------------+------+----------+-----------+-------------+---------------------+
```

2.设置服务

1. 登录主集群，创建图空间 `basketballplayer`。

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> CREATE SPACE basketballplayer(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
```

2. 进入图空间 `basketballplayer`，注册 drainer 服务。

```
nebula> USE basketballplayer;
//注册 drainer 服务。
nebula> SIGN IN DRAINER SERVICE(192.168.10.104:9889);
//检查是否注册成功。
nebula> SHOW DRAINER CLIENTS;
+-----------+------------------+------+
| Type      | Host             | Port |
+-----------+------------------+------+
| "DRAINER" | "192.168.10.104" | 9889 |
+-----------+------------------+------+
```

Note

注册多个 drainer 服务的命令示例：`SIGN IN DRAINER SERVICE(192.168.8.x:9889),(192.168.8.x:9889)`

3. 设置 listener 服务。

```
//设置 listener 服务，待同步的图空间名称为 replication_basketballplayer（下文将在从集群中创建）。
nebula> ADD LISTENER SYNC META 192.168.10.103:9569 STORAGE 192.168.10.103:9789 TO SPACE replication_basketballplayer;
//查看 listener 状态。
nebula> SHOW LISTENER SYNC;
+--------+--------+------------------------+-------------------------------+----------+
| PartId | Type   | Host                   | SpaceName                     | Status   |
+--------+--------+------------------------+-------------------------------+----------+
| 0      | "SYNC" | ""192.168.10.103":9569" | "replication_basketballplayer" | "ONLINE" |
| 1      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 2      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 3      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 4      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 5      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 6      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 7      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 8      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 9      | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 10     | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 11     | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 12     | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 13     | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 14     | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
| 15     | "SYNC" | ""192.168.10.103":9789" | "replication_basketballplayer" | "ONLINE" |
+--------+--------+------------------------+-------------------------------+----------+
```

Note

添加多个 Storage listener 服务的命令示例：`ADD LISTENER SYNC META 192.168.10.xxx:9569 STORAGE 192.168.10.xxx:9789,192.168.10.xxx:9789 TO SPACE replication_basketballplayer`

4. 登录从集群，创建图空间 `replication_basketballplayer`。

```
nebula> CREATE SPACE replication_basketballplayer(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
```

5. 进入图空间 `replication_basketballplayer`，设置 drainer 服务。

```
nebula> USE replication_basketballplayer;
//设置 drainer 服务。
nebula> ADD DRAINER 192.168.10.104:9889;
//查看 drainer 状态。
nebula> SHOW DRAINERS;
+-------------------------+----------+
| Host                    | Status   |
+-------------------------+----------+
| ""192.168.10.104":9889" | "ONLINE" |
+-------------------------+----------+
```

Note

添加多个 drainer 服务的命令示例：`ADD DRAINER 192.168.8.x:9889,192.168.8.x:9889`

6. 修改图空间 `replication_basketballplayer` 为只读。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

修改为只读是防止误操作导致数据不一致。只影响该图空间，其他图空间仍然可以读写。

```
//设置当前图空间为只读。
nebula> SET VARIABLES read_only=true;
//查看当前图空间的读写属性。
nebula> GET VARIABLES read_only;
+-------------+--------+-------+
| name        | type   | value |
+-------------+--------+-------+
| "read_only" | "bool" | true  |
+-------------+--------+-------+
```

3.验证数据

7. 登录主集群，创建 Schema，插入数据。

```
nebula> USE basketballplayer;
nebula> CREATE TAG player(name string, age int);
nebula> CREATE EDGE follow(degree int);
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);
nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);
```

8. 登录从集群，检查数据。

```
nebula> USE replication_basketballplayer;
nebula> SUBMIT JOB STATS;
nebula> SHOW STATS;
+---------+------------+-------+
| Type    | Name       | Count |
+---------+------------+-------+
| "Tag"   | "player"   | 2     |
| "Edge"  | "follow"   | 1     |
| "Space" | "vertices" | 2     |
| "Space" | "edges"    | 1     |
+---------+------------+-------+

nebula> FETCH PROP ON player "player100" YIELD properties(vertex);
+------------------------------+
| properties(VERTEX)           |
+------------------------------+
| {age: 42, name: "Tim Duncan"} |
+------------------------------+

nebula> GO FROM "player101" OVER follow YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player100" |
+-------------+
```

11.2.5 停止/重启数据同步

数据同步时，listener 会持续发送数据给 drainer。

如果需要停止数据同步，可以使用 `STOP SYNC` 命令。此时 listener 会停止向 drainer 同步数据。

如果需要重启数据同步，可以使用 `RESTART SYNC` 命令。此时 listener 会向 drainer 发送停止期间堆积的数据。如果 listener 上的 WAL 丢失，listener 会从主集群拉取快照重新进行同步。

11.2.6 查看集群间数据同步状态

用户向主集群中写入数据时，可以查看集群间数据同步的状态，以判断数据同步是否正常。

查看主集群同步数据的状态

在主集群中执行 `SHOW SYNC STATUS` 命令可查看主集群给从集群发送数据的状态。 `SHOW SYNC STATUS` 实时获取集群间数据同步状态的信息，只有当主集群写入成功了，才往从集群发送同步数据。

Copyright © Yueshu Graph Technology - All rights reserved.

示例如下：

```
// 在主集群中写入数据。
nebula> INSERT VERTEX player(name,age) VALUES "player102":("LaMarcus Aldridge", 33);
nebula> INSERT VERTEX player(name,age) VALUES "player102":("LaMarcus Aldridge", 33);
nebula> INSERT VERTEX player(name,age) VALUES "player103":("Rudy Gay", 32);
nebula> INSERT VERTEX player(name,age) VALUES "player104":("Marco Belinelli", 32);

// 查看当前集群数据同步的状态（返回结果表示正在发送数据给从集群中）。
nebula> SHOW SYNC STATUS;
+--------+-------------+-----------+--------------+
| PartId | Sync Status | LogId Lag | Time Latency |
+--------+-------------+-----------+--------------+
| 0      | "ONLINE"    | 0         | 0            |
| 1      | "ONLINE"    | 0         | 0            |
| 2      | "ONLINE"    | 0         | 0            |
| 3      | "ONLINE"    | 0         | 0            |
| 4      | "ONLINE"    | 0         | 0            |
| 5      | "ONLINE"    | 1         | 46242122     |
| 6      | "ONLINE"    | 0         | 0            |
| 7      | "ONLINE"    | 0         | 0            |
| 8      | "ONLINE"    | 0         | 0            |
| 9      | "ONLINE"    | 0         | 0            |
| 10     | "ONLINE"    | 0         | 0            |
| 11     | "ONLINE"    | 0         | 0            |
| 12     | "ONLINE"    | 0         | 0            |
| 13     | "ONLINE"    | 0         | 0            |
| 14     | "ONLINE"    | 0         | 0            |
| 15     | "ONLINE"    | 0         | 0            |
+--------+-------------+-----------+--------------+
// 再次查看当前集群数据同步的状态（返回结果表示数据已完全同步至从集群，没有需要待同步的数据）。
nebula> SHOW SYNC STATUS;
+--------+-------------+-----------+--------------+
| PartId | Sync Status | LogId Lag | Time Latency |
+--------+-------------+-----------+--------------+
| 0      | "ONLINE"    | 0         | 0            |
| 1      | "ONLINE"    | 0         | 0            |
| 2      | "ONLINE"    | 0         | 0            |
| 3      | "ONLINE"    | 0         | 0            |
| 4      | "ONLINE"    | 0         | 0            |
| 5      | "ONLINE"    | 0         | 0            |
| 6      | "ONLINE"    | 0         | 0            |
| 7      | "ONLINE"    | 0         | 0            |
| 8      | "ONLINE"    | 0         | 0            |
| 9      | "ONLINE"    | 0         | 0            |
| 10     | "ONLINE"    | 0         | 0            |
| 11     | "ONLINE"    | 0         | 0            |
| 12     | "ONLINE"    | 0         | 0            |
| 13     | "ONLINE"    | 0         | 0            |
| 14     | "ONLINE"    | 0         | 0            |
| 15     | "ONLINE"    | 0         | 0            |
+--------+-------------+-----------+--------------+
```

执行 `SHOW SYNC STATUS` 命令，返回结果中的参数说明如下：

参数	说明
PartId	主集群中图空间对应的分片 ID。当值为 `0` 时，表示图空间中 Meta listener 同步 Meta 数据所在的分片 ID。当为其他值时，表示相应图空间中 Storage listener 同步 Storage 数据所在的分片ID。
Sync Status	表示 listener 的状态。 当值为 `ONLINE` 时，listener 持续发送数据给 drainer。 当值为 `OFFLINE` 时，listener 停止发送数据给 drainer。
LogId Lag	表示的 Log ID 间隔，也就是主集群对应分片还有多少条 Log 往从集群发送。 当值为 `0` 时，表示主集群对应分片中没有 Log 需要发送。
Time Latency	主集群的对应分片中需要发送最后一条 Log 的 WAL 中的时间戳与已经发送的最后一条 Log 的 WAL 中的时间戳差值。 当值为 `0` 时，表示数据已经发送至从集群。 单位：毫秒（ms）。

查看从集群同步数据的状态

在从集群中，执行 `SHOW DRAINER SYNC STATUS` 查看从集群同步接收的数据至从集群 Meta 和 Storage 的状态。

```
nebula> SHOW DRAINER SYNC STATUS;
+--------+-------------+-----------+--------------+
| PartId | Sync Status | LogId Lag | Time Latency |
+--------+-------------+-----------+--------------+
| 0      | "ONLINE"    | 0         | 0            |
| 1      | "ONLINE"    | 0         | 0            |
| 2      | "ONLINE"    | 0         | 0            |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
| 3      | "ONLINE"    | 0         | 0            |
| 4      | "ONLINE"    | 0         | 0            |
| 5      | "ONLINE"    | 0         | 0            |
| 6      | "ONLINE"    | 0         | 0            |
| 7      | "ONLINE"    | 0         | 0            |
| 8      | "ONLINE"    | 0         | 0            |
| 9      | "ONLINE"    | 0         | 0            |
| 10     | "ONLINE"    | 0         | 0            |
| 11     | "ONLINE"    | 0         | 0            |
| 12     | "ONLINE"    | 0         | 0            |
| 13     | "ONLINE"    | 0         | 0            |
| 14     | "ONLINE"    | 0         | 0            |
| 15     | "ONLINE"    | 0         | 0            |
+--------+-------------+-----------+--------------+
```

执行 `SHOW DRAINER SYNC STATUS` 命令，返回结果中的参数说明如下：

参数	说明
PartId	主集群中图空间对应的分片 ID。当值为 `0` 时，表示要同步的 Meta 所在的分片 ID。当为其他值时，表示要同步的 Storage 所在的分片ID。
Sync Status	表示 drainer 的状态。 当值为 `ONLINE` 时，drainer 持续发送 WAL 给从集群的 `metaClient` / `storageClient` 进行同步。 当值为 `OFFLINE` 时，drainer 停止发送 WAL 给从集群的 `metaClient` / `storageClient` 进行同步。
LogId Lag	表示的 Log ID 间隔，也就是从集群 drainer 中对应分片还有多少条 Log 往从集群的 `metaClient` / `storageClient` 进行同步。 当值为 `0` 时，表示从集群的 drainer 中对应的分片没有 Log 需要同步。
Time Latency	从集群 drainer 中对应分片接收到的最新 Log 的 WAL 中的时间戳与已经同步给从集群的最后一条 Log 的 WAL 中的时间戳差值。 当值为 `0` 时，表示 drainer 中对应分片数据已经同步至从集群中。 单位：毫秒（ms）。

11.2.7 切换主从集群

如果因为业务需要进行数据迁移，或者灾备恢复后需要切换主从集群，需要手动进行切换。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

在切换主从之前需要为新的主集群搭建并启动 listener 服务（示例 IP 为`192.168.10.105`），为新的从集群搭建并启动 drainer 服务（示例 IP 为`192.168.10.106`）。

Copyright © Yueshu Graph Technology - All rights reserved.

1. 登录旧的主集群并设置图空间为只读，防止有新的数据写入旧的主集群，而导致数据不一致。

```
nebula> USE basketballplayer;
nebula> SET VARIABLES read_only=true;
```

2. 查看旧的主集群中的图空间的数据是否已经同步至旧的从集群中，确保旧的主集群中的数据已经同步至旧的从集群中。

a. 在旧的主集群中查看旧的主集群同步数据的状态。

```
nebula> SHOW SYNC STATUS;
```

a. 登录旧的从集群并查看旧的从集群同步数据的状态。

```
nebula> USE replication_basketballplayer;
nebula> SHOW DRAINER SYNC STATUS;
```

当旧的主从集群返回结果中的 `LogId Lag` 和 `Time Latency` 对应的值都为 `0` 时，表示旧的主集群中的数据已经被同步至旧的从集群中。

3. 在旧的从集群中设置图空间为可读写。

```
nebula> SET VARIABLES read_only=false;
```

> **Note**
>
> 如果有业务在等待写数据，此时，可在旧的从集群（新的主集群）中进行业务数据的写入操作。

4. 在旧的从集群中移除 drainer 服务。

```
nebula> REMOVE DRAINER;
```

5. 登录旧的主集群，修改图空间为可读写并移除之前注册的 drainer 服务和之前添加的 listener 服务。

```
nebula> USE basketballplayer;
//需先修改旧的主集群图空间为可读写，否则无法设置 drainer 服务。
nebula> SET VARIABLES read_only=false;
nebula> SIGN OUT DRAINER SERVICE;
nebula> REMOVE LISTENER SYNC;
```

6. 在旧的主集群中将旧的主集群更改为新的从集群。

> **Note**
>
> 确保已为新的从集群搭建并启动 drainer 服务。

```
//添加新的 drainer 服务。
nebula> ADD DRAINER 192.168.10.106:9889;
//设置图空间为只读。
nebula> SET VARIABLES read_only=true;
```

7. 登录旧的从集群，将旧的从集群更改为新的主集群。

> **Note**
>
> 确保已为新的主集群搭建并启动 Meta listener 和 Storage listener 服务。

```
nebula> USE replication_basketballplayer;
nebula> SIGN IN DRAINER SERVICE(192.168.10.106:9889);
nebula> ADD LISTENER SYNC META 192.168.10.105:9569 STORAGE 192.168.10.105:9789 TO SPACE basketballplayer;
```

至此主从集群切换完成。

Copyright © Yueshu Graph Technology - All rights reserved.

11.2.8 权限说明

集群同步操作中各命令需要的用户角色权限不同，不同命令所需的角色权限如下（打勾代表有权限）。

命令	God	Admin	DBA	User	Guest
`SIGN IN / SIGN OUT DRAINER SERVICE`	√				
`ADD / REMOVE LISTENER SYNC`	√	√	√		
`SHOW DRAINER CLIENTS`	√	√	√	√	√
`SHOW LISTENER SYNC`	√	√	√	√	√
`ADD / REMOVE DRAINER`	√	√	√		
`SET VARIABLES read_only`	√				
`SHOW DRAINERS`	√	√	√	√	√

11.2.9 常见问题

主集群中已经有 **data** 了，从集群可以同步到之前的存量 **data** 吗?

可以。对于主集群中的存量 data，主集群的 listener 会从各个分片的 leader 节点拉取快照，然后以 WAL 的形式发送给 drainer。存量 data 相关的 WAL 发送完毕后，开始发送主集群的增量 data 相关的 WAL 给 drainer。

从集群中已经有 **data** 了，数据同步会有影响吗?

仍然会进行全量数据同步。如果从集群中的数据是主集群数据的子集，最终会数据一致；如果不是主集群数据的子集，从集群不会进行反向同步，而且这部分数据可能会受到影响，请确保主从集群数据不会冲突；建议保持从集群数据为空。

从集群中已经有 **Schema** 了，数据同步会有影响吗?

数据同步时主集群中的 Schema 会覆盖从集群中的 Schema，因此可能导致被覆盖的 Schema 对应的数据失效或者数据不一致。请确保从集群中的 Schema 和主集群中的 Schema 没有冲突。

修改主集群的 **Schema** 会影响数据同步吗?

可能会增加数据同步延迟。因为 Schema 数据和 data 数据是分开处理的（Meta listener 和 Storage listener），data 数据同步时，drainer 会检查自身的 Schema 版本，如果版本大于当前存储的版本，说明 Schema 有更新，这时候会暂缓更新，等待从集群中的 Schema 数据先更新完成。

主从集群的机器数量、副本数量、分片数量需要相同吗?

不需要。因为是以图空间为基本单位，主集群不需要知道从集群的架构信息，只需要 listener 将 WAL 发送给 drainer 即可。

如果同步时出现故障，如何修复?

可以根据故障节点，进行如下处理：

- 主集群故障：会导致同步暂停，重启主集群服务即可。
- listener/drainer/从集群故障：服务恢复后，会收到前一节点发送的故障期间的 WAL。例如 drainer 故障恢复后，会收到 listener 发送来的故障期间的 WAL。如果用新的节点替换故障的 drainer 节点或从集群，需要将原节点的数据复制到新节点，否则相当于重新同步全量数据。

如何判断数据同步进度?

用户可以执行 `SHOW SYNC STATUS` 查看主集群发送数据的状态，执行 `SHOW DRAINER SYNC STATUS` 查看从集群接收数据的状态。如果同时满足主集群中的所有数据都发送成功，并且从集群成功接收所有数据，则说明数据同步完成。

Copyright © Yueshu Graph Technology - All rights reserved.

WAL 日志文件过期了对集群数据同步有影响吗?

如果 WAL 日志文件过期了（超过了 `--wal-ttl` 设置的时间），数据会不同步。用户可以通过手动在 Meta 和 Storage 服务的配置文件中添加 `--snapshot_send_files=false` 配置以同步数据。更新文件中的配置后，需要重启服务。关于配置文件的详细信息，参见配置文件简介。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12. 导入与导出

12.1 数据导入导出概述

悦数图数据库提供了一套全面的数据导入导出工具集，这些工具在开源社区版本的基础上进行了扩展和优化，旨在满足多样化的数据交互需求。本文列出相关的导入导出工具。

12.1.1 导入工具

有多种方式可以将数据写入悦数图数据库 3.5.1：

- 使用命令行 -f 的方式导入：可以导入少量准备好的 nGQL 文件，适合少量手工测试数据准备；
- 使用 Importer 导入：导入单机多个 csv 文件，大小没有限制，格式灵活；数据量十亿级以内；
- 使用 Exchange 导入：从 Neo4j, Hive, MySQL 等多种源分布式导入，需要有 Spark 集群；数据量十亿级以上
- 使用 Spark-connector/Flink-connector 导入：有相应组件 (Spark/Flink)，撰写少量代码；
- 使用 C++/GO/Java/Python SDK：编写程序的方式导入，需要有一定编程和调优能力。

下图给出了几种方式的定位：

12.1.2 导出工具

目前支持使用Exchange 导出功能将悦数图数据库 3.5.1数据导出至 CSV 文件或另一个图空间（支持不同悦数图数据库集群）中。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.2 NebulaGraph Importer

NebulaGraph Importer（简称 Importer）是一款悦数图数据库的 CSV 文件单机导入工具，可以读取并导入多种数据源的 CSV 文件数据。

12.2.1 功能

- 支持多种数据源，包括本地、S3、OSS、HDFS、FTP、SFTP。
- 支持导入 CSV 格式文件的数据。单个文件内可以包含多种 Tag、多种 Edge type 或者二者混合的数据。
- 支持同时连接多个 Graph 服务进行导入并且动态负载均衡。
- 支持失败后重连、重试。
- 支持多维度显示统计信息，包括导入时间、导入百分比等。统计信息支持打印在 Console 或日志中。

12.2.2 优势

- 轻量快捷：不需要复杂环境即可使用，快速导入数据。
- 灵活筛选：通过配置文件可以实现对 CSV 文件数据的灵活筛选。

12.2.3 版本兼容性

NebulaGraph Importer 版本和悦数图数据库内核的版本对应关系如下。

悦数图数据库版本	**NebulaGraph Importer** 版本
3.x.x	3.x.x、4.x.x
2.x.x	2.x.x、3.x.x

Note

Importer 4.0.0 对 Importer 进行了重做，性能得到了提高，但配置文件不兼容旧版本。建议使用新版 Importer。

12.2.4 更新说明

Release notes

12.2.5 前提条件

在使用 NebulaGraph Importer 之前，请确保：

- 已部署悦数图数据库服务。部署方式如下：
- RPM/DEB 包安装
- 悦数图数据库中已创建 Schema，包括图空间、Tag 和 Edge type，或者通过参数 `manager.hooks.before.statements` 设置。

12.2.6 操作步骤

创建 **CSV** 文件

准备好待导入的 CSV 文件并配置 YAML 文件，即可使用本工具向悦数图数据库批量导入数据。

Copyright © Yueshu Graph Technology - All rights reserved.

> **Note**
>
> YAML 配置文件说明请参见下文的配置文件说明。

下载二进制包运行

1. 在 Release 页面下载和安装二进制包，并添加执行权限。

> **Note**
>
> 使用 RPM/DEB 包安装的文件路径为 `/usr/bin/nebula-importer` 。

2. 在 `nebula-importer` 的安装目录下，执行以下命令导入数据。

```
$ ./<binary_file_name> --config <yaml_config_file_path>
```

源码编译运行

编译源码需要部署 Golang 环境。详情请参见 Golang 环境搭建。

1. 克隆仓库。

```
$ git clone -b release-4.0 https://github.com/vesoft-inc/nebula-importer.git
```

> **Note**
>
> 请使用正确的分支。不同分支的 rpc 协议不同。

2. 进入目录 `nebula-importer` 。

```
$ cd nebula-importer
```

3. 编译源码。

```
$ make build
```

4. 开始导入数据。

```
$ ./bin/nebula-importer --config <yaml_config_file_path>
```

Docker 方式运行

使用 Docker 可以不必在本地安装 Go 语言环境，只需要拉取 NebulaGraph Importer 的镜像，并将本地配置文件和 CSV 数据文件挂载到容器中。命令如下：

```
$ docker pull vesoft/nebula-importer:<version>
$ docker run --rm -ti \
      --network=host \
      -v <config_file>:<config_file> \
      -v <data_dir>:<data_dir> \
      vesoft/nebula-importer:<version> \
      --config <config_file>
```

- `<config_file>` ：填写 YAML 配置文件的绝对路径。
- `<data_dir>` ：填写 CSV 数据文件的绝对路径。如果文件不在本地，请忽略该参数。
- `<version>` ：填写 Importer 的版本号，请填写 `v4` 。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

建议使用相对路径。如果使用本地绝对路径，请检查路径映射到 Docker 中的路径。

例如：

```
$ docker pull vesoft/nebula-importer:v4
$ docker run --rm -ti \
      --network=host \
      -v /home/user/config.yaml:/home/user/config.yaml \
      -v /home/user/data:/home/user/data \
      vesoft/nebula-importer:v4 \
      --config /home/user/config.yaml
```

12.2.7 配置文件说明

NebulaGraph Importer 的 Github 内提供多种示例配置文件。配置文件用来描述待导入文件信息、悦数图数据库服务器信息等。下文将分类介绍配置文件内的字段。

Note

如果用户下载的是二进制包，请手动创建配置文件。

Client 配置

Client 配置存储客户端连接悦数图数据库相关的配置。

示例配置如下：

```
client:
  version: v3
  address: "192.168.1.100:9669,192.168.1.101:9669"
  user: root
  password: nebula
  concurrencyPerAddress: 10
  reconnectInitialInterval: 1s
  retry: 3
  retryInitialInterval: 1s
```

参数	默认值	是否必须	说明
`client.version`	`v3`	是	指定连接的悦数图数据库的大版本。当前仅支持 `v3` 。
`client.address`	`"127.0.0.1:9669"`	是	指定连接的悦数图数据库地址。多个地址用英文逗号（`,`）分隔。
`client.user`	`root`	否	悦数图数据库的用户名。
`client.password`	`nebula`	否	悦数图数据库用户名对应的密码。
`client.concurrencyPerAddress`	`10`	否	单个 Graph 服务的客户端并发连接数。
`client.retryInitialInterval`	`1s`	否	重连间隔时间。
`client.retry`	`3`	否	nGQL 语句执行失败的重试次数。
`client.retryInitialInterval`	`1s`	否	重试间隔时间。

Manager 配置

Manager 配置是连接数据库后的人为控制配置。

示例配置如下：

```
manager:
  spaceName: basic_string_examples
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  batch: 128
  readerConcurrency: 50
  importerConcurrency: 512
  statsInterval: 10s
  hooks:
    before:
      - statements:
        - |
            DROP SPACE IF EXISTS basic_string_examples;
            CREATE SPACE IF NOT EXISTS basic_string_examples(partition_num=5, replica_factor=1, vid_type=int);
            USE basic_string_examples;
        wait: 10s
    after:
      - statements:
          - |
            SHOW SPACES;
```

参数	默认值	是否必须	说明
`manager.spaceName`	-	是	指定数据要导入的图空间。不支持同时导入多个图空间。
`manager.batch`	`128`	否	执行语句的批处理量（全局配置）。对某个数据源单独设置批处理量可以使用下文的 `sources.batch` 。
`manager.readerConcurrency`	`50`	否	读取器读取数据源的并发数。
`manager.importerConcurrency`	`512`	否	生成待执行的 nGQL 语句的并发数，然后会调用客户端执行这些语句。
`manager.statsInterval`	`10s`	否	打印统计信息的时间间隔。
`manager.hooks.before.[].statements`	-	否	导入前在图空间内执行的命令。
`manager.hooks.before.[].wait`	-	否	执行 `statements` 语句后的等待时间。
`manager.hooks.after.[].statements`	-	否	导入后在图空间内执行的命令。
`manager.hooks.after.[].wait`	-	否	执行 `statements` 语句后的等待时间。

Log 配置

Log 配置是设置日志相关配置。

示例配置如下：

```
log:
  level: INFO
  console: true
  files:
   - logs/nebula-importer.log
```

参数	默认值	是否必须	说明
`log.level`	`INFO`	否	日志级别。可选值为 `DEBUG` 、 `INFO` 、 `WARN` 、 `ERROR` 、 `PANIC` 、 `FATAL` 。
`log.console`	`true`	否	存储日志时是否将日志同步打印到 Console。
`log.files`	-	否	日志文件路径。需手动创建日志文件目录。

Source 配置

Source 配置中需要配置数据源信息、数据处理方式和 Schema 映射。

示例配置如下：

```
sources:
  - path: ./person.csv  # 指定存储数据文件的路径。如果使用相对路径，则路径和当前配置文件目录拼接。也支持通配符文件名，例如：./follower-*.csv，请确保所有匹配的文件具有相同的架构。
#  - s3: # AWS S3
#      endpoint: endpoint    # 可选。S3 服务端点，如果使用 AWS S3 可以省略。
#      region: us-east-1     # 必填。S3 服务的区域。
#      bucket: gdelt-open-data    # 必填。S3 服务中的 bucket。
#      key: events/20190918.export.csv     # 必填。S3 服务中文件的 key。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
#      accessKeyID: ""    # 可选。S3 服务的访问密钥。如果是公共数据，则无需配置。
#      accessKeySecret: ""     # 可选。S3 服务的密钥。如果是公共数据，则无需配置。
#  - oss:
#      endpoint: https://oss-cn-hangzhou.aliyuncs.com    # 必填。OSS 服务端点。
#      bucket: bucketName    # 必填。OSS 服务中的 bucket。
#      key: objectKey    # 必填。OSS 服务中文件的 object key。
#      accessKeyID: accessKey    # 必填。OSS 服务的访问密钥。
#      accessKeySecret: secretKey    # 必填。OSS 服务的秘钥。
#  - ftp:
#      host: 192.168.0.10    # 必填。FTP 服务的主机。
#      port: 21    # 必填。FTP 服务的端口。
#      user: user    # 必填。FTP 服务的用户名。
#      password: password    # 必填。FTP 服务的密码。
#      path: "/events/20190918.export.csv"    # FTP 服务中文件的路径。
#  - sftp:
#      host: 192.168.0.10    # 必填。SFTP 服务的主机。
#      port: 22    # 必填。SFTP 服务的端口。
#      user: user    # 必填。SFTP 服务的用户名。
#      password: password    # 可选。SFTP 服务的密码。
#      keyFile: keyFile    # 可选。SFTP 服务的 SSH 密钥文件路径。
#      keyData: keyData    $ 可选。SFTP 服务的 SSH 密钥文件内容。
#      passphrase: passphrase    # 可选。SFTP 服务的 SSH 密钥密码。
#      path: "/events/20190918.export.csv"    # 必填。SFTP 服务中文件的路径。
#  - hdfs:
#      address: "127.0.0.1:8020"    # 必填。HDFS 服务的地址。
#      user: "hdfs"    # 可选。HDFS 服务的用户名。
#      path: "/events/20190918.export.csv"    # 必填。HDFS 服务中文件的路径。
    batch: 256
    csv:
      delimiter: "|"
      withHeader: false
      lazyQuotes: false
    tags:
    - name: Person
      id:
        type: "STRING"
        function: "hash"
#       index: 0
        concatItems:
          - person_
          - 0
          - _id
      props:
        - name: "firstName"
          type: "STRING"
          index: 1
        - name: "lastName"
          type: "STRING"
          index: 2
        - name: "gender"
          type: "STRING"
          index: 3
          nullable: true
          defaultValue: female
        - name: "birthday"
          type: "DATE"
          index: 4
          nullable: true
          nullValue: _NULL_
        - name: "creationDate"
          type: "DATETIME"
          index: 5
        - name: "locationIP"
          type: "STRING"
          index: 6
        - name: "browserUsed"
          type: "STRING"
          index: 7
  - path: ./knows.csv
    batch: 256
    edges:
    - name: KNOWS # person_knows_person
      src:
        id:
          type: "STRING"
          concatItems:
            - person_
            - 0
            - _id
      dst:
        id:
          type: "STRING"
          concatItems:
            - person_
            - 1
            - _id
      props:
        - name: "creationDate"
          type: "DATETIME"
          index: 2
          nullable: true
          nullValue: _NULL_
          defaultValue: 0000-00-00T00:00:00
```


Copyright © Yueshu Graph Technology - All rights reserved.

配置主要包括以下几个部分:

- 指定数据源信息。
- 指定执行语句的批处理量。
- 指定 CSV 文件格式信息。
- 指定 Tag 的模式映射。
- 指定 Edge type 的模式映射。

Copyright © Yueshu Graph Technology - All rights reserved.

参数	默认值	是否必须	说明
`sources.path` `sources.s3` `sources.oss` `sources.ftp` `sources.sftp` `sources.hdfs`	-	否	指定数据源信息，例如本地文件、HDFS、S3 等。一个 `so` 参见示例内的注释。
`sources.batch`	`256`	否	导入该数据源时执行语句的批处理量。优先级高于 `manager.`
`sources.csv.delimiter`	`,`	否	CSV 文件的分隔符。仅支持 1 个字符的字符串分隔符。使 为：`"\x03"` 或 `"\u0003"` 。关于 yaml 格式特殊字符转义的
`sources.csv.withHeader`	`false`	否	是否忽略 CSV 文件中的第一条记录。
`sources.csv.lazyQuotes`	`false`	否	是否允许惰性解析引号。如果值为 `true` ，引号可以出现在
`sources.tags.name`	-	是	Tag 名称。
`sources.tags.id.type`	`STRING`	否	VID 的类型。
`sources.tags.id.function`	-	否	生成 VID 的函数。目前仅支持 `hash` 。
`sources.tags.id.index`	-	否	VID 对应的数据文件中的列号。如果未配置 `sources.tags.id`
`sources.tags.id.concatItems`	-	否	用于连接两个或多个数组，连接项可以是 `string` 、 `int` 或者 效。
`sources.tags.ignoreExistedIndex`	`true`	否	是否启用 `IGNORE_EXISTED_INDEX` ，即插入点后不更新索引。
`sources.tags.props.name`	-	是	VID 上的属性名称，必须与数据库中的属性相同。
`sources.tags.props.type`	`STRING`	否	VID 上属性的数据类型。目前支持 `BOOL` 、 `INT` 、 `FLOAT` 、 `DO` `GEOGRAPHY(LINESTRING)` 、 `GEOGRAPHY(POLYGON)` 。
`sources.tags.props.index`	-	是	属性值对应的数据文件中的列号。
`sources.tags.props.nullable`	`false`	否	属性是否可以为 `NULL` ，可选 `true` 或者 `false` 。
`sources.tags.props.nullValue`	-	否	`nullable` 设置为 `true` 时，属性的值与 `nullValue` 相等则将该
`sources.tags.props.alternativeIndices`	-	否	当 `nullable` 为 `false` 时忽略。该属性根据索引顺序从文件中
`sources.tags.props.defaultValue`	-	否	当 `nullable` 为 `false` 时忽略。根据 `index` 和 `alternativeIndices`
`sources.edges.name`	-	是	Edge type 名称。
`sources.edges.src.id.type`	`STRING`	否	边上起点 VID 的数据类型。
`sources.edges.src.id.index`	-	是	边上起点 VID 对应的数据文件中的列号。
`sources.edges.dst.id.type`	`STRING`	否	边上终点 VID 的数据类型。
`sources.edges.dst.id.index`	-	是	边上终点 VID 对应的数据文件中的列号。
`sources.edges.rank.index`	-	否	边上 RANK 对应的数据文件中的列号。
`sources.edges.ignoreExistedIndex`	`true`	否	是否启用 `IGNORE_EXISTED_INDEX` ，即插入点后不更新索引。
`sources.edges.props.name`	-	是	边上属性的名称，必须与数据库中的属性相同。
`sources.edges.props.type`	`STRING`	否	边上属性的数据类型。目前支持 `BOOL` 、 `INT` 、 `FLOAT` 、 `DOUBL` `GEOGRAPHY(LINESTRING)` 、 `GEOGRAPHY(POLYGON)` 。
`sources.edges.props.index`	-	是	属性值对应的数据文件中的列号。
`sources.edges.props.nullable`	-	否	属性是否可以为 `NULL` ，可选 `true` 或者 `false` 。
`sources.edges.props.nullValue`	-	否	`nullable` 设置为 `true` 时，属性的值与 `nullValue` 相等则将该

Copyright © Yueshu Graph Technology - All rights reserved.

参数	默认值	是否必须	说明
`sources.edges.props.defaultValue`	-	否	当 `nullable` 为 `false` 时忽略。根据 `index` 和 `alternativeIndices`

Note

CSV 文件中列的序号从 0 开始，即第一列的序号为 0，第二列的序号为 1。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.3 NebulaGraph Exchange

12.3.1 认识 NebulaGraph Exchange

什么是 **NebulaGraph Exchange**

NebulaGraph Exchange（简称 Exchange）是一款 Apache Spark™ 应用，用于在分布式环境中将集群中的数据批量迁移到悦数图数库中，能支持多种不同格式的批式数据和流式数据的迁移。

Exchange 由 Reader、Processor 和 Writer 三部分组成。Reader 读取不同来源的数据返回 DataFrame 后，Processor 遍历 DataFrame 的每一行，根据配置文件中 `fields` 的映射关系，按列名获取对应的值。在遍历指定批处理的行数后，Writer 会将获取的数据一次性写入到悦数图数库中。下图描述了 Exchange 完成数据转换和迁移的过程。

版本系列

Exchange 有社区版和企业版两个系列，二者功能不同。社区版在 GitHub 开源开发，企业版属于悦数图数库企业套餐。详情参见版本对比。

适用场景

Exchange 适用于以下场景：

- 需要将来自 Kafka、Pulsar 平台的流式数据，如日志文件、网购数据、游戏内玩家活动、社交网站信息、金融交易或地理空间服务，以及来自数据中心内所连接设备或仪器的遥测数据等转化为属性图的点或边数据，并导入悦数图数库。
- 需要从关系型数据库（如 MySQL）或者分布式文件系统（如 HDFS）中读取批式数据，如某个时间段内的数据，将它们转化为属性图的点或边数据，并导入悦数图数库。
- 需要将大批量数据生成悦数图数库能识别的 SST 文件，再导入悦数图数库。
- 需要导出悦数图数库中保存的数据。

Enterpriseonly

仅企业版 Exchange 支持从悦数图数库中导出数据。

Copyright © Yueshu Graph Technology - All rights reserved.

产品优点

Exchange 具有以下优点：

- 适应性强：支持将多种不同格式或不同来源的数据导入悦数图数据库，便于迁移数据。
- 支持导入 SST：支持将不同来源的数据转换为 SST 文件，用于数据导入。
- 支持 SSL 加密：支持在 Exchange 与悦数图数据库之间建立 SSL 加密传输通道，保障数据安全。
- 支持断点续传：导入数据时支持断点续传，有助于节省时间，提高数据导入效率。

> **Note**
>
> 目前仅迁移 Neo4j 数据时支持断点续传。

- 异步操作：会在源数据中生成一条插入语句，发送给 Graph 服务，最后再执行插入操作。
- 灵活性强：支持同时导入多个 Tag 和 Edge type，不同 Tag 和 Edge type 可以是不同的数据来源或格式。
- 统计功能：使用 Apache Spark™ 中的累加器统计插入操作的成功和失败次数。
- 易于使用：采用 HOCON（Human-Optimized Config Object Notation）配置文件格式，具有面向对象风格，便于理解和操作。

版本兼容性

Exchange 支持 Spark 版本 2.2.x、2.4.x 和 3.x.x，针对不同 Spark 版本命名为：`nebula-exchange_spark_2.2`、`nebula-exchange_spark_2.4` 和 `nebula-exchange_spark_3.0`。

Copyright © Yueshu Graph Technology - All rights reserved.

NebulaGraph Exchange 版本（即 JAR 包版本）、悦数图数据库内核版本和 Spark 版本对应关系如下。

Exchange 版本	悦数图数据库版本	**Spark** 版本
nebula-exchange_spark_3.0-3.0-SNAPSHOT.jar	nightly	3.3.x、3.2.x、3.1.x、3.0.x
nebula-exchange_spark_2.4-3.0-SNAPSHOT.jar	nightly	2.4.x
nebula-exchange_spark_2.2-3.0-SNAPSHOT.jar	nightly	2.2.x
nebula-exchange_spark_3.0-3.4.0.jar	3.x.x	3.3.x、3.2.x、3.1.x、3.0.x
nebula-exchange_spark_2.4-3.4.0.jar	3.x.x	2.4.x
nebula-exchange_spark_2.2-3.4.0.jar	3.x.x	2.2.x
nebula-exchange_spark_3.0-3.3.0.jar	3.x.x	3.3.x、3.2.x、3.1.x、3.0.x
nebula-exchange_spark_2.4-3.3.0.jar	3.x.x	2.4.x
nebula-exchange_spark_2.2-3.3.0.jar	3.x.x	2.2.x
nebula-exchange_spark_3.0-3.0.0.jar	3.x.x	3.3.x、3.2.x、3.1.x、3.0.x
nebula-exchange_spark_2.4-3.0.0.jar	3.x.x	2.4.x
nebula-exchange_spark_2.2-3.0.0.jar	3.x.x	2.2.x
nebula-exchange-2.6.3.jar	2.6.1、2.6.0	2.4.x
nebula-exchange-2.6.2.jar	2.6.1、2.6.0	2.4.x
nebula-exchange-2.6.1.jar	2.6.1、2.6.0	2.4.x
nebula-exchange-2.6.0.jar	2.6.1、2.6.0	2.4.x
nebula-exchange-2.5.2.jar	2.5.1、2.5.0	2.4.x
nebula-exchange-2.5.1.jar	2.5.1、2.5.0	2.4.x
nebula-exchange-2.5.0.jar	2.5.1、2.5.0	2.4.x
nebula-exchange-2.1.0.jar	2.0.1、2.0.0	2.4.x
nebula-exchange-2.0.1.jar	2.0.1、2.0.0	2.4.x
nebula-exchange-2.0.0.jar	2.0.1、2.0.0	2.4.x

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

Copyright © Yueshu Graph Technology - All rights reserved.

数据源

Exchange 3.5.0 支持将以下格式或来源的数据转换为悦数图数据库能识别的点和边数据，然后通过 nGQL 语句的形式导入悦数图数据库：

- 存储在 HDFS 或本地的数据：
- Apache Parquet
- Apache ORC
- JSON
- CSV
- Apache HBase™
- 数据仓库：
- Hive
- MaxCompute
- 图数据库：Neo4j（Client 版本 2.4.5-M1）
- 关系型数据库：
- MySQL
- PostgreSQL
- Oracle
- 列式数据库：ClickHouse
- 流处理软件平台：Apache Kafka®
- 发布/订阅消息平台：Apache Pulsar 2.4.5
- JDBC 数据源

除了用 nGQL 语句的形式导入数据，Exchange 还支持将数据源的数据生成 SST 文件，然后通过 Console 导入 SST 文件。

此外，企业版 Exchange 支持以悦数图数据库为源，将数据导出到 CSV 文件或另一个图空间。

更新说明

Release

视频

- 悦数图数据库数据导入工具——Exchange（3 分 08 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

使用限制

本文描述 Exchange 3.5.0 的一些使用限制。

使用环境

Exchange 3.x 支持以下操作系统：

- CentOS 7
- macOS

软件依赖

为保证 Exchange 正常工作，请确认机器上已经安装如下软件：

- Java 1.8 版本
- Scala 2.10.7、2.11.12 或 2.12.10 版本
- Apache Spark。使用 Exchange 从不同数据源导出数据对 Spark 版本的要求如下：

  Note

  使用 Exchange 时，需根据 Spark 版本选择相应的 JAR 文件。例如，当 Spark 版本为 2.4 时，选择 nebula-exchange_spark_2.4-3.5.0.jar。

数据源	**Spark 2.2**	**Spark 2.4**	**Spark 3**
CSV 文件	支持	支持	支持
JSON 文件	支持	支持	支持
ORC 文件	支持	支持	支持
Parquet 文件	支持	支持	支持
HBase	支持	支持	支持
MySQL	支持	支持	支持
PostgreSQL	支持	支持	支持
Oracle	支持	支持	支持
ClickHouse	支持	支持	支持
Neo4j	不支持	支持	不支持
Hive	支持	支持	支持
MaxCompute	不支持	支持	不支持
Pulsar	不支持	支持	未测试
Kafka	不支持	支持	未测试
悦数图数据库	不支持	支持	不支持

在以下使用场景，还需要部署 Hadoop Distributed File System (HDFS)：

- 迁移 HDFS 的数据
- 生成 SST 文件

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.3.2 获取 NebulaGraph Exchange

本文介绍如何获取 NebulaGraph Exchange 的 JAR 文件。

直接下载 JAR 文件

社区版 Exchange 的 JAR 文件可以直接下载。

要下载企业版 Exchange，需先获取悦数图数据库企业版套餐。

编译源代码获取 JAR 文件

社区版 Exchange 的 JAR 文件还可以通过编译源代码获取。下文介绍如何编译 Exchange 源代码。

> **Enterpriseonly**
>
> 企业版 Exchange 仅能在悦数图数据库企业版套餐中获取。

前提条件

- 安装 Maven。
- 根据数据源安装需要的 Spark 版本，从各数据源导出数据支持的 Spark 版本参见软件依赖。

操作步骤

1. 在根目录克隆仓库 `nebula-exchange` 。

```
git clone -b release-3.5 https://github.com/vesoft-inc/nebula-exchange.git
```

2. 切换到目录 `nebula-exchange` 。

```
cd nebula-exchange
```

3. 根据 Exchange 使用环境中的 Spark 版本打包 Exchange。

- Spark 2.2：

```
mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true \
-pl nebula-exchange_spark_2.2 -am -Pscala-2.11 -Pspark-2.2
```

- Spark 2.4：

```
mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true \
-pl nebula-exchange_spark_2.4 -am -Pscala-2.11 -Pspark-2.4
```

- Spark 3.0：

```
mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true \
-pl nebula-exchange_spark_3.0 -am -Pscala-2.12 -Pspark-3.0
```

编译成功后，可以在 `nebula-exchange_spark_x.x/target/` 目录里找到 `nebula-exchange_spark_x.x-release-3.5.jar` 文件。 `x.x` 代表 Spark 版本，例如 `2.4` 。

> **Note**
>
> JAR 文件版本号会因 NebulaGraph Java Client 的发布版本而变化。用户可以在 Releases 页面查看最新版本。

迁移数据时，用户可以参考配置文件 `target/classes/application.conf` 。

Copyright © Yueshu Graph Technology - All rights reserved.

下载依赖包失败

如果编译时下载依赖包失败：

- 检查网络设置，确认网络正常。
- 修改 Maven 安装目录下 `libexec/conf/settings.xml` 文件的 `mirror` 部分：

```
<mirror>
 <id>alimaven</id>
 <mirrorOf>central</mirrorOf>
 <name>aliyun maven</name>
 <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
</mirror>
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.3.3 参数说明

导入命令参数

完成配置文件修改后，可以运行以下命令将指定来源的数据导入悦数图数据库数据库。

- 首次导入

```
<spark_install_path>/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-2.x.y.jar_path> -c <application.conf_path>
```

- 导入 reload 文件

  如果首次导入时有一些数据导入失败，会将导入失败的数据存入 reload 文件，可以用参数 `-r` 尝试导入 reload 文件。

```
<spark_install_path>/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-2.x.y.jar_path> -c <application.conf_path> -r "<reload_file_path>"
```

Note

JAR 文件版本号以实际编译得到的 JAR 文件名称为准。

Faq

如果使用 yarn-cluster 模式提交任务，请参考如下示例，尤其是示例中的两个 `--conf` 。

```
$SPARK_HOME/bin/spark-submit     --master yarn-cluster \
--class com.vesoft.nebula.exchange.Exchange \
--files application.conf \
--conf spark.driver.extraClassPath=./ \
--conf spark.executor.extraClassPath=./ \
nebula-exchange-3.5.0.jar \
-c application.conf
```

下表列出了命令的相关参数。

参数	是否必需	默认值	说明
`--class`	是	无	指定驱动的主类。
`--master`	是	无	指定 Spark 集群中 master 进程的 URL。详情请参见 master-urls。
`-c` / `--config`	是	无	指定配置文件的路径。
`-h` / `--hive`	否	`false`	添加这个参数表示支持从 Hive 中导入数据。
`-D` / `--dry`	否	`false`	添加这个参数表示检查配置文件的格式是否符合要求，但不会校验 `tags` 和 `edges` 的配置项是否正确。正式导入数据时不能添加这个参数。
-r / --reload	否	无	指定需要重新加载的 reload 文件路径。

更多 Spark 的参数配置说明请参见 Spark Configuration。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

配置说明

本文介绍使用 NebulaGraph Exchange 时如何修改配置文件 `application.conf`。

修改配置文件之前，建议根据数据源复制并修改文件名称，便于区分。例如数据源为 CSV 文件，可以复制为 `csv_application.conf`。

配置文件的内容主要分为如下几类：

- Spark 相关配置
- Hive 配置（可选）
- 悦数图数据库相关配置
- 点配置
- 边配置

SPARK 相关配置

本文只列出部分 Spark 参数，更多参数请参见官方文档。

参数	数据类型	默认值	是否必须	说明
`spark.app.name`	string	-	否	Spark 驱动程序名称。
`spark.driver.cores`	int	`1`	否	驱动程序使用的 CPU 核数，仅适用于集群模式。
`spark.driver.maxResultSize`	string	`1G`	否	单个 Spark 操作（例如 collect）时，所有分区的序列化结果的总大小限制（字节为单位）。最小值为 1M，0 表示无限制。
`spark.executor.memory`	string	`1G`	否	Spark 驱动程序使用的内存量，可以指定单位，例如 512M、1G。
`spark.cores.max`	int	`16`	否	当驱动程序以“粗粒度”共享模式在独立部署集群或 Mesos 集群上运行时，跨集群（而非从每台计算机）请求应用程序的最大 CPU 核数。如果未设置，则值为 Spark 的独立集群管理器上的 `spark.deploy.defaultCores` 或 Mesos 上的 infinite（所有可用的内核）。

HIVE 配置（可选）

如果 Spark 和 Hive 部署在不同集群，才需要配置连接 Hive 的参数，否则请忽略这些配置。

参数	数据类型	默认值	是否必须	说明
`hive.warehouse`	string	-	是	HDFS 中的 warehouse 路径。用双引号括起路径，以 `hdfs://` 开头。
`hive.connectionURL`	string	-	是	JDBC 连接的 URL。例如 `"jdbc:mysql://127.0.0.1:3306/hive_spark?characterEncoding=UTF-8"`。
`hive.connectionDriverName`	string	`"com.mysql.jdbc.Driver"`	是	驱动名称。
`hive.connectionUserName`	list[string]	-	是	连接的用户名。
`hive.connectionPassword`	list[string]	-	是	用户名对应的密码。

Copyright © Yueshu Graph Technology - All rights reserved.

悦数图数据库相关配置

Copyright © Yueshu Graph Technology - All rights reserved.

参数	数据类型	默认值	是否必须	说明
`nebula.address.graph`	list[string]	`["127.0.0.1:9669"]`	是	所有 Graph 服务的地址，包括 IP 和端口，多个地址用英文逗号（,）分隔。格式为 `["ip1:port1","ip2:port2","ip3:port3"]`。
`nebula.address.meta`	list[string]	`["127.0.0.1:9559"]`	是	所有 Meta 服务的地址，包括 IP 和端口，多个地址用英文逗号（,）分隔。格式为 `["ip1:port1","ip2:port2","ip3:port3"]`。
`nebula.user`	string	-	是	拥有悦数图数据库写权限的用户名。
`nebula.pswd`	string	-	是	用户名对应的密码。
`nebula.space`	string	-	是	需要导入数据的的图空间名称。
`nebula.ssl.enable.graph`	bool	`false`	是	开启 Exchange 与 Graph 服务之间的 SSL 加密传输。当值为 `true` 时开启，下方的 SSL 相关参数生效。如果 Exchange 运行在多机集群上，在设置以下 SSL 相关路径时，需要在每台机器的相同路径都存储相应的文件。
`nebula.ssl.enable.meta`	bool	`false`	是	开启 Exchange 与 Meta 服务之间的 SSL 加密传输。当值为 `true` 时开启，下方的 SSL 相关参数生效。如果 Exchange 运行在多机集群上，在设置以下 SSL 相关路径时，需要在每台机器的相同路径都存储相应的文件。
`nebula.ssl.sign`	string	`ca`	是	签名方式，可选值：`ca`（CA 签名）或 `self`（自签名）。
`nebula.ssl.ca.param.caCrtFilePath`	string	`"/path/caCrtFilePath"`	是	`nebula.ssl.sign` 的值为 `ca` 时生效，用于指定 CA 证书的存储路径。
`nebula.ssl.ca.param.crtFilePath`	string	`"/path/crtFilePath"`	是	`nebula.ssl.sign` 的值为 `ca` 时生效，用于指定 CRT 证书的存储路径。
`nebula.ssl.ca.param.keyFilePath`	string	`"/path/keyFilePath"`	是	`nebula.ssl.sign` 的值为 `ca` 时生效，用于指定私钥文件的存储路径。
`nebula.ssl.self.param.crtFilePath`	string	`"/path/crtFilePath"`	是	`nebula.ssl.sign` 的值为 `self` 时生效，用于指定 CRT 证书的存储路径。
`nebula.ssl.self.param.keyFilePath`	string	`"/path/keyFilePath"`	是	`nebula.ssl.sign` 的值为 `self` 时生效，用于指定私钥文件的存储路径。
`nebula.ssl.self.param.password`	string	`"nebula"`	是	`nebula.ssl.sign` 的值为 `self` 时生效，用于指定密码文件的存储路径。
`nebula.path.local`	string	`"/tmp"`	否	导入 SST 文件时需要设置本地 SST 文件路径。
`nebula.path.remote`	string	`"/sst"`	否	导入 SST 文件时需要设置远端 SST 文件路径。
`nebula.path.hdfs.namenode`	string	`"hdfs://name_node:9000"`	否	导入 SST 文件时需要设置 HDFS 的 namenode。
`nebula.connection.timeout`	int	`3000`	否	Thrift 连接的超时时间，单位为 ms。
`nebula.connection.retry`	int	`3`	否	Thrift 连接重试次数。
`nebula.execution.retry`	int	`3`	否	nGQL 语句执行重试次数。

Copyright © Yueshu Graph Technology - All rights reserved.

参数	数据类型	默认值	是否必须	说明
`nebula.error.max`	int	`32`	否	导入过程中的最大失败次数。当失败次数达到最大值时，提交的 Spark 作业将自动停止。
`nebula.error.output`	string	`/tmp/errors`	否	输出错误日志的路径。错误日志保存执行失败的 nGQL 语句。
`nebula.rate.limit`	int	`1024`	否	导入数据时令牌桶的令牌数量限制。
`nebula.rate.timeout`	int	`1000`	否	令牌桶中拿取令牌的超时时间，单位：毫秒。

Note

悦数图数据库默认不支持无 Tag 的点。如果需要导入无 Tag 的点，需要先在集群内开启支持无 Tag 点，然后在 Exchange 的配置文件内新增 `nebula.enableTagless` 参数，值为 `true`。示例如下：

```
nebula: {
    address:{
      graph:["127.0.0.1:9669"]
      meta:["127.0.0.1:9559"]
    }
    user: root
    pswd: nebula
    space: test
    enableTagless: true
    ......

 }
```

点配置

对于不同的数据源，点的配置也有所不同，有很多通用参数，也有部分特有参数，配置时需要配置通用参数和不同数据源的特有参数。

Copyright © Yueshu Graph Technology - All rights reserved.

通用参数

参数	数据类型	默认值	是否必须	说明
`tags.name`	string	-	是	悦数图数据库中定义的 Tag 名称。
`tags.type.source`	string	-	是	指定数据源。例如 `csv` 。
`tags.type.sink`	string	`client`	是	指定导入方式，可选值为 `client` 和 `SST` 。
`tags.writeMode`	string	`INSERT`	否	对数据的批量操作类型，包括批量导入、更新和删除。可选值为 `INSERT` 、 `UPDATE` 、 `DELETE` 。
`tags.deleteEdge`	string	`false`	否	进行批量删除操作时是否删除该点关联的出边和入边。 `tags.writeMode` 为 `DELETE` 时该参数生效。
`tags.fields`	list[string]	-	是	属性对应的列的表头或列名。如果有表头或列名，请直接使用该名称。如果 CSV 文件没有表头，用 `[_c0, _c1, _c2]` 的形式表示第一列、第二列、第三列，以此类推。
`tags.nebula.fields`	list[string]	-	是	悦数图数据库中定义的属性名称，顺序必须和 `tags.fields` 一一对应。例如 `[_c1, _c2]` 对应 `[name, age]` ，表示第二列为属性 name 的值，第三列为属性 age 的值。
`tags.vertex.field`	string	-	是	点 ID 的列。例如 CSV 文件没有表头时，可以用 `_c0` 表示第一列的值作为点 ID。
`tags.vertex.udf.separator`	string	-	否	通过自定义规则合并多列，该参数指定连接符。
`tags.vertex.udf.oldColNames`	list	-	否	通过自定义规则合并多列，该参数指定待合并的列名。多个列用英文逗号（,）分隔。
`tags.vertex.udf.newColName`	string	-	否	通过自定义规则合并多列，该参数指定新列的列名。
`tags.vertex.prefix`	string	-	否	为 VID 增加指定的前缀。例如 VID 为 `12345` ，增加前缀 `tag1` 后为 `tag1_12345` 。下划线无法修改。
`tags.vertex.policy`	string	-	否	仅支持取值 `hash` 。对 string 类型的 VID 进行哈希化操作。
`tags.batch`	int	`256`	是	单批次写入悦数图数据库的最大点数量。
`tags.partition`	int	`32`	是	Spark 分片数量。

Parquet/JSON/ORC 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.path`	string	-	是	HDFS 中点数据文件的路径。用双引号括起路径，以 `hdfs://` 开头。

CSV 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.path`	string	-	是	HDFS 中点数据文件的路径。用双引号括起路径，以 `hdfs://` 开头。
`tags.separator`	string	`,`	是	分隔符。默认值为英文逗号（,）。对于特殊字符，如控制符 `^A` ,可以用 ASCII 八进制 `\001` 或 UNICODE 编码十六进制 `\u0001` 表示，控制符 `^B` ，用 ASCII 八进制 `\002` 或 UNICODE 编码十六进制 `\u0002` 表示，控制符 `^C` ，用 ASCII 八进制 `\003` 或 UNICODE 编码十六进制 `\u0003` 表示。
`tags.header`	bool	`true`	是	文件是否有表头。

Copyright © Yueshu Graph Technology - All rights reserved.

Hive 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.exec`	string	-	是	查询数据源的语句。例如 `select name,age from mooc.users` 。

MaxCompute 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.table`	string	-	是	MaxCompute 的表名。
`tags.project`	string	-	是	MaxCompute 的项目名。
`tags.odpsUrl`	string	-	是	MaxCompute 服务的 odpsUrl。地址可根据阿里云文档查看。
`tags.tunnelUrl`	string	-	是	MaxCompute 服务的 tunnelUrl。地址可根据阿里云文档查看。
`tags.accessKeyId`	string	-	是	MaxCompute 服务的 accessKeyId。
`tags.accessKeySecret`	string	-	是	MaxCompute 服务的 accessKeySecret。
`tags.partitionSpec`	string	-	否	MaxCompute 表的分区描述。
`tags.numPartitions`	int	`1`	否	MaxCompute 的 Spark 连接器在读取 MaxCompute 数据时使用的分区数。
`tags.sentence`	string	-	否	查询数据源的语句。SQL 语句中的表名和上方 table 的值相同。

Neo4j 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.exec`	string	-	是	查询数据源的语句。例如 `match (n:label) return n.neo4j-field-0` 。
`tags.server`	string	`"bolt://127.0.0.1:7687"`	是	Neo4j 服务器地址。
`tags.user`	string	-	是	拥有读取权限的 Neo4j 用户名。
`tags.password`	string	-	是	用户名对应密码。
`tags.database`	string	-	是	Neo4j 中保存源数据的数据库名。
`tags.check_point_path`	string	`/tmp/test`	否	设置保存导入进度信息的目录，用于断点续传。如果未设置，表示不启用断点续传。

Copyright © Yueshu Graph Technology - All rights reserved.

MySQL/PostgreSQL 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.host`	string	-	是	MySQL/PostgreSQL 服务器地址。
`tags.port`	string	-	是	MySQL/PostgreSQL 服务器端口。
`tags.database`	string	-	是	数据库名称。
`tags.table`	string	-	是	需要作为数据源的表名称。
`tags.user`	string	-	是	拥有读取权限的 MySQL/PostgreSQL 用户名。
`tags.password`	string	-	是	用户名对应密码。
`tags.sentence`	string	-	是	查询数据源的语句。例如 `"select teamid, name from team order by teamid"` 。

Oracle 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.url`	string	-	是	Oracle 数据库连接地址。
`tags.driver`	string	-	是	Oracle 驱动地址。
`tags.user`	string	-	是	拥有读取权限的 Oracle 用户名。
`tags.password`	string	-	是	用户名对应密码。
`tags.table`	string	-	是	需要作为数据源的表名称。
`tags.sentence`	string	-	是	查询数据源的语句。例如 `"select playerid, name, age from player"` 。

ClickHouse 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.url`	string	-	是	ClickHouse 的 JDBC URL。
`tags.user`	string	-	是	有读取权限的 ClickHouse 用户名。
`tags.password`	string	-	是	用户名对应密码。
`tags.numPartition`	string	-	是	ClickHouse 分区数。
`tags.sentence`	string	-	是	查询数据源的语句。

Hbase 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.host`	string	`127.0.0.1`	是	Hbase 服务器地址。
`tags.port`	string	`2181`	是	Hbase 服务器端口。
`tags.table`	string	-	是	需要作为数据源的表名称。
`tags.columnFamily`	string	-	是	表所属的列族（column family）。

Copyright © Yueshu Graph Technology - All rights reserved.

Pulsar 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.service`	string	`"pulsar://localhost:6650"`	是	Pulsar 服务器地址。
`tags.admin`	string	`"http://localhost:8081"`	是	连接 pulsar 的 admin.url。
`tags.options.<topic|topics|topicsPattern>`	string	-	是	Pulsar 的选项，可以从 `topic` 、 `topics` 和 `topicsPattern` 选择一个进行配置。
`tags.interval.seconds`	int	`10`	是	读取消息的间隔。单位：秒。

Kafka 源特有参数

参数	数据类型	默认值	是否必须	说明
`tags.service`	string	-	是	Kafka 服务器地址。
`tags.topic`	string	-	是	消息类别。
`tags.interval.seconds`	int	`10`	是	读取消息的间隔。单位：秒。

生成 **SST** 时的特有参数

参数	数据类型	默认值	是否必须	说明
`tags.path`	string	-	是	指定需要生成 SST 文件的源文件的路径。
`tags.repartitionWithNebula`	bool	`true`	否	生成 SST 文件时是否要基于悦数图数据库中图空间的 partition 进行数据重分区。开启该功能可减少 DOWNLOAD 和 INGEST SST 文件需要的时间。

悦数图数据库源特有参数

Enterpriseonly

悦数图数据库源特有参数用于导出悦数图数据库数据，仅企业版 Exchange 支持。

参数	数据类型	默认值	是否必须	说明
`tags.path`	string	`"hdfs://namenode:9000/path/vertex"`	是	指定 CSV 文件的存储路径。设置的路径必须不存在，Exchange 会自动创建该路径。存储到 HDFS 服务器时路径格式同默认值，例如 `"hdfs://192.168.8.177:9000/vertex/player"` 。存储到本地时路径格式为 `"file:///path/vertex"` ，例如 `"file:///home/nebula/vertex/player"` 。有多个 Tag 时必须为每个 Tag 设置不同的目录。
`tags.noField`	bool	`false`	是	当值为 `true` 时，仅导出 VID 而不导出属性数据；当值为 `false` 时导出 VID 和属性数据。
`tags.return.fields`	list	`[]`	是	指定要导出的属性。例如，要导出 `name` 和 `age` 属性，需将参数值设置为 `["name","age"]` 。该参数仅在 `tags.noField` 的值为 `false` 时生效。

边配置

对于不同的数据源，边的配置也有所不同，有很多通用参数，也有部分特有参数，配置时需要配置通用参数和不同数据源的特有参数。

边配置的不同数据源特有参数请参见上方点配置内的特有参数介绍，注意区分 tags 和 edges 即可。

Copyright © Yueshu Graph Technology - All rights reserved.

通用参数

参数	数据类型	默认值	是否必须	说明
`edges.name`	string	-	是	悦数图数据库中定义的 Edge type 名称。
`edges.type.source`	string	-	是	指定数据源。例如 `csv` 。
`edges.type.sink`	string	`client`	是	指定导入方式，可选值为 `client` 和 `SST` 。
`edges.writeMode`	string	`INSERT`	否	对数据的批量操作类型，包括批量导入、更新和删除。可选值为 `INSERT` 、 `UPDATE` 、 `DELETE` 。
`edges.fields`	list[string]	-	是	属性对应的列的表头或列名。如果有表头或列名，请直接使用该名称。如果 CSV 文件没有表头，用 `[_c0, _c1, _c2]` 的形式表示第一列、第二列、第三列，以此类推。
`edges.nebula.fields`	list[string]	-	是	悦数图数据库中定义的属性名称，顺序必须和 `edges.fields` 一一对应。例如 `[_c2, _c3]` 对应 `[start_year, end_year]` ，表示第三列为开始年份的值，第四列为结束年份的值。
`edges.source.field`	string	-	是	边的起始点的列。例如 `_c0` 表示第一列的值作为边的起始点。
`edges.source.prefix`	string	-	否	为 VID 增加指定的前缀。例如 VID 为 `12345` ，增加前缀 `tag1` 后为 `tag1_12345` 。下划线无法修改。
`edges.source.policy`	string	-	否	仅支持取值 `hash` 。对 string 类型的 VID 进行哈希化操作。
`edges.target.field`	string	-	是	边的目的点的列。例如 `_c1` 表示第二列的值作为边的目的点。
`edges.target.prefix`	string	-	否	为 VID 增加指定的前缀。例如 VID 为 `12345` ，增加前缀 `tag1` 后为 `tag1_12345` 。下划线无法修改。
`edges.target.policy`	string	-	否	仅支持取值 `hash` 。对 string 类型的 VID 进行哈希化操作。
`edges.ranking`	int	-	否	rank 值的列。没有指定时，默认所有 rank 值为 `0` 。
`edges.batch`	int	`256`	是	单批次写入悦数图数据库的最大边数量。
`edges.partition`	int	`32`	是	Spark 分片数量。

生成 **SST** 时的特有参数

参数	数据类型	默认值	是否必须	说明
`edges.path`	string	-	是	指定需要生成 SST 文件的源文件的路径。
`edges.repartitionWithNebula`	bool	`true`	否	生成 SST 文件时是否要基于悦数图数据库中图空间的 partition 进行数据重分区。开启该功能可减少 DOWNLOAD 和 INGEST SST 文件需要的时间。

Copyright © Yueshu Graph Technology - All rights reserved.

悦数图数据库源特有参数

参数	数据类型	默认值	是否必须	说明
`edges.path`	string	`"hdfs://namenode:9000/path/edge"`	是	指定 CSV 文件的存储路径。设置的路径必须不存在，Exchange 会自动创建该路径。存储到 HDFS 服务器时路径格式同默认值，例如 `"hdfs://192.168.8.177:9000/edge/follow"` 。存储到本地时路径格式为 `"file:///path/edge"` ，例如 `"file:///home/nebula/edge/follow"` 。有多个 Edge 时必须为每个 Edge 设置不同的目录。
`edges.noField`	bool	`false`	是	当值为 `true` 时，仅导出起始点 VID、目的点 VID 和 Rank，而不导出属性数据；当值为 `false` 时导出起始点 VID、目的点 VID、Rank 和属性数据。
`edges.return.fields`	list	`[]`	是	指定要导出的属性。例如，要导出 `start_year` 和 `end_year` 属性，需将参数值设置为 `["start_year","end_year"]` 。该参数仅在 `edges.noField` 的值为 `false` 时生效。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.3.4 使用 NebulaGraph Exchange

导入 **CSV** 文件数据

本文以一个示例说明如何使用 Exchange 将存储在 HDFS 或本地的 CSV 文件数据导入悦数图数据库。

如果要向悦数图数据库导入本地 CSV 文件，请参见 NebulaGraph Importer。

数据集

本文以 basketballplayer 数据集为例。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB
- Spark：2.4.7 单机版
- Hadoop：2.9.2 伪分布式部署
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。
- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 如果文件存储在 HDFS 上，需要确认 Hadoop 服务运行正常。
- 如果文件存储在本地且悦数图数据库是集群架构，需要在集群每台机器本地相同目录下放置文件。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 1：在悦数图数据库中创建 **Schema**

分析 CSV 文件中的数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用悦数图数据库Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 2：处理 **CSV** 文件

确认以下信息：

1. 处理 CSV 文件以满足 Schema 的要求。

Note

Exchange 支持上传有表头或者无表头的 CSV 文件。

2. 获取 CSV 文件存储路径。

步骤 3：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 CSV 数据源相关的配置。在本示例中，复制的文件名为 `csv_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    executor: {
        memory:1G
    }

    cores: {
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    max: 16
  }
}

# 悦数图数据库 相关配置
nebula: {
  address:{
    # 指定 Graph 服务和所有 Meta 服务的 IP 地址和端口。
    # 如果有多台服务器，地址之间用英文逗号（,）分隔。
    # 格式："ip1:port","ip2:port","ip3:port"
    graph:["127.0.0.1:9669"]
    #任意一个 Meta 服务的地址。
    #如果您的悦数图数据库在虚拟网络中，如k8s，请配置 Leader Meta的地址。
    meta:["127.0.0.1:9559"]
  }

  # 指定拥有悦数图数据库写权限的用户名和密码。
  user: root
  pswd: nebula

  # 指定图空间名称。
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  }
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}

# 处理点
tags: [
  # 设置 Tag player 相关信息。
  {
    # 指定悦数图数据库中定义的 Tag 名称。
    name: player
    type: {
      # 指定数据源，使用 CSV。
      source: csv

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 指定 CSV 文件的路径。
    # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
    # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.csv"。
    path: "hdfs://192.168.*.*:9000/data/vertex_player.csv"

    # 如果 CSV 文件没有表头，使用 [_c0, _c1, _c2, ..., _cn] 表示其表头，并将列指示为属性值的源。
    # 如果 CSV 文件有表头，则使用实际的列名。
    fields: [_c1, _c2]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [age, name]

    # 指定一个列作为 VID 的源。
    # vertex 的值必须与上述 fields 或者 csv.fields 中的列名保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    vertex: {
      field:_c0
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定的分隔符。默认值为英文逗号（,）。
    separator: ","

    # 如果 CSV 文件有表头，请将 header 设置为 true。
    # 如果 CSV 文件没有表头，请将 header 设置为 false。默认值为 false。
    header: false

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 指定单批次写入 悦数图数据库 的最大点数量。
    batch: 256

    # 指定 Spark 分片数量。
    partition: 32
  }

  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: csv
      sink: client
    }
    path: "hdfs://192.168.*.*:9000/data/vertex_team.csv"
    fields: [_c1]
    nebula.fields: [name]
    vertex: {
      field:_c0
    }
    separator: ","
    header: false
    batch: 256
    partition: 32
  }
  # 如果需要添加更多点，请参考前面的配置进行添加。
]
# 处理边
edges: [
  # 设置 Edge type follow 相关信息。
  {
    # 指定悦数图数据库中定义的 Edge type 名称。
    name: follow
    type: {
      # 指定数据源，使用 CSV。
      source: csv

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 指定 CSV 文件的路径。
    # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
    # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.csv"。
    path: "hdfs://192.168.*.*:9000/data/edge_follow.csv"

    # 如果 CSV 文件没有表头，使用 [_c0, _c1, _c2, ..., _cn] 表示其表头，并将列指示为属性值的源。
    # 如果 CSV 文件有表头，则使用实际的列名。
    fields: [_c2]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [degree]

    # 指定一个列作为起始点和目的点的源。
    # vertex 的值必须与上述 fields 或者 csv.fields 中的列名保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    source: {
      field: _c0
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }
    target: {
      field: _c1
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定的分隔符。默认值为英文逗号（,）。
    separator: ","

    # 指定一个列作为 rank 的源（可选）。

    #ranking: rank

    # 如果 CSV 文件有表头，请将 header 设置为 true。
    # 如果 CSV 文件没有表头，请将 header 设置为 false。默认值为 false。
    header: false
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 指定单批次写入 悦数图数据库 的最大边数量。
      batch: 256

      # 指定 Spark 分片数量。
      partition: 32
    }

    # 设置 Edge type serve 相关信息。
    {
      name: serve
      type: {
        source: csv
        sink: client
      }
      path: "hdfs://192.168.*.*:9000/data/edge_serve.csv"
      fields: [_c2,_c3]
      nebula.fields: [start_year, end_year]
      source: {
        field: _c0
      }
      target: {
        field: _c1
      }
      separator: ","
      header: false
      batch: 256
      partition: 32
    }

  ]
  # 如果需要添加更多边，请参考前面的配置进行添加。
}
```

步骤 **4**：向悦数图数据库导入数据

运行如下命令将 CSV 文件数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <csv_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/csv_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **JSON** 文件数据

本文以一个示例说明如何使用 Exchange 将存储在 HDFS 或本地的 JSON 文件数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。部分示例数据如下：

- player

```
{"id":"player100","age":42,"name":"Tim Duncan"}
{"id":"player101","age":36,"name":"Tony Parker"}
{"id":"player102","age":33,"name":"LaMarcus Aldridge"}
{"id":"player103","age":32,"name":"Rudy Gay"}
...
```

- team

```
{"id":"team200","name":"Warriors"}
{"id":"team201","name":"Nuggets"}
...
```

- follow

```
{"src":"player100","dst":"player101","degree":95}
{"src":"player101","dst":"player102","degree":90}
...
```

- serve

```
{"src":"player100","dst":"team204","start_year":"1997","end_year":"2016"}
{"src":"player101","dst":"team204","start_year":"1999","end_year":"2018"}
...
```

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.3.0，单机版

- Hadoop：2.9.2，伪分布式部署

- 悦数图数据库：3.5.1。

Copyright © Yueshu Graph Technology - All rights reserved.

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 如果文件存储在 HDFS 上，需要确认 Hadoop 服务运行正常。
- 如果文件存储在本地且悦数图数据库是集群架构，需要在集群每台机器本地相同目录下放置文件。

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析文件中的数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用 NebulaGraph Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：处理 **JSON** 文件

确认以下信息：

1. 处理 JSON 文件以满足 Schema 的要求。
2. 获取 JSON 文件存储路径。

Copyright © Yueshu Graph Technology - All rights reserved.

步骤 **3.** 修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 JSON 数据源相关的配置。在本示例中，复制的文件名为 `json_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    executor: {
        memory:1G
    }

    cores: {
      max: 16
    }
  }

  # 悦数图数据库 相关配置
  nebula: {
    address:{
      # 指定 Graph 服务和所有 Meta 服务的 IP 地址和端口。
      # 如果有多台服务器，地址之间用英文逗号（,）分隔。
      # 格式："ip1:port","ip2:port","ip3:port"
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }

    # 指定拥有 悦数图数据库 写权限的用户名和密码。
    user: root
    pswd: nebula

    # 指定图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
    execution: {
      retry: 3
    }
    error: {
      max: 32
      output: /tmp/errors
    }
    rate: {
      limit: 1024
      timeout: 1000
    }
  }

  # 处理点
  tags: [
    # 设置 Tag player 相关信息。
    {
      # 指定 悦数图数据库 中定义的 Tag 名称。
      name: player
      type: {
        # 指定数据源，使用 JSON。
        source: json

        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }

      # 指定 JSON 文件的路径。
      # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
      # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.json"。
      path: "hdfs://192.168.*.*:9000/data/vertex_player.json"

      # 在 fields 里指定 JSON 文件中 key 名称，其对应的 value 会作为 悦数图数据库 中指定属性的数据源。
      # 如果需要指定多个值，用英文逗号（,）隔开。
      fields: [age,name]

      # 指定 悦数图数据库 中定义的属性名称。
      # fields 与 nebula.fields 的顺序必须一一对应。
      nebula.fields: [age, name]

      # 指定一个列作为 VID 的源。
      # vertex 的值必须与 JSON 文件中的字段保持一致。
      # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
      vertex: {
        field:id
      # udf:{
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 指定单批次写入 悦数图数据库 的最大点数量。
      batch: 256

      # 指定 Spark 分片数量。
      partition: 32
    }

    # 设置 Tag team 相关信息。
    {
      name: team
      type: {
        source: json
        sink: client
      }
      path: "hdfs://192.168.*.*:9000/data/vertex_team.json"
      fields: [name]
      nebula.fields: [name]
      vertex: {
        field:id
      }
      batch: 256
      partition: 32
    }

    # 如果需要添加更多点，请参考前面的配置进行添加。
  ]
  # 处理边
  edges: [
    # 设置 Edge type follow 相关信息。
    {
      # 指定 悦数图数据库 中定义的 Edge type 名称。
      name: follow
      type: {
        # 指定数据源，使用 JSON。
        source: json

        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }

      # 指定 JSON 文件的路径。
      # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
      # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.json"。
      path: "hdfs://192.168.*.*:9000/data/edge_follow.json"

      # 在 fields 里指定 JSON 文件中 key 名称，其对应的 value 会作为 悦数图数据库 中指定属性的数据源。
      # 如果需要指定多个值，用英文逗号（,）隔开。
      fields: [degree]

      # 指定 悦数图数据库 中定义的属性名称。
      # fields 与 nebula.fields 的顺序必须一一对应。
      nebula.fields: [degree]

      # 指定一个列作为起始点和目的点的源。
      # vertex 的值必须与 JSON 文件中的字段保持一致。
      # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
      source: {
        field: src
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }
      target: {
        field: dst
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 指定单批次写入 悦数图数据库 的最大边数量。
      batch: 256

      # 指定 Spark 分片数量。
      partition: 32
    }

    # 设置 Edge type serve 相关信息。
    {
      name: serve
      type: {
        source: json
        sink: client
      }
      path: "hdfs://192.168.*.*:9000/data/edge_serve.json"
      fields: [start_year,end_year]
      nebula.fields: [start_year, end_year]
      source: {
        field: src
      }
      target: {
        field: dst
      }
      batch: 256
      partition: 32
    }

  ]
  # 如果需要添加更多边，请参考前面的配置进行添加。
}
```

步骤 **4**：向悦数图数据库导入数据

运行如下命令将 JSON 文件数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <json_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-echange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/json_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>` ，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **ORC** 文件数据

本文以一个示例说明如何使用 Exchange 将存储在 HDFS 或本地的 ORC 文件数据导入 NebulaGraph。

如果要向悦数图数据库导入本地 ORC 文件，请参见 NebulaGraph Importer。

数据集

本文以 basketballplayer 数据集为例。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7 单机版
- Hadoop：2.9.2 伪分布式部署
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 如果文件存储在 HDFS 上，需要确认 Hadoop 服务运行正常。
- 如果文件存储在本地且悦数图数据库是集群架构，需要在集群每台机器本地相同目录下放置文件。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 1：在悦数图数据库中创建 Schema

分析 ORC 文件中的数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用 NebulaGraph Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 2：处理 ORC 文件

确认以下信息：

1. 处理 ORC 文件以满足 Schema 的要求。
2. 获取 ORC 文件存储路径。

步骤 3：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 ORC 数据源相关的配置。在本示例中，复制的文件名为 `orc_application.conf`。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    executor: {
        memory:1G
    }

    cores: {
      max: 16
    }
  }

  # 悦数图数据库相关配置
  nebula: {
    address:{
      # 指定 Graph 服务和所有 Meta 服务的 IP 地址和端口。
      # 如果有多台服务器，地址之间用英文逗号（,）分隔。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 格式："ip1:port","ip2:port","ip3:port"
    graph:["127.0.0.1:9669"]
    #任意一个 Meta 服务的地址。
    #如果您的悦数图数据库在虚拟网络中，如k8s，请配置 Leader Meta的地址。
    meta:["127.0.0.1:9559"]
  }

  # 指定拥有悦数图数据库写权限的用户名和密码。
  user: root
  pswd: nebula

  # 指定图空间名称。
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  }
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}

# 处理点
tags: [
  # 设置 Tag player 相关信息。
  {
    # 指定悦数图数据库中定义的 Tag 名称。
    name: player
    type: {
      # 指定数据源，使用 ORC。
      source: orc

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 指定 ORC 文件的路径。
    # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
    # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.orc"。
    path: "hdfs://192.168.*.*:9000/data/vertex_player.orc"

    # 在 fields 里指定 ORC 文件中 key 名称，其对应的 value 会作为悦数图数据库中指定属性的数据源。
    # 如果需要指定多个值，用英文逗号（,）隔开。
    fields: [age,name]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [age, name]

    # 指定一个列作为 VID 的源。
    # vertex 的值必须与 ORC 文件中的字段保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    vertex: {
      field:id
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 指定单批次写入悦数图数据库的最大点数量。
    batch: 256

    # 指定 Spark 分片数量。
    partition: 32
  }

  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: orc
      sink: client
    }
    path: "hdfs://192.168.*.*:9000/data/vertex_team.orc"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    fields: [name]
    nebula.fields: [name]
    vertex: {
      field:id
    }
    batch: 256
    partition: 32
  }

  # 如果需要添加更多点，请参考前面的配置进行添加。
]
# 处理边
edges: [
  # 设置 Edge type follow 相关信息。
  {
    # 指定悦数图数据库中定义的 Edge type 名称。
    name: follow
    type: {
      # 指定数据源，使用 ORC。
      source: orc

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 指定 ORC 文件的路径。
    # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
    # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.orc"。
    path: "hdfs://192.168.*.*:9000/data/edge_follow.orc"

    # 在 fields 里指定 ORC 文件中 key 名称，其对应的 value 会作为悦数图数据库中指定属性的数据源。
    # 如果需要指定多个值，用英文逗号（,）隔开。
    fields: [degree]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [degree]

    # 指定一个列作为起始点和目的点的源。
    # vertex 的值必须与 ORC 文件中的字段保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    source: {
      field: src
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }
    target: {
      field: dst
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定一个列作为 rank 的源（可选）。
    #ranking: rank

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 指定单批次写入悦数图数据库的最大边数量。
    batch: 256

    # 指定 Spark 分片数量。
    partition: 32
  }

  # 设置 Edge type serve 相关信息。
  {
    name: serve
    type: {
      source: orc
      sink: client
    }
    path: "hdfs://192.168.*.*:9000/data/edge_serve.orc"
    fields: [start_year,end_year]
    nebula.fields: [start_year, end_year]
    source: {
      field: src
    }
    target: {
      field: dst
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      }
      batch: 256
      partition: 32
    }

  ]
  # 如果需要添加更多边，请参考前面的配置进行添加。
}
```

步骤 **4**: 向悦数图数据库导入数据

运行如下命令将 ORC 文件数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <orc_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/orc_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**: （可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **Parquet** 文件数据

本文以一个示例说明如何使用 Exchange 将存储在 HDFS 或本地的 Parquet 文件数据导入悦数图数据库。

如果要向悦数图数据库导入本地 Parquet 文件，请参见 NebulaGraph Importer。

数据集

本文以 basketballplayer 数据集为例。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7 单机版
- Hadoop：2.9.2 伪分布式部署
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 如果文件存储在 HDFS 上，需要确认 Hadoop 服务运行正常。
- 如果文件存储在本地且悦数图数据库是集群架构，需要在集群每台机器本地相同目录下放置文件。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析 Parquet 文件中的数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用 NebulaGraph Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：处理 **Parquet** 文件

确认以下信息：

1. 处理 Parquet 文件以满足 Schema 的要求。
2. 获取 Parquet 文件存储路径。

步骤 **3**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 Parquet 数据源相关的配置。在本示例中，复制的文件名为 `parquet_application.conf`。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    executor: {
        memory:1G
    }

    cores: {
      max: 16
    }
  }

  #悦数图数据库相关配置
  nebula: {
    address:{
      # 指定 Graph 服务和所有 Meta 服务的 IP 地址和端口。
      # 如果有多台服务器，地址之间用英文逗号（,）分隔。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 格式："ip1:port","ip2:port","ip3:port"
    graph:["127.0.0.1:9669"]
    #任意一个 Meta 服务的地址。
    #如果您的悦数图数据库在虚拟网络中，如k8s，请配置 Leader Meta的地址。
    meta:["127.0.0.1:9559"]
  }

  # 指定拥有悦数图数据库写权限的用户名和密码。
  user: root
  pswd: nebula

  # 指定图空间名称。
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  }
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}

# 处理点
tags: [
  # 设置 Tag player 相关信息。
  {
    # 指定悦数图数据库中定义的 Tag 名称。
    name: player
    type: {
      # 指定数据源，使用 Parquet。
      source: parquet

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 指定 Parquet 文件的路径。
    # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
    # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.csv"。
    path: "hdfs://192.168.11.139000/data/vertex_player.parquet"

    # 在 fields 里指定 Parquet 文件中 key 名称，其对应的 value 会作为悦数图数据库中指定属性的数据源。
    # 如果需要指定多个值，用英文逗号（,）隔开。
    fields: [age,name]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [age, name]

    # 指定一个列作为 VID 的源。
    # vertex 的值必须与 Parquet 文件中的字段保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    vertex: {
      field:id
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 指定单批次写入悦数图数据库的最大点数量。
    batch: 256

    # 指定 Spark 分片数量。
    partition: 32
  }

  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: parquet
      sink: client
    }
    path: "hdfs://192.168.11.13:9000/data/vertex_team.parquet"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    fields: [name]
    nebula.fields: [name]
    vertex: {
      field:id
    }
    batch: 256
    partition: 32
  }

  # 如果需要添加更多点，请参考前面的配置进行添加。
]
# 处理边
edges: [
  # 设置 Edge type follow 相关信息。
  {
    # 指定悦数图数据库中定义的 Edge type 名称。
    name: follow
    type: {
      # 指定数据源，使用 Parquet。
      source: parquet

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 指定 Parquet 文件的路径。
    # 如果文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx"。
    # 如果文件存储在本地，用双引号括起路径，以 file://开头，例如"file:///tmp/xx.csv"。
    path: "hdfs://192.168.11.13:9000/data/edge_follow.parquet"

    # 在 fields 里指定 Parquet 文件中 key 名称，其对应的 value 会作为悦数图数据库中指定属性的数据源。
    # 如果需要指定多个值，用英文逗号（,）隔开。
    fields: [degree]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [degree]

    # 指定一个列作为起始点和目的点的源。
    # vertex 的值必须与 Parquet 文件中的字段保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    source: {
      field: src
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }
    target: {
      field: dst
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定一个列作为 rank 的源（可选）。
    #ranking: rank

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 指定单批次写入悦数图数据库的最大边数量。
    batch: 256

    # 指定 Spark 分片数量。
    partition: 32
  }

  # 设置 Edge type serve 相关信息。
  {
    name: serve
    type: {
      source: parquet
      sink: client
    }
    path: "hdfs://192.168.11.13:9000/data/edge_serve.parquet"
    fields: [start_year,end_year]
    nebula.fields: [start_year, end_year]
    source: {
      field: src
    }
    target: {
      field: dst
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      }
      batch: 256
      partition: 32
    }

  ]
  # 如果需要添加更多边，请参考前面的配置进行添加。
}
```

步骤 **4**：向悦数图数据库导入数据

运行如下命令将 Parquet 文件数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <parquet_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/parquet_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **HBase** 数据

本文以一个示例说明如何使用 Exchange 将存储在 HBase 上的数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

在本示例中，该数据集已经存入 HBase 中，以 `player` 、 `team` 、 `follow` 和 `serve` 四个表存储了所有点和边的信息。以下为各个表的部分数据。

```
hbase(main):002:0> scan "player"
ROW                                    COLUMN+CELL
 player100                             column=cf:age, timestamp=1618881347530, value=42
 player100                             column=cf:name, timestamp=1618881354604, value=Tim Duncan
 player101                             column=cf:age, timestamp=1618881369124, value=36
 player101                             column=cf:name, timestamp=1618881379102, value=Tony Parker
 player102                             column=cf:age, timestamp=1618881386987, value=33
 player102                             column=cf:name, timestamp=1618881393370, value=LaMarcus Aldridge
 player103                             column=cf:age, timestamp=1618881402002, value=32
 player103                             column=cf:name, timestamp=1618881407882, value=Rudy Gay
 ...

hbase(main):003:0> scan "team"
ROW                                    COLUMN+CELL
 team200                               column=cf:name, timestamp=1618881445563, value=Warriors
 team201                               column=cf:name, timestamp=1618881453636, value=Nuggets
 ...

hbase(main):004:0> scan "follow"
ROW                                    COLUMN+CELL
 player100                             column=cf:degree, timestamp=1618881804853, value=95
 player100                             column=cf:dst_player, timestamp=1618881791522, value=player101
 player101                             column=cf:degree, timestamp=1618881824685, value=90
 player101                             column=cf:dst_player, timestamp=1618881816042, value=player102
 ...

hbase(main):005:0> scan "serve"
ROW                                    COLUMN+CELL
 player100                             column=cf:end_year, timestamp=1618881899333, value=2016
 player100                             column=cf:start_year, timestamp=1618881890117, value=1997
 player100                             column=cf:teamid, timestamp=1618881875739, value=team204
 ...
```

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7，单机版
- HBase：2.2.7
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 HBase 数据源相关的配置。在本示例中，复制的文件名为 `hbase_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

  # 悦数图数据库 相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}
# 处理点
tags: [
  # 设置 Tag player 相关信息。
  # 如果需要将 rowkey 设置为数据源，请填写“rowkey”，列族内的列请填写实际列名。
  {
    # 悦数图数据库 中对应的 Tag 名称。
    name: player
    type: {
      # 指定数据源文件格式，设置为 HBase。
      source: hbase
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }
    host:192.168.*.*
    port:2181
    table:"player"
    columnFamily:"cf"

    # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [age,name]
    nebula.fields: [age,name]

    # 指定表中某一列数据为 悦数图数据库 中点 VID 的来源。
    # 例如 rowkey 作为 VID 的来源，请填写“rowkey”。
    vertex:{
        field:rowkey
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 单批次写入 悦数图数据库 的数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }
  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: hbase
      sink: client
    }
    host:192.168.*.*
    port:2181
    table:"team"
    columnFamily:"cf"
    fields: [name]
    nebula.fields: [name]
    vertex:{
        field:rowkey
    }
    batch: 256
    partition: 32
  }

]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库 中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 HBase。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
        source: hbase

        # 指定边数据导入 悦数图数据库 的方式，
        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }

      host:192.168.*.*
      port:2181
      table:"follow"
      columnFamily:"cf"

      # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
      # fields 和 nebula.fields 里的配置必须一一对应。
      # 如果需要指定多个列名称，用英文逗号（,）隔开。
      fields: [degree]
      nebula.fields: [degree]

      # 在 source 里，将 follow 表中某一列作为边的起始点数据源。示例使用 rowkey。
      # 在 target 里，将 follow 表中某一列作为边的目的点数据源。示例使用列 dst_player。
      source:{
          field:rowkey
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      target:{
          field:dst_player
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 单批次写入 悦数图数据库 的数据条数。
      batch: 256

      # Spark 分区数量
      partition: 32
    }

    # 设置 Edge type serve 相关信息
    {
      name: serve
      type: {
        source: hbase
        sink: client
      }
      host:192.168.*.*
      port:2181
      table:"serve"
      columnFamily:"cf"

      fields: [start_year,end_year]
      nebula.fields: [start_year,end_year]
      source:{
          field:rowkey
      }

      target:{
          field:teamid
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 256
      partition: 32
    }
  ]
}
```


Copyright © Yueshu Graph Technology - All rights reserved.

步骤 **3**: 向悦数图数据库导入数据

运行如下命令将 HBase 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <hbase_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/hbase_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **4**: （可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **5**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **MySQL/PostgreSQL** 数据

本文以一个示例说明如何使用 Exchange 将存储在 MySQL 上的数据导入悦数图数据库，也适用于从 PostgreSQL 导出数据到悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

在本示例中，该数据集已经存入 MySQL 中名为 `basketball` 的数据库中，以 `player` 、 `team` 、 `follow` 和 `serve` 四个表存储了所有点和边的信息。以下为各个表的结构。

```
mysql> desc player;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| playerid | varchar(30) | YES  |     | NULL    |       |
| age      | int         | YES  |     | NULL    |       |
| name     | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+

mysql> desc team;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| teamid | varchar(30) | YES  |     | NULL    |       |
| name   | varchar(30) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+

mysql> desc follow;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| src_player | varchar(30) | YES  |     | NULL    |       |
| dst_player | varchar(30) | YES  |     | NULL    |       |
| degree     | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+

mysql> desc serve;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| playerid   | varchar(30) | YES  |     | NULL    |       |
| teamid     | varchar(30) | YES  |     | NULL    |       |
| start_year | int         | YES  |     | NULL    |       |
| end_year   | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB
- Spark：2.4.7，单机版
- MySQL： 8.0.23
- 悦数图数据库：3.5.1。

Copyright © Yueshu Graph Technology - All rights reserved.

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- mysql-connector-java-xxx.jar 已经下载并放置在 Spark 的 `SPARK_HOME/jars` 目录下。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 已经安装并开启 Hadoop 服务。

注意事项

nebula-exchange_spark_2.2 仅支持单表查询，不支持多表查询。

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 MySQL 数据源相关的配置。在本示例中，复制的文件名为 `mysql_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

# 悦数图数据库 相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
    execution: {
      retry: 3
    }
    error: {
      max: 32
      output: /tmp/errors
    }
    rate: {
      limit: 1024
      timeout: 1000
    }
  }
  # 处理点
  tags: [
    # 设置 Tag player 相关信息。
    {
      # 悦数图数据库 中对应的 Tag 名称。
      name: player
      type: {
        # 指定数据源文件格式，设置为 MySQL。
        source: mysql
        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }

      host:192.168.*.*
      port:3306
      user:"test"
      password:"123456"
      database:"basketball"

      # 扫描单个表读取数据。
      # nebula-exchange_spark_2.2 必须配置该参数。不支持配置 sentence。
      # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 sentence 同时配置。
      table:"basketball.player"

      # 通过查询语句读取数据。
      # nebula-exchange_spark_2.2 不支持该参数。
      # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 table 同时配置。支持多表查询。
      # sentence: "select * from  people, player, team"

      # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
      # fields 和 nebula.fields 里的配置必须一一对应。
      # 如果需要指定多个列名称，用英文逗号 (,) 隔开。
      fields: [age,name]
      nebula.fields: [age,name]

      # 指定表中某一列数据为 悦数图数据库 中点 VID 的来源。
      vertex: {
        field:playerid
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 单批次写入 悦数图数据库 的数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }
  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: mysql
      sink: client
    }

    host:192.168.*.*
    port:3306
    database:"basketball"
    table:"team"
    user:"test"
    password:"123456"
    sentence:"select teamid, name from team order by teamid"

    fields: [name]
    nebula.fields: [name]
    vertex: {
      field: teamid
    }
    batch: 256
    partition: 32
  }

]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库 中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 MySQL。
      source: mysql

      # 指定边数据导入 悦数图数据库 的方式，
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    host:192.168.*.*
    port:3306
    user:"test"
    password:"123456"
    database:"basketball"

    # 扫描单个表读取数据。
    # nebula-exchange_spark_2.2 必须配置该参数。不支持配置 sentence。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 sentence 同时配置。
    table:"basketball.follow"

    # 通过查询语句读取数据。
    # nebula-exchange_spark_2.2 不支持该参数。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 table 同时配置。支持多表查询。
    # sentence: "select * from  follow, serve"

    # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [degree]
    nebula.fields: [degree]

    # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
    # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
    source: {
      field: src_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    target: {
      field: dst_player
    # udf:{
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 单批次写入 悦数图数据库 的数据条数。
      batch: 256

      # Spark 分区数量
      partition: 32
    }

    # 设置 Edge type serve 相关信息
    {
      name: serve
      type: {
        source: mysql
        sink: client
      }

      host:192.168.*.*
      port:3306
      database:"basketball"
      table:"serve"
      user:"test"
      password:"123456"
      sentence:"select playerid,teamid,start_year,end_year from serve order by playerid"
      fields: [start_year,end_year]
      nebula.fields: [start_year,end_year]
      source: {
        field: playerid
      }
      target: {
        field: teamid
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 256
      partition: 32
    }
  ]
}
```

步骤 **3**：向悦数图数据库导入数据

运行如下命令将 MySQL 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <mysql_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/mysql_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **4**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

Copyright © Yueshu Graph Technology - All rights reserved.

步骤 **5**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **Oracle** 数据

本文以一个示例说明如何使用 Exchange 将存储在 Oracle 上的数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

在本示例中，该数据集已经存入 Oracle 中名为 `basketball` 的数据库中，以 `player` 、 `team` 、 `follow` 和 `serve` 四个表存储了所有点和边的信息。以下为各个表的结构。

```
oracle> desc player;
+-----------+-------+---------------+
| Column    | Null  | Type          |
+-----------+-------+---------------+
| PLAYERID  |  -    | VARCHAR2(30)  |
| NAME      |  -    | VARCHAR2(30)  |
| AGE       |  -    | NUMBER        |
+-----------+-------+---------------+

oracle> desc team;
+-----------+-------+---------------+
| Column    | Null  | Type          |
+-----------+-------+---------------+
| TEAMID    |  -    | VARCHAR2(30)  |
| NAME      |  -    | VARCHAR2(30)  |
+-----------+-------+---------------+

oracle> desc follow;
+-------------+-------+---------------+
| Column      | Null  | Type          |
+-------------+-------+---------------+
| SRC_PLAYER  |  -    | VARCHAR2(30)  |
| DST_PLAYER  |  -    | VARCHAR2(30)  |
| DEGREE      |  -    | NUMBER        |
+-------------+-------+---------------+

oracle> desc serve;
+------------+-------+---------------+
| Column     | Null  | Type          |
+------------+-------+---------------+
| PLAYERID   |  -    | VARCHAR2(30)  |
| TEAMID     |  -    | VARCHAR2(30)  |
| START_YEAR |  -    | NUMBER        |
| END_YEAR   |  -    | NUMBER        |
+------------+-------+---------------+
```

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB
- Spark：2.4.7，单机版
- 悦数图数据库：3.5.1。

Copyright © Yueshu Graph Technology - All rights reserved.

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建的 Schema 信息，包括 Tag 和 Edge type 的名称、属性等。
- 已经安装并开启 Hadoop 服务。

注意事项

nebula-exchange_spark_2.2 仅支持单表查询，不支持多表查询。

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 Oracle 数据源相关的配置。在本示例中，复制的文件名为 `oracle_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

# 悦数图数据库 相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
    execution: {
      retry: 3
    }
    error: {
      max: 32
      output: /tmp/errors
    }
    rate: {
      limit: 1024
      timeout: 1000
    }
  }
  # 处理点
  tags: [
    {
      # 悦数图数据库 中对应的 Tag 名称。
      name: player
      type: {
        # 指定数据源文件格式，设置为 Oracle。
        source: oracle
        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }


      url:"jdbc:oracle:thin:@host:1521:basketball"
      driver: "oracle.jdbc.driver.OracleDriver"
      user: "root"
      password: "123456"

      # 扫描单个表读取数据。
      # nebula-exchange_spark_2.2 必须配置该参数。不支持配置 sentence。
      # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 sentence 同时配置。
      table:"basketball.player"

      # 通过查询语句读取数据。
      # nebula-exchange_spark_2.2 不支持该参数。
      # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 table 同时配置。支持多表查询。
      # sentence: "select * from  people, player, team"

      # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为悦数图数据库中指定属性。
      # fields 和 nebula.fields 里的配置必须一一对应。
      # 如果需要指定多个列名称，用英文逗号 (,) 隔开。
      fields: [age,name]
      nebula.fields: [age,name]

      # 指定表中某一列数据为悦数图数据库中点 VID 的来源。
      vertex: {
        field:playerid
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
      #deleteEdge: false
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 单批次写入悦数图数据库的数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }
  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: oracle
      sink: client
    }

    url:"jdbc:oracle:thin:@host:1521:db"
    driver: "oracle.jdbc.driver.OracleDriver"
    user: "root"
    password: "123456"
    table: "basketball.team"
    sentence: "select teamid, name from team"

    fields: [name]
    nebula.fields: [name]
    vertex: {
      field: teamid
    }
    batch: 256
    partition: 32
  }

]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 Oracle。
      source: oracle

      # 指定边数据导入 悦数图数据库 的方式，
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    url:"jdbc:oracle:thin:@host:1521:db"
    driver: "oracle.jdbc.driver.OracleDriver"
    user: "root"
    password: "123456"

    # 扫描单个表读取数据。
    # nebula-exchange_spark_2.2 必须配置该参数。不支持配置 sentence。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 sentence 同时配置。
    table:"basketball.follow"

    # 通过查询语句读取数据。
    # nebula-exchange_spark_2.2 不支持该参数。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 table 同时配置。支持多表查询。
    # sentence: "select * from  follow, serve"

    # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为悦数图数据库中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [degree]
    nebula.fields: [degree]

    # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
    # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
    source: {
      field: src_player
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    target: {
      field: dst_player
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 单批次写入悦数图数据库的数据条数。
      batch: 256

      # Spark 分区数量
      partition: 32
    }

    # 设置 Edge type serve 相关信息
    {
      name: serve
      type: {
        source: oracle
        sink: client
      }

      url:"jdbc:oracle:thin:@host:1521:db"
      driver: "oracle.jdbc.driver.OracleDriver"
      user: "root"
      password: "123456"
      table: "basketball.serve"
      sentence: "select playerid, teamid, start_year, end_year from serve"

      fields: [start_year,end_year]
      nebula.fields: [start_year,end_year]
      source: {
        field: playerid
      }
      target: {
        field: teamid
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 256
      partition: 32
    }
  ]
}
```

步骤 **3**：向悦数图数据库导入数据

运行如下命令将 Oracle 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <oracle_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/oracle_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300`。

步骤 **4**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **5**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 ClickHouse 数据

本文以一个示例说明如何使用 Exchange 将存储在 ClickHouse 上的数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7，单机版
- ClickHouse：docker 部署 yandex/clickhouse-server tag: latest(2021.07.01)
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 ClickHouse 数据源相关的配置。在本示例中，复制的文件名为 `clickhouse_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: Nebula Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

# 悦数图数据库相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]

      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    }
    execution: {
      retry: 3
    }
    error: {
      max: 32
      output: /tmp/errors
    }
    rate: {
      limit: 1024
      timeout: 1000
    }
  }
  # 处理点
  tags: [
    # 设置 Tag player 相关信息。
    {
      name: player
      type: {
        # 指定数据源文件格式，设置为 ClickHouse。
        source: clickhouse
        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }

      # ClickHouse 的 JDBC URL
      url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"

      user:"user"
      password:"123456"

      # ClickHouse 分区数
      numPartition:"5"

      table:"player"
      sentence:"select * from player"

      # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为悦数图数据库中指定属性。
      # fields 和 nebula.fields 里的配置必须一一对应。
      # 如果需要指定多个列名称，用英文逗号（,）隔开。
      fields: [name,age]
      nebula.fields: [name,age]

      # 指定表中某一列数据为悦数图数据库中点 VID 的来源。
      vertex: {
        field:playerid
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
      #deleteEdge: false

      # 单批次写入悦数图数据库的数据条数。
      batch: 256

      # Spark 分区数量
      partition: 32
    }

    # 设置 Tag team 相关信息。
    {
      name: team
      type: {
        source: clickhouse
        sink: client
      }
      url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"
      user:"user"
      password:"123456"
      numPartition:"5"
      table:"team"
      sentence:"select * from team"
      fields: [name]
      nebula.fields: [name]
      vertex: {
        field:teamid
      }

      batch: 256
      partition: 32
    }
  ]
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 ClickHouse。
      source: clickhouse

      # 指定边数据导入悦数图数据库的方式，
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # ClickHouse 的 JDBC URL
    url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"

    user:"user"
    password:"123456"

    # ClickHouse 分区数
    numPartition:"5"

    table:"follow"
    sentence:"select * from follow"

    # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为悦数图数据库中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [degree]
    nebula.fields: [degree]

    # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
    source: {
      field:src_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
    target: {
      field:dst_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定一个列作为 rank 的源（可选）。
    #ranking: rank

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 单批次写入悦数图数据库的数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }

  # 设置 Edge type serve 相关信息
  {
    name: serve
    type: {
      source: clickhouse
      sink: client
    }
    url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"
    user:"user"
    password:"123456"
    numPartition:"5"
    sentence:"select * from serve"
    fields: [start_year,end_year]
    nebula.fields: [start_year,end_year]
    source: {
      field:playerid
    }
    target: {
      field:teamid
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 256
      partition: 32
    }
  ]
}
```

步骤 **3**：向悦数图数据库导入数据

运行如下命令将 ClickHouse 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <clickhouse_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/clickhouse_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **4**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **5**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **Neo4j** 数据

本文以一个示例说明如何使用 Exchange 将存储在 Neo4j 的数据导入悦数图数据库。

实现方法

Exchange 使用 **Neo4j Driver 4.0.1** 实现对 Neo4j 数据的读取。执行批量导出之前，用户需要在配置文件中写入针对标签（label）和关系类型（Relationship Type）自动执行的 Cypher 语句，以及 Spark 分区数，提高数据导出性能。

Exchange 读取 Neo4j 数据时需要完成以下工作：

1. Exchange 中的 Reader 会将配置文件中 `exec` 部分的 Cypher `RETURN` 语句后面的语句替换为 `COUNT(*)`，并执行这个语句，从而获取数据总量，再根据 Spark 分区数量计算每个分区的起始偏移量和大小。
2. （可选）如果用户配置了 `check_point_path` 目录，Reader 会读取目录中的文件。如果处于续传状态，Reader 会计算每个 Spark 分区应该有的偏移量和大小。
3. 在每个 Spark 分区里，Exchange 中的 Reader 会在 Cypher 语句后面添加不同的 `SKIP` 和 `LIMIT` 语句，调用 Neo4j Driver 并行执行，将数据分布到不同的 Spark 分区中。
4. Reader 最后将返回的数据处理成 DataFrame。

至此，Exchange 即完成了对 Neo4j 数据的导出。之后，数据被并行写入悦数图数据库数据库中。

整个过程如下图所示。

数据集

本文以 basketballplayer 数据集为例。

Copyright © Yueshu Graph Technology - All rights reserved.

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz
- CPU 内核数：14
- 内存：251 GB

- Spark：单机版，2.4.6 pre-build for Hadoop 2.7
- Neo4j：3.5.20 Community Edition
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。

操作步骤

步骤 1：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用 NebulaGraph Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```


Copyright © Yueshu Graph Technology - All rights reserved.

更多信息，请参见快速开始。

步骤 **2**：配置源数据

为了提高 Neo4j 数据的导出速度，在 Neo4j 数据库中为相应属性创建索引。详细信息，参考 Neo4j 用户手册。

步骤 **3**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置数据源相关的配置。在本示例中，复制的文件名为 `neo4j_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }

    driver: {
      cores: 1
      maxResultSize: 1G
    }

    executor: {
        memory:1G
    }

    cores:{
      max: 16
    }
  }

  # 悦数图数据库 相关配置
  nebula: {
    address:{
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    user: root
    pswd: nebula
    space: basketballplayer

    connection: {
      timeout: 3000
      retry: 3
    }

    execution: {
      retry: 3
    }

    error: {
      max: 32
      output: /tmp/errors
    }

    rate: {
      limit: 1024
      timeout: 1000
    }
  }

  # 处理点
  tags: [

    # 设置 Tag player 相关信息。
    {
      name: player
      type: {
        source: neo4j
        sink: client
      }
      server: "bolt://192.168.*.*:7687"
      user: neo4j
      password:neo4j
      database:neo4j
      exec: "match (n:player) return n.id as id, n.age as age, n.name as name"
      fields: [age,name]
      nebula.fields: [age,name]
      vertex: {
        field:id
      # udf:{
      #            separator:"_"
      #            oldColNames:[field-0,field-1,field-2]
      #            newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    partition: 10
    batch: 1000
    check_point_path: /tmp/test
 }
# 设置 Tag team 相关信息。
{
    name: team
    type: {
      source: neo4j
      sink: client
    }
    server: "bolt://192.168.*.*:7687"
    user: neo4j
    password:neo4j
    # bolt 3 does not support `select database`, please do not config database
    # database:neo4j
    exec: "match (n:team) return n.id as id,n.name as name order by id(n)"
    fields: [name]
    nebula.fields: [name]
    vertex: {
      field:id
    }
    partition: 10
    batch: 1000
    check_point_path: /tmp/test
 }
]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    name: follow
    type: {
      source: neo4j
      sink: client
    }
    server: "bolt://192.168.*.*:7687"
    user: neo4j
    password:neo4j
    # bolt 3 不支持多数据库，请勿配置数据库名。4 及以上可以配置数据库名。
    # database:neo4j
    exec: "match (a:player)-[r:follow]->(b:player) return a.id as src, b.id as dst, r.degree as degree  order by id(r)"
    fields: [degree]
    nebula.fields: [degree]
    source: {
      field: src
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }
    target: {
      field: dst
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }
    #ranking: rank
    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT
    partition: 10
    batch: 1000
    check_point_path: /tmp/test
  }
 # 设置 Edge type serve 相关信息
 {
    name: serve
    type: {
      source: neo4j
      sink: client
    }
    server: "bolt://192.168.*.*:7687"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      user: neo4j
      password:neo4j
      exec: "match (a:player)-[r:serve]->(b:team) return a.id as src, b.id as dst, r.start_year as start_year, r.end_year as end_year  order by id(r)"
      fields: [start_year,end_year]
      nebula.fields: [start_year,end_year]
      source: {
        field: src
      }
      target: {
        field: dst
      }
      partition: 10
      batch: 1000
      check_point_path: /tmp/test
    }
   ]
}
```

exec 配置说明

在配置 `tags.exec` 或者 `edges.exec` 参数时，需要填写 Cypher 查询语句。为了保证每次查询结果排序一致，并且为了防止在导入时丢失数据，强烈建议在 Cypher 查询语句中加入 `ORDER BY` 子句，同时，为了提高数据导入效率，最好选取有索引的属性作为排序的属性。如果没有索引，用户也可以观察默认的排序，选择合适的属性用于排序，以提高效率。如果默认的排序找不到规律，用户可以根据点或关系的 ID 进行排序，并且将 `partition` 设置为一个尽量小的值，减轻 Neo4j 的排序压力。

说明：使用 `ORDER BY` 子句会延长数据导入的时间。

另外，Exchange 需要在不同 Spark 分区执行不同 `SKIP` 和 `LIMIT` 的 Cypher 语句，所以在 `tags.exec` 和 `edges.exec` 对应的 Cypher 语句中不能含有 `SKIP` 和 `LIMIT` 子句。

tags.vertex 或 edges.vertex 配置说明

悦数图数据库在创建点和边时会将 ID 作为唯一主键，如果主键已存在则会覆盖该主键中的数据。所以，假如将某个 Neo4j 属性值作为悦数图数据库的 ID，而这个属性值在 Neo4j 中是有重复的，就会导致重复 ID，它们对应的数据有且只有一条会存入悦数图数据库中，其它的则会被覆盖掉。由于数据导入过程是并发地往悦数图数据库中写数据，最终保存的数据并不能保证是 Neo4j 中最新的数据。

check_point_path 配置说明

如果启用了断点续传功能，为避免数据丢失，在断点和续传之间，数据库不应该改变状态，例如不能添加数据或删除数据，同时，不能更改 `partition` 数量配置。

步骤 **4**：向悦数图数据库导入数据

运行如下命令将文件数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <neo4j_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/neo4j_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **Hive** 数据

本文以一个示例说明如何使用 Exchange 将存储在 Hive 上的数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

在本示例中，该数据集已经存入 Hive 中名为 `basketball` 的数据库中，以 `player` 、 `team` 、 `follow` 和 `serve` 四个表存储了所有点和边的信息。以下为各个表的结构。

```
scala> spark.sql("describe basketball.player").show
+--------+---------+-------+
|col_name|data_type|comment|
+--------+---------+-------+
|playerid|   string|   null|
|     age|   bigint|   null|
|    name|   string|   null|
+--------+---------+-------+

scala> spark.sql("describe basketball.team").show
+----------+---------+-------+
|  col_name|data_type|comment|
+----------+---------+-------+
|    teamid|   string|   null|
|      name|   string|   null|
+----------+---------+-------+

scala> spark.sql("describe basketball.follow").show
+----------+---------+-------+
|  col_name|data_type|comment|
+----------+---------+-------+
|src_player|   string|   null|
|dst_player|   string|   null|
|    degree|   bigint|   null|
+----------+---------+-------+

scala> spark.sql("describe basketball.serve").show
+----------+---------+-------+
|  col_name|data_type|comment|
+----------+---------+-------+
|  playerid|   string|   null|
|    teamid|   string|   null|
|start_year|   bigint|   null|
|  end_year|   bigint|   null|
+----------+---------+-------+
```

说明：Hive 的数据类型 `bigint` 与悦数图数据库的 `int` 对应。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB
- Spark：2.4.7，单机版
- Hive：2.3.7，Hive Metastore 数据库为 MySQL 8.0.22
- 悦数图数据库：3.5.1。

Copyright © Yueshu Graph Technology - All rights reserved.

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 已经启动 Hive Metastore 数据库（本示例中为 MySQL）。

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：使用 **Spark SQL** 确认 **Hive SQL** 语句

启动 spark-shell 环境后，依次运行以下语句，确认 Spark 能读取 Hive 中的数据。

```
scala> sql("select playerid, age, name from basketball.player").show
scala> sql("select teamid, name from basketball.team").show
scala> sql("select src_player, dst_player, degree from basketball.follow").show
scala> sql("select playerid, teamid, start_year, end_year from basketball.serve").show
```

以下为表 `basketball.player` 中读出的结果。

```
+---------+----+----------------+
| playerid| age|            name|
+---------+----+----------------+
|player100|  42|       Tim Duncan|
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
|player101|  36|      Tony Parker|
|player102|  33|LaMarcus Aldridge|
|player103|  32|         Rudy Gay|
|player104|  32|  Marco Belinelli|
+---------+----+-----------------+
...
```

步骤 3：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 Hive 数据源相关的配置。在本示例中，复制的文件名为 `hive_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

  # 如果 Spark 和 Hive 部署在不同集群，才需要配置连接 Hive 的参数，否则请忽略这些配置。
  #hive: {
  #  waredir: "hdfs://NAMENODE_IP:9000/apps/svr/hive-xxx/warehouse/"
  #  connectionURL: "jdbc:mysql://your_ip:3306/hive_spark?characterEncoding=UTF-8"
  #  connectionDriverName: "com.mysql.jdbc.Driver"
  #  connectionUserName: "user"
  #  connectionPassword: "password"
  #}

  # 悦数图数据库 相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和所有 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
    execution: {
      retry: 3
    }
    error: {
      max: 32
      output: /tmp/errors
    }
    rate: {
      limit: 1024
      timeout: 1000
    }
  }
  # 处理点
  tags: [
    # 设置 Tag player 相关信息。
    {
      # 悦数图数据库 中对应的 Tag 名称。
      name: player
      type: {
        # 指定数据源文件格式，设置为 hive。
        source: hive
        # 指定如何将点数据导入悦数图数据库：Client 或 SST。
        sink: client
      }

      # 设置读取数据库 basketball 中 player 表数据的 SQL 语句
      exec: "select playerid, age, name from basketball.player"

      # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
      # fields 和 nebula.fields 里的配置必须一一对应。
      # 如果需要指定多个列名称，用英文逗号 (,) 隔开。
      fields: [age,name]
      nebula.fields: [age,name]

      # 指定表中某一列数据为 悦数图数据库 中点 VID 的来源。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    vertex:{
      field:playerid
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 单批次写入 悦数图数据库 的最大数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }
  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: hive
      sink: client
    }
    exec: "select teamid, name from basketball.team"
    fields: [name]
    nebula.fields: [name]
    vertex: {
      field: teamid
    }
    batch: 256
    partition: 32
  }

]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库 中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 hive。
      source: hive

      # 指定边数据导入 悦数图数据库 的方式，
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # 设置读取数据库 basketball 中 follow 表数据的 SQL 语句。
    exec: "select src_player, dst_player, degree from basketball.follow"

    # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [degree]
    nebula.fields: [degree]

    # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
    # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
    source: {
      field: src_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    target: {
      field: dst_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # policy:hash
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 单批次写入 悦数图数据库 的最大数据条数。
      batch: 256

      # Spark 分区数量
      partition: 32
    }

    # 设置 Edge type serve 相关信息
    {
      name: serve
      type: {
        source: hive
        sink: client
      }
      exec: "select playerid, teamid, start_year, end_year from basketball.serve"
      fields: [start_year,end_year]
      nebula.fields: [start_year,end_year]
      source: {
        field: playerid
      }
      target: {
        field: teamid
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 256
      partition: 32
    }
  ]
}
```

步骤 **4**：向悦数图数据库导入数据

运行如下命令将 Hive 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <hive_application.conf_path> -h
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/hive_application.conf -h
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 MaxCompute 数据

本文以一个示例说明如何使用 Exchange 将存储在 MaxCompute 上的数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7，单机版
- MaxCompute：阿里云官方版本
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 MaxCompute 数据源相关的配置。在本示例中，复制的文件名为 `maxcompute_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

# 悦数图数据库 相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}
# 处理点
tags: [
  # 设置 Tag player 相关信息
  {
    name: player
    type: {
      # 指定数据源文件格式，设置为 MaxCompute。
      source: maxcompute
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # MaxCompute 的表名
    table:player

    # MaxCompute 的项目名
    project:project

    # MaxCompute 服务的 odpsUrl 和 tunnelUrl，
    # 地址可在 https://help.aliyun.com/document_detail/34951.html 查看。
    odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
    tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com"

    # MaxCompute 服务的 accessKeyId 和 accessKeySecret。
    accessKeyId:xxx
    accessKeySecret:xxx

    # MaxCompute 表的分区描述，该配置可选。
    partitionSpec:"dt='partition1'"

    # MaxCompute 的 Spark 连接器在读取 MaxCompute 数据时使用的分区数。默认为1，该配置可选。
    numPartitions:100

    # 请确保 SQL 语句中的表名和上方 table 的值相同，该配置可选。
    sentence:"select id, name, age, playerid from player where id < 10"

    # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields:[name, age]
    nebula.fields:[name, age]

    # 指定表中某一列数据为 悦数图数据库 中点 VID 的来源。
    vertex:{
      field: playerid
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 单批次写入 悦数图数据库 的数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }

  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: maxcompute
      sink: client
    }
    table:team
    project:project
    odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
    tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com"
    accessKeyId:xxx
    accessKeySecret:xxx
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    partitionSpec:"dt='partition1'"
    sentence:"select id, name, teamid from team where id < 10"
    fields:[name]
    nebula.fields:[name]
    vertex:{
      field: teamid
    }
    batch: 256
    partition: 32
  }
]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库 中对应的 Edge type 名称。
    name: follow

    type:{
      # 指定数据源文件格式，设置为 MaxCompute。
      source:maxcompute

      # 指定边数据导入 悦数图数据库 的方式，
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink:client
    }

    # MaxCompute 的表名
    table:follow

    # MaxCompute 的项目名
    project:project

    # MaxCompute 服务的 odpsUrl 和 tunnelUrl，
    # 地址可在 https://help.aliyun.com/document_detail/34951.html 查看。
    odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
    tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com"

    # MaxCompute 服务的 accessKeyId 和 accessKeySecret。
    accessKeyId:xxx
    accessKeySecret:xxx

    # MaxCompute 表的分区描述，该配置可选。
    partitionSpec:"dt='partition1'"

    # 请确保 SQL 语句中的表名和上方 table 的值相同，该配置可选。
    sentence:"select * from follow"

    # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields:[degree]
    nebula.fields:[degree]

    # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
    source:{
      field: src_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
    target:{
      field: dst_player
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定一个列作为 rank 的源（可选）。
    #ranking: rank

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # Spark 分区数量
    partition:10

    # 单批次写入 悦数图数据库 的数据条数。
    batch:10
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    }

    # 设置 Edge type serve 相关信息
    {
      name: serve
      type:{
        source:maxcompute
        sink:client
      }
      table:serve
      project:project
      odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
      tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com"
      accessKeyId:xxx
      accessKeySecret:xxx
      partitionSpec:"dt='partition1'"
      sentence:"select * from serve"
      fields:[start_year,end_year]
      nebula.fields:[start_year,end_year]
      source:{
        field: playerid
      }
      target:{
        field: teamid
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      partition:10
      batch:10
    }
  ]
}
```

步骤 **3**：向悦数图数据库导入数据

运行如下命令将 MaxCompute 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <maxcompute_application.conf_path>
```

> **Note**
>
> JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/maxcompute_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **4**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **5**：（如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **Pulsar** 数据

本文简单说明如何使用 Exchange 将存储在 Pulsar 上的数据导入悦数图数据库。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB
- Spark：2.4.7，单机版
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。
- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 已经安装并开启 Pulsar 服务。

注意事项

- 导入 Pulsar 数据时只支持 Client 模式，即参数 `tags.type.sink` 和 `edges.type.sink` 的值为 `client` 。
- 导入 Pulsar 数据时请勿使用 Exchange 3.4.0 版本，该版本增加了对导入数据的缓存，不支持流式数据导入。请使用 Exchange 3.0.0、3.3.0、3.5.0 版本。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 Pulsar 数据源相关的配置。在本示例中，复制的文件名为 `pulsar_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

  #悦数图数据库相关配置
  nebula: {
    address:{
      # 以下为悦数图数据库的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的悦数图数据库在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
    # 填写的账号必须拥有悦数图数据库相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写悦数图数据库中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}
# 处理点
tags: [
  # 设置 Tag player 相关信息。
  {
    #悦数图数据库中对应的 Tag 名称。
    name: player
    type: {
      # 指定数据源文件格式，设置为 Pulsar。
      source: pulsar
      # 指定如何将数据导入悦数图数据库。只支持 Client。
      sink: client
    }
    # Pulsar 服务器地址。
    service: "pulsar://127.0.0.1:6650"
    # 连接 pulsar 的 admin.url。
    admin: "http://127.0.0.1:8081"
    # Pulsar 的选项，可以从 topic、topics 和 topicsPattern 选择一个进行配置。
    options: {
      topics: "topic1,topic2"
    }

    # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为悦数图数据库中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [age,name]
    nebula.fields: [age,name]

    # 指定表中某一列数据为悦数图数据库中点 VID 的来源。
    vertex:{
        field:playerid
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 单批次写入悦数图数据库的数据条数。
    batch: 10

    # Spark 分区数量
    partition: 10
    # 读取消息的间隔。单位：秒。
    interval.seconds: 10
  }
  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: pulsar
      sink: client
    }
    service: "pulsar://127.0.0.1:6650"
    admin: "http://127.0.0.1:8081"
    options: {
      topics: "topic1,topic2"
    }
    fields: [name]
    nebula.fields: [name]
    vertex:{
        field:teamid
    }
    batch: 10
    partition: 10
    interval.seconds: 10
  }

]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  {
    #悦数图数据库中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 Pulsar。
      source: pulsar

      # 指定边数据导入悦数图数据库的方式，
      # 指定如何将数据导入悦数图数据库。只支持 Client。
      sink: client
    }

    # Pulsar 服务器地址。
    service: "pulsar://127.0.0.1:6650"
    # 连接 pulsar 的 admin.url。
    admin: "http://127.0.0.1:8081"
    # Pulsar 的选项，可以从 topic、topics 和 topicsPattern 选择一个进行配置。
    options: {
      topics: "topic1,topic2"
    }

    # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为悦数图数据库中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
    fields: [degree]
    nebula.fields: [degree]

    # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
    # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
    source:{
        field:src_player
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    target:{
        field:dst_player
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定一个列作为 rank 的源（可选）。
    #ranking: rank

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 单批次写入悦数图数据库的数据条数。
    batch: 10

    # Spark 分区数量
    partition: 10

    # 读取消息的间隔。单位：秒。
    interval.seconds: 10
  }

  # 设置 Edge type serve 相关信息
  {
    name: serve
    type: {
      source: Pulsar
      sink: client
    }
    service: "pulsar://127.0.0.1:6650"
    admin: "http://127.0.0.1:8081"
    options: {
      topics: "topic1,topic2"
    }

    fields: [start_year,end_year]
    nebula.fields: [start_year,end_year]
    source:{
        field:playerid
    }

    target:{
        field:teamid
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 10
      partition: 10
      interval.seconds: 10
    }
  ]
}
```

步骤 **3**: 向悦数图数据库导入数据

运行如下命令将 Pulsar 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <pulsar_application.conf_path>
```

> **Note**
>
> JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/pulsar_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **4**: （可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **5**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **Kafka** 数据

本文简单说明如何使用 Exchange 将存储在 Kafka 上的数据导入悦数图数据库。

> **Compatibility**
>
> 导入 Kafka 数据时请使用 Exchange 3.5.0/3.3.0/3.0.0 版本。3.4.0 版本增加了对导入数据的缓存，不支持流式数据导入。

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7，单机版

- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。

- 已经安装 Spark。

- 以下 JAR 包已经下载并放置在 Spark 的 `SPARK_HOME/jars` 目录下：
- spark-streaming-kafka_xxx.jar
- spark-sql-kafka-0-10_xxx.jar
- kafka-clients-xxx.jar

- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。

- 已经安装并开启 Kafka 服务。

注意事项

- 导入 Kafka 数据时只支持 Client 模式，即参数 `tags.type.sink` 和 `edges.type.sink` 的值为 `client`。

- 导入 Kafka 数据时请勿使用 Exchange 3.4.0 版本，该版本增加了对导入数据的缓存，不支持流式数据导入。请使用 Exchange 3.0.0、3.3.0、3.5.0 版本。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 在悦数图数据库中创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 Kafka 数据源相关的配置。在本示例中，复制的文件名为 `kafka_application.conf` 。各个配置项的详细说明请参见配置说明。

> **Note**
>
> 导入 Kafka 数据时，一个配置文件只能处理一个 Tag 或 Edge type。如果有多个 Tag 或 Edge type，需要创建多个配置文件。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
    driver: {
      cores: 1
      maxResultSize: 1G
    }
    cores: {
      max: 16
    }
  }

  # 悦数图数据库 相关配置
  nebula: {
    address:{
      # 以下为 悦数图数据库 的 Graph 服务和 Meta 服务所在机器的 IP 地址及端口。
      # 如果有多个地址，格式为 "ip1:port","ip2:port","ip3:port"。
      # 不同地址之间以英文逗号 (,) 隔开。
      graph:["127.0.0.1:9669"]
      #任意一个 Meta 服务的地址。
      #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
      meta:["127.0.0.1:9559"]
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 填写的账号必须拥有 悦数图数据库 相应图空间的写数据权限。
    user: root
    pswd: nebula
    # 填写 悦数图数据库 中需要写入数据的图空间名称。
    space: basketballplayer
    connection: {
      timeout: 3000
      retry: 3
    }
    execution: {
      retry: 3
    }
    error: {
      max: 32
      output: /tmp/errors
    }
    rate: {
      limit: 1024
      timeout: 1000
    }
  }
  # 处理点
  tags: [
    # 设置 Tag player 相关信息。
    {
      # 悦数图数据库 中对应的 Tag 名称。
      name: player
      type: {
        # 指定数据源文件格式，设置为 Kafka。
        source: kafka
        # 指定如何将数据导入悦数图数据库。只支持 Client。
        sink: client
      }
      # Kafka 服务器地址。
      service: "127.0.0.1:9092"
      # 消息类别。
      topic: "topic_name1"

      # 在 fields 里指定 Kafka value 中的字段名称，多个字段用英文逗号（,）隔开。Spark Structured Streaming 读取 Kafka 数据后会将其以 JSON 格式存储于 value 字段中，而这里的 fields 要配置
JSON 的 key 名。示例如下：
      fields: [personName, personAge]
      # 设置与 fields 中的 key 对应的 悦数图数据库 属性名，key 的 value 将保存为相应的属性值。下方设置会将 personName 的 value 保存到 悦数图数据库 中的 name 属性，personAge 的 value 则保存
到 age 属性。
      nebula.fields: [name, age]

      # 指定表中某一列数据为 悦数图数据库 中点 VID 的来源。
      # 这里的值 key 和上面的 key 重复，表示 key 既作为 VID，也作为属性 name。
      vertex:{
          field:personId
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 单批次写入 悦数图数据库 的数据条数。
      batch: 10

      # Spark 分区数量
      partition: 10
      # 读取消息的间隔。单位：秒。
      interval.seconds: 10
      # 消费起点，默认值 latest。 可选 latest、earliest
      startingOffsets: latest
      # 流控，对每个触发区间处理的最大偏移量的速率限制，可不配置。
      # maxOffsetsPerTrigger:10000
    }
  ]
  # 处理边数据
  #edges: [
  #  # 设置 Edge type follow 相关信息
  #  {
  #    # 悦数图数据库 中对应的 Edge type 名称。
  #    name: follow

  #    type: {
  #      # 指定数据源文件格式，设置为 Kafka。
  #      source: kafka

  #      # 指定边数据导入 悦数图数据库 的方式，
  #      # 指定如何将数据导入悦数图数据库。只支持 Client。
  #      sink: client
  #    }

  #    # Kafka 服务器地址。
  #    service: "127.0.0.1:9092"
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  #    # 消息类别。
  #    topic: "topic_name3"

  #    # 在 fields 里指定 Kafka value 中的字段名称，多个字段用英文逗号（,）隔开。Spark Structured Streaming 读取 Kafka 数据后会将其以 JSON 格式存储于 value 字段中，而这里的 fields 要配置
JSON 的 key 名。示例如下：
  #    fields: [degree]
  #    # 设置与 fields 中的 key 对应的 悦数图数据库 属性名，key 的 value 将保存为相应的属性值。下方设置会将 degree 的 value 保存到 悦数图数据库 中的 degree 属性。
  #    nebula.fields: [degree]

  #    # 在 source 里，将 topic 中某一列作为边的起始点数据源。
  #    # 在 target 里，将 topic 中某一列作为边的目的点数据源。
  #    source:{
  #        field:srcPersonId
  #    # udf:{
  #    #            separator:"_"
  #    #            oldColNames:[field-0,field-1,field-2]
  #    #            newColName:new-field
  #    #        }
  #    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
  #    # prefix:"tag1"
  #    # 对 string 类型的 VID 进行哈希化操作。
  #    # policy:hash
  #    }

  #    target:{
  #        field:dstPersonId
  #    # udf:{
  #    #            separator:"_"
  #    #            oldColNames:[field-0,field-1,field-2]
  #    #            newColName:new-field
  #    #        }
  #    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
  #    # prefix:"tag1"
  #    # 对 string 类型的 VID 进行哈希化操作。
  #    # policy:hash
  #    }

  #    # 指定一个列作为 rank 的源（可选）。
  #    #ranking: rank

  #    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
  #    #writeMode: INSERT

  #    # 单批次写入 悦数图数据库 的数据条数。
  #    batch: 10

  #    # Spark 分区数量
  #    partition: 10

  #    # 读取消息的间隔。单位：秒。
  #    interval.seconds: 10
  #    # 消费起点，默认值 latest。 可选 latest、earliest
  #    startingOffsets: latest
  #    # 流控，对每个触发区间处理的最大偏移量的速率限制，可不配置。
  #    # maxOffsetsPerTrigger:10000
  #  }
  #]
}
```

步骤 **3**：向悦数图数据库导入数据

运行如下命令将 Kafka 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <kafka_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-
exchange/nebula-exchange/target/classes/kafka_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **4**：（可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```


Copyright © Yueshu Graph Technology - All rights reserved.

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **5**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入通用 **JDBC** 数据

JDBC 数据是指用 JDBC 接口访问的各类数据库的数据的统称。本文以 MySQL 数据库为例说明如何使用 Exchange 将 JDBC 数据导入悦数图数据库。

数据集

本文以 basketballplayer 数据集为例。

在本示例中，该数据集已经存入 MySQL 中名为 `basketball` 的数据库中，以 `player` 、 `team` 、 `follow` 和 `serve` 四个表存储了所有点和边的信息。以下为各个表的结构。

```
mysql> desc player;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| playerid | int         | YES  |     | NULL    |       |
| age      | int         | YES  |     | NULL    |       |
| name     | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+

mysql> desc team;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| teamid | int         | YES  |     | NULL    |       |
| name   | varchar(30) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+

mysql> desc follow;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| src_player | int         | YES  |     | NULL    |       |
| dst_player | int         | YES  |     | NULL    |       |
| degree     | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+

mysql> desc serve;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| playerid   | int         | YES  |     | NULL    |       |
| teamid     | int         | YES  |     | NULL    |       |
| start_year | int         | YES  |     | NULL    |       |
| end_year   | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB
- Spark：2.3.0，单机版
- 悦数图数据库：3.5.1。

Copyright © Yueshu Graph Technology - All rights reserved.

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。

- 已经编译 Exchange。详情请参见编译 Exchange。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 了解悦数图数据库中创建 Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。
- 如果文件存储在本地且悦数图数据库是集群架构，需要在集群每台机器本地相同目录下放置文件。

注意事项

nebula-exchange_spark_2.2 仅支持单表查询，不支持多表查询。

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析文件中的数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用 NebulaGraph Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2.** 修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置 JDBC 数据源相关的配置。在本示例中，复制的文件名为 `jdbc_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  driver: {
    cores: 1
    maxResultSize: 1G
  }
  executor: {
      memory:1G
  }

  cores: {
    max: 16
  }
}

# 悦数图数据库 相关配置
nebula: {
  address:{
    # 指定 Graph 服务和所有 Meta 服务的 IP 地址和端口。
    # 如果有多台服务器，地址之间用英文逗号（,）分隔。
    # 格式："ip1:port","ip2:port","ip3:port"
    graph:["127.0.0.1:9669"]
    #任意一个 Meta 服务的地址。
    #如果您的 悦数图数据库 在虚拟网络中，如k8s，请配置 Leader Meta的地址。
    meta:["127.0.0.1:9559"]
  }

  # 指定拥有 悦数图数据库 写权限的用户名和密码。
  user: root
  pswd: nebula

  # 指定图空间名称。
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  }
  execution: {
    retry: 3
  }
  error: {
    max: 32
    output: /tmp/errors
  }
  rate: {
    limit: 1024
    timeout: 1000
  }
}

# 处理点
tags: [
  # 设置 Tag player 相关信息。
  {
    # 指定 悦数图数据库 中定义的 Tag 名称。
    name: player
    type: {
      # 指定数据源，使用 JDBC。
      source: jdbc

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    # JDBC 数据源的 URL。示例为 MySql 数据库。
    url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true&characterEncoding=utf-8"

    # JDBC 驱动。
    driver:"com.mysql.cj.jdbc.Driver"

    # 数据库用户名和密码。
    user:"root"
    password:"12345"

    # 扫描单个表读取数据。
    # nebula-exchange_spark_2.2 必须配置该参数，还可以额外配置 sentence。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 sentence 同时配置。
    table:"basketball.player"

    # 通过查询语句读取数据。
    # nebula-exchange_spark_2.2 可以配置该参数。不支持多表查询。在 from 后只需要写表名，不支持`库名.表名`。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 table 同时配置。支持多表查询。
    # sentence:"select playerid, age, name from player, team order by playerid"

    #  （可选）多连接读取参数 参见 https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html
    partitionColumn:playerid    # 可选。数值类型必须为数字、日期或时间戳。
    lowerBound:1                # 可选
    upperBound:5                # 可选
    numPartitions:5             # 可选


    fetchSize:2           # 每次请求数据库要读取的行数。

    # 在 fields 里指定 player 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
    # fields 和 nebula.fields 里的配置必须一一对应。
    # 如果需要指定多个列名称，用英文逗号（,）隔开。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    fields: [age,name]
    nebula.fields: [age,name]

    # 指定表中某一列数据为 悦数图数据库 中点 VID 的来源。
    vertex: {
      field:playerid
    # udf:{
    #           separator:"_"
    #           oldColNames:[field-0,field-1,field-2]
    #           newColName:new-field
    #        }
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 批量删除时是否删除该点关联的出边和入边。`writeMode`为`DELETE`时该参数生效。
    #deleteEdge: false

    # 单批次写入 悦数图数据库 的数据条数。
    batch: 256

    # Spark 分区数量
    partition: 32
  }
  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: jdbc
      sink: client
    }


    url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true&characterEncoding=utf-8"
    driver:"com.mysql.cj.jdbc.Driver"
    user:root
    password:"12345"
    table:team
    sentence:"select teamid, name from team order by teamid"
    partitionColumn:teamid
    lowerBound:1
    upperBound:5
    numPartitions:5
    fetchSize:2

    fields: [name]
    nebula.fields: [name]
    vertex: {
      field: teamid
    }
    batch: 256
    partition: 32
  }

]

# 处理边数据
edges: [
  # 设置 Edge type follow 相关信息
  {
    # 悦数图数据库 中对应的 Edge type 名称。
    name: follow

    type: {
      # 指定数据源文件格式，设置为 JDBC。
      source: jdbc

      # 指定边数据导入 悦数图数据库 的方式，
      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: client
    }

    url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true&characterEncoding=utf-8"
    driver:"com.mysql.cj.jdbc.Driver"
    user:root
    password:"12345"

    # 扫描单个表读取数据。
    # nebula-exchange_spark_2.2 必须配置该参数，还可以额外配置 sentence。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 sentence 同时配置。
    table:"basketball.follow"

    # 通过查询语句读取数据。
    # nebula-exchange_spark_2.2 可以配置该参数。不支持多表查询。在 from 后只需要写表名，不支持`库名.表名`。
    # nebula-exchange_spark_2.4 和 nebula-exchange_spark_3.0 可以配置该参数，但是不能和 table 同时配置。支持多表查询。
    # sentence:"select src_player,dst_player,degree from follow order by src_player"

    partitionColumn:src_player
    lowerBound:1
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      upperBound:5
      numPartitions:5
      fetchSize:2

      # 在 fields 里指定 follow 表中的列名称，其对应的 value 会作为 悦数图数据库 中指定属性。
      # fields 和 nebula.fields 里的配置必须一一对应。
      # 如果需要指定多个列名称，用英文逗号（,）隔开。
      fields: [degree]
      nebula.fields: [degree]

      # 在 source 里，将 follow 表中某一列作为边的起始点数据源。
      # 在 target 里，将 follow 表中某一列作为边的目的点数据源。
      source: {
        field: src_player
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      target: {
        field: dst_player
      # udf:{
      #           separator:"_"
      #           oldColNames:[field-0,field-1,field-2]
      #           newColName:new-field
      #        }
      # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
      # prefix:"tag1"
      # 对 string 类型的 VID 进行哈希化操作。
      # policy:hash
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 单批次写入 悦数图数据库 的数据条数。
      batch: 256

      # Spark 分区数量
      partition: 32
    }

    # 设置 Edge type serve 相关信息
    {
      name: serve
      type: {
        source: jdbc
        sink: client
      }

      url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true&characterEncoding=utf-8"
      driver:"com.mysql.cj.jdbc.Driver"
      user:root
      password:"12345"
      table:serve
      sentence:"select playerid,teamid,start_year,end_year from serve order by playerid"
      partitionColumn:playerid
      lowerBound:1
      upperBound:5
      numPartitions:5
      fetchSize:2

      fields: [start_year,end_year]
      nebula.fields: [start_year,end_year]
      source: {
        field: playerid
      }
      target: {
        field: teamid
      }

      # 指定一个列作为 rank 的源（可选）。
      #ranking: rank

      batch: 256
      partition: 32
    }
  ]
}
```

步骤 **4**：向悦数图数据库导入数据

运行如下命令将 JDBC 数据导入到悦数图数据库中。关于参数的说明，请参见导入命令参数。

Copyright © Yueshu Graph Technology - All rights reserved.

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c <jdbc_application.conf_path>
```

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --class com.vesoft.nebula.exchange.Exchange  /root/nebula-echange/nebula-exchange/target/nebula-exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/jdbc_application.conf
```

用户可以在返回信息中搜索 `batchSuccess.<tag_name/edge_name>`，确认成功的数量。例如 `batchSuccess.follow: 300` 。

步骤 **5**: （可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **6**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导入 **SST** 文件数据

本文以一个示例说明如何将数据源的数据生成 SST（Sorted String Table）文件并保存在 HDFS 上，然后导入悦数图数据库，示例数据源是 CSV 文件。

注意事项

- 仅 Linux 系统支持导入 SST 文件。
- 不支持属性的 Default 值。
- 企业版 Exchange 3.5.0不支持基于 GEOGRAPHY 类型的数据生成 SST 文件。

背景信息

Exchange 支持两种数据导入模式：

- 直接将数据源的数据通过 **nGQL** 语句的形式导入悦数图数据库。
- 将数据源的数据生成 SST 文件，然后借助 Console 将 SST 文件导入悦数图数据库。

下文将介绍生成 SST 文件并用其导入数据的适用场景、实现方法、前提条件、操作步骤等内容。

适用场景

- 适合在线业务，因为生成时几乎不会影响业务（只是读取 Schema），导入速度快。

  Caution

  导入期间（大约 10 秒）会阻塞对应空间的写操作，并且之后数小时内可能有历史数据整理，建议在业务低峰期进行导入。

- 适合数据源数据量较大的场景，导入速度快。

实现方法

悦数图数据库底层使用 RocksDB 作为键值型存储引擎。RocksDB 是基于硬盘的存储引擎，提供了一系列 API 用于创建及导入 SST 格式的文件，有助于快速导入海量数据。

SST 文件是一个内部包含了任意长度的有序键值对集合的文件，用于高效地存储大量键值型数据。生成 SST 文件的整个过程主要由 Exchange 的 Reader、sstProcessor 和 sstWriter 完成。整个数据处理过程如下：

1. Reader 从数据源中读取数据。
2. sstProcessor 根据悦数图数据库的 Schema 信息生成 SST 文件，然后上传至 HDFS。SST 文件的格式请参见数据存储格式。
3. sstWriter 打开一个文件并插入数据。生成 SST 文件时，Key 必须按照顺序写入。
4. 生成 SST 文件之后，RocksDB 通过 `IngestExternalFile()` 方法将 SST 文件导入到悦数图数据库中。例如：

```
IngestExternalFileOptions ifo;
# 导入两个 SST 文件
Status s = db_->IngestExternalFile({"/home/usr/file1.sst", "/home/usr/file2.sst"}, ifo);
if (!s.ok()) {
  printf("Error while adding file %s and %s, Error %s\n",
         file_path1.c_str(), file_path2.c_str(), s.ToString().c_str());
  return 1;
}
```

调用 `IngestExternalFile()` 方法时，RocksDB 默认会将文件拷贝到数据目录，并且阻塞 RocksDB 写入操作。如果 SST 文件中的键范围覆盖了 Memtable 键的范围，则将 Memtable 落盘（flush）到硬盘。将 SST 文件放置在 LSM 树最优位置后，为文件分配一个全局序列号，并打开写操作。

数据集

本文以 basketballplayer 数据集为例。

Copyright © Yueshu Graph Technology - All rights reserved.

环境配置

本文示例在 MacOS 下完成，以下是相关的环境配置信息：

- 硬件规格：
- CPU：1.7 GHz Quad-Core Intel Core i7
- 内存：16 GB

- Spark：2.4.7 单机版
- Hadoop：2.9.2 伪分布式部署
- 悦数图数据库：3.5.1。

前提条件

开始导入数据之前，用户需要确认以下信息：

- 已经安装部署悦数图数据库 3.5.1 并获取如下信息：
- Graph 服务和 Meta 服务的的 IP 地址和端口。
- 拥有悦数图数据库写权限的用户名和密码。
- Meta 服务配置文件中的 `--ws_storage_http_port` 和 Storage 服务配置文件中的 `--ws_http_port` 一致。例如都为 `19779`。
- Graph 服务配置文件中的 `--ws_meta_http_port` 和 Meta 服务配置文件中的 `--ws_http_port` 一致。例如都为 `19559`。
- Schema 的信息，包括 Tag 和 Edge type 的名称、属性等。

- 已经编译 Exchange，或者直接下载编译完成的 jar 文件。本示例中使用 Exchange 3.5.0。
- 已经安装 Spark。
- 已经安装 JDK 1.8 或以上版本，并配置环境变量 JAVA_HOME。
- 确认 Hadoop 服务在所有部署 Storage 服务的机器上运行正常。

Note

- 如果需要生成其他数据源的 SST 文件，请参见相应数据源的文档，查看前提条件部分。
- 如果只需要生成 SST 文件，不需要在部署 Storage 服务的机器上安装 Hadoop 服务。
- 如需在 INGEST（数据导入）结束后自动移除 SST 文件，在 Storage 服务配置文件中增加 `--move_files=true`，该配置会让悦数图数据库在 INGEST 后将 SST 文件移动（`mv`）到 `data` 目录下。`--move_files` 的默认值为 `false`，此时悦数图数据库会复制（`cp`）SST 文件而不是移动。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

步骤 **1**：在悦数图数据库中创建 **Schema**

分析 CSV 文件中的数据，按以下步骤在悦数图数据库中创建 Schema：

1. 确认 Schema 要素。悦数图数据库中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge Type	`follow`	`degree int`
Edge Type	`serve`	`start_year int, end_year int`

2. 使用 NebulaGraph Console 创建一个图空间 **basketballplayer**，并创建一个 Schema，如下所示。

```
## 创建图空间
nebula> CREATE SPACE basketballplayer \
        (partition_num = 10, \
        replica_factor = 1, \
        vid_type = FIXED_STRING(30));

## 选择图空间 basketballplayer
nebula> USE basketballplayer;

## 创建 Tag player
nebula> CREATE TAG player(name string, age int);

## 创建 Tag team
nebula> CREATE TAG team(name string);

## 创建 Edge type follow
nebula> CREATE EDGE follow(degree int);

## 创建 Edge type serve
nebula> CREATE EDGE serve(start_year int, end_year int);
```

更多信息，请参见快速开始。

步骤 **2**：处理 **CSV** 文件

确认以下信息：

1. 处理 CSV 文件以满足 Schema 的要求。

Note

可以使用有表头或者无表头的 CSV 文件。

2. 获取 CSV 文件存储路径。

步骤 **3**：修改配置文件

编译 Exchange 后，复制 `target/classes/application.conf` 文件设置相关配置。在本示例中，复制的文件名为 `sst_application.conf` 。各个配置项的详细说明请参见配置说明。

```
{
  # Spark 相关配置
  spark: {
    app: {
      name: NebulaGraph Exchange 3.5.0
    }

    master:local

    driver: {
      cores: 1
      maxResultSize: 1G
    }

    executor: {
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      memory:1G
  }

  cores:{
    max: 16
  }
}

#悦数图数据库相关配置
nebula: {
  address:{
    graph:["192.168.8.XXX:9669"]
    #任意一个 Meta 服务的地址。
    #如果您的悦数图数据库在虚拟网络中，如k8s，请配置 Leader Meta的地址。
    meta:["192.168.8.XXX:9559"]
  }
  user: root
  pswd: nebula
  space: basketballplayer

  # SST 文件相关配置
  path:{
      # 本地临时存放生成的 SST 文件的目录
      local:"/tmp"

      # SST 文件在 HDFS 的存储路径
      remote:"/sst"

      # HDFS 的 NameNode 地址，例如 "hdfs://<ip/hostname>:<port>"。
      hdfs.namenode: "hdfs://*.*.*.*:9000"
  }

  # 客户端连接参数
  connection: {
    # socket 连接、执行的超时时间，单位：毫秒。
    timeout: 30000
  }

  error: {
    # 最大失败数，超过后会退出应用程序。
    max: 32
    # 失败的导入作业将记录在输出路径中。
    output: /tmp/errors
  }

  # 使用Google Guava RateLimiter 来限制发送到悦数图数据库的请求。
  rate: {
    # RateLimiter 的稳定吞吐量。
    limit: 1024

    # 从 RateLimiter 获取允许的超时时间，单位：毫秒
    timeout: 1000
  }
}

# 处理点
tags: [
  # 设置 Tag player 相关信息。
  {
    # 指定悦数图数据库中定义的 Tag 名称。
    name: player
    type: {
      # 指定数据源，使用 CSV。
      source: csv

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: sst
    }

    # 指定 CSV 文件的路径。
    # 文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://ip:port/xx/xx.csv"。
    path: "hdfs://*.*.*.*:9000/dataset/vertex_player.csv"

    # 如果 CSV 文件没有表头，使用 [_c0, _c1, _c2, ..., _cn] 表示其表头，并将列指示为属性值的源。
    # 如果 CSV 文件有表头，则使用实际的列名。
    fields: [_c1, _c2]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [age, name]

    # 指定一个列作为 VID 的源。
    # vertex 的值必须与上述 fields 或者 csv.fields 中的列名保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    vertex: {
      field:_c0
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定的分隔符。默认值为英文逗号（,）。
    separator: ","
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    # 如果 CSV 文件有表头，请将 header 设置为 true。
    # 如果 CSV 文件没有表头，请将 header 设置为 false。默认值为 false。
    header: false

    # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
    #writeMode: INSERT

    # 指定单批次写入悦数图数数据库的最大点数量。
    batch: 256

    # 指定 Spark 分片数量。
    partition: 32

    # 生成 SST 文件时是否要基于悦数图数数据库中图空间的 partition 进行数据重分区。
    repartitionWithNebula: false
  }

  # 设置 Tag team 相关信息。
  {
    name: team
    type: {
      source: csv
      sink: sst
    }
    path: "hdfs://*.*.*.*:9000/dataset/vertex_team.csv"
    fields: [_c1]
    nebula.fields: [name]
    vertex: {
      field:_c0
    }
    separator: ","
    header: false
    batch: 256
    partition: 32
    repartitionWithNebula: false
  }

  # 如果需要添加更多点，请参考前面的配置进行添加。
]
# 处理边
edges: [
  # 设置 Edge type follow 相关信息。
  {
    # 指定悦数图数据库中定义的 Edge type 名称。
    name: follow
    type: {
      # 指定数据源，使用 CSV。
      source: csv

      # 指定如何将点数据导入悦数图数据库：Client 或 SST。
      sink: sst
    }

    # 指定 CSV 文件的路径。
    # 文件存储在 HDFS 上，用双引号括起路径，以 hdfs://开头，例如"hdfs://<ip/hostname>:port/xx/xx.csv"。
    path: "hdfs://*.*.*.*:9000/dataset/edge_follow.csv"

    # 如果 CSV 文件没有表头，使用 [_c0, _c1, _c2, ..., _cn] 表示其表头，并将列指示为属性值的源。
    # 如果 CSV 文件有表头，则使用实际的列名。
    fields: [_c2]

    # 指定悦数图数据库中定义的属性名称。
    # fields 与 nebula.fields 的顺序必须一一对应。
    nebula.fields: [degree]

    # 指定一个列作为起始点和目的点的源。
    # vertex 的值必须与上述 fields 或者 csv.fields 中的列名保持一致。
    # 目前，悦数图数据库 3.5.1仅支持字符串或整数类型的 VID。
    source: {
      field: _c0
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }
    target: {
      field: _c1
    # 为 VID 增加指定的前缀。例如 VID 为 12345，增加前缀 tag1 后为 tag1_12345。下划线无法修改。
    # prefix:"tag1"
    # 对 string 类型的 VID 进行哈希化操作。
    # policy:hash
    }

    # 指定的分隔符。默认值为英文逗号（,）。
    separator: ","

    # 指定一个列作为 rank 的源（可选）。

    #ranking: rank

    # 如果 CSV 文件有表头，请将 header 设置为 true。
    # 如果 CSV 文件没有表头，请将 header 设置为 false。默认值为 false。
    header: false
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # 批量操作类型，包括 INSERT、UPDATE 和 DELETE。默认为 INSERT。
      #writeMode: INSERT

      # 指定单批次写入悦数图数据库的最大边数量。
      batch: 256

      # 指定 Spark 分片数量。
      partition: 32

      # 生成 SST 文件时是否要基于悦数图数据库中图空间的 partition 进行数据重分区。
      repartitionWithNebula: false
    }

    # 设置 Edge type serve 相关信息。
    {
      name: serve
      type: {
        source: csv
        sink: sst
      }
      path: "hdfs://*.*.*.*:9000/dataset/edge_serve.csv"
      fields: [_c2,_c3]
      nebula.fields: [start_year, end_year]
      source: {
        field: _c0
      }
      target: {
        field: _c1
      }
      separator: ","
      header: false
      batch: 256
      partition: 32
      repartitionWithNebula: false
    }

  ]
  # 如果需要添加更多边，请参考前面的配置进行添加。
}
```

步骤 **4**：生成 **SST** 文件

运行如下命令将 CSV 源文件生成为 SST 文件。关于参数的说明，请参见命令参数。

```
${SPARK_HOME}/bin/spark-submit --master "local" --conf spark.sql.shuffle.partitions=<shuffle_concurrency> --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.5.0.jar_path> -c
<sst_application.conf_path>
```

Note

生成 SST 文件时，会涉及到 Spark 的 shuffle 操作，请注意在提交命令中增加 `spark.sql.shuffle.partitions` 的配置。

Note

JAR 包有两种获取方式：自行编译或者从 maven 仓库下载。

示例：

```
${SPARK_HOME}/bin/spark-submit  --master "local" --conf spark.sql.shuffle.partitions=200 --class com.vesoft.nebula.exchange.Exchange  /root/nebula-exchange/nebula-exchange/target/nebula-
exchange-3.5.0.jar  -c /root/nebula-exchange/nebula-exchange/target/classes/sst_application.conf
```

任务执行完成后，可以在 HDFS 上的 `/sst` 目录（`nebula.path.remote` 参数指定）内查看到生成的 SST 文件。

Note

如果对 Schema 有修改操作，例如重建图空间、修改 Tag、修改 Edge type 等，需要重新生成 SST 文件，因为 SST 文件会验证 Space ID、Tag ID、Edge ID 等信息。

Copyright © Yueshu Graph Technology - All rights reserved.

步骤 **5**: 导入 **SST** 文件

Note

导入前请确认以下信息：

- 确认所有部署 Storage 服务的机器上都已部署 Hadoop 服务，并配置 HADOOP_HOME 和 JAVA_HOME。
- Meta 服务配置文件中的 `--ws_storage_http_port` （如果没有，请手动添加）和 Storage 服务配置文件中的 `--ws_http_port` 一致。例如都为 `19779` 。
- Graph 服务配置文件中的 `--ws_meta_http_port` （如果没有，请手动添加）和 Meta 服务配置文件中的 `--ws_http_port` 一致。例如都为 `19559` 。

使用客户端工具连接悦数图数据库，按如下操作导入 SST 文件：

1. 执行命令选择之前创建的图空间。

```
nebula> USE basketballplayer;
```

2. 执行命令下载 SST 文件：

```
nebula> SUBMIT JOB DOWNLOAD HDFS "hdfs://<hadoop_address>:<hadoop_port>/<sst_file_path>";
```

示例：

```
nebula> SUBMIT JOB DOWNLOAD HDFS "hdfs://*.*.*.*:9000/sst";
```

3. 执行命令导入 SST 文件：

```
nebula> SUBMIT JOB INGEST;
```

Note

- 如果需要重新下载，请在悦数图数据库安装路径内的 `data/storage/nebula` 目录内，将对应 Space ID 目录内的 `download` 文件夹删除，然后重新下载 SST 文件。如果图空间是多副本，保存副本的所有机器都需要删除 `download` 文件夹。
- 如果导入时出现问题需要重新导入，重新执行 `SUBMIT JOB INGEST;` 即可。

步骤 **6**: （可选）验证数据

用户可以在悦数图数据库客户端（例如 NebulaGraph Studio）中执行查询语句，确认数据是否已导入。例如：

```
LOOKUP ON player YIELD id(vertex);
```

用户也可以使用命令 `SHOW STATS` 查看统计数据。

步骤 **7**: （如有）在悦数图数据库中重建索引

导入数据后，用户可以在悦数图数据库中重新创建并重建索引。详情请参见索引介绍。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

导出悦数图数据库数据

Exchange 支持将悦数图数据库中的数据导出到 CSV 文件或另一个图空间（不同悦数图数据库集群也支持）中。本文介绍具体的操作步骤。

> **Enterpriseonly**
>
> 仅企业版 Exchange 支持导出悦数图数据库数据。

环境准备

本示例在 Linux 系统的虚拟机环境下完成，导出数据前准备的软硬件信息如下。

硬件

类型	信息
CPU	4 Intel(R) Xeon(R) Platinum 8260 CPU @ 2.30GHz
内存	16G
硬盘	50G

系统

CentOS 7.9.2009

软件

名称	版本
JDK	1.8.0
Scala	2.12.11
Spark	2.4.7
悦数图数据库	3.5.1

数据集

在本示例中，作为数据源的悦数图数据库存储着 basketballplayer 数据集，其中的 Schema 要素如下表所示。

要素	名称	属性
Tag	`player`	`name string, age int`
Tag	`team`	`name string`
Edge type	`follow`	`degree int`
Edge type	`serve`	`start_year int, end_year int`

操作步骤

1. 从 悦数图数据库企业版套餐 中获取企业版 Exchange 的 JAR 文件。
2. 修改配置文件。

企业版 Exchange 提供了导出悦数图数据库数据专用的配置文件模板 `export_to_csv.conf` 和 `export_to_nebula.conf`，其中各配置项的说明参见 Exchange 配置。本示例使用的配置文件核心内容如下：

- 导出到 CSV 文件：

```
# Use the command to submit the exchange job:

# spark-submit \
# --master "spark://master_ip:7077" \
# --driver-memory=2G --executor-memory=30G  \
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# --total-executor-cores=60 --executor-cores=20 \
# --class com.vesoft.nebula.exchange.Exchange \
# nebula-exchange-3.0-SNAPSHOT.jar -c export_to_csv.conf

{
  # Spark config
  spark: {
    app: {
      name: 悦数图数据库 Exchange
    }
  }

  # Nebula Graph config
  # if you export nebula data to csv, please ignore these nebula config
  nebula: {
    address:{
      graph:["127.0.0.1:9669"]

      # the address of any of the meta services.
      # if your 悦数图数据库 server is in virtual network like k8s, please config the leader address of meta.
      meta:["127.0.0.1:9559"]
    }
    user: root
    pswd: nebula
    space: test

    # nebula client connection parameters
    connection {
      # socket connect & execute timeout, unit: millisecond
      timeout: 30000
    }

    error: {
      # max number of failures, if the number of failures is bigger than max, then exit the   application.
      max: 32
      # failed data will be recorded in output path, format with ngql
      output: /tmp/errors
    }

    # use google's RateLimiter to limit the requests send to 悦数图数据库
    rate: {
      # the stable throughput of RateLimiter
      limit: 1024
      # Acquires a permit from RateLimiter, unit: MILLISECONDS
      # if it can't be obtained within the specified timeout, then give up the request.
      timeout: 1000
    }
  }

  # Processing tags
  tags: [
    {
      # you can ignore the tag name when export nebula data to csv
      name: tag-name-1
      type: {
        source: nebula
        sink: csv
      }
      metaAddress:"127.0.0.1:9559"
      space:"test"
      label:"person"
      # config the fields you want to export from nebula. If you want to export all properties, you can set this parameter to empty, that is `fields: []`.
      fields: [nebula-field-0, nebula-field-1, nebula-field-2]
      noFields:false  # default false, if true, just export id
      partition: 60
      # config the path to save your csv file. if your file in not in hdfs, config "file:///path/  test.csv"
      path: "hdfs://ip:port/path/person"
      separator: ","
      header: true
    }
  ]

  # process edges
  edges: [
    {
      # you can ignore the edge name when export nebula data to csv
      name: edge-name-1
      type: {
        source: nebula
        sink: csv
      }
      metaAddress:"127.0.0.1:9559"
      space:"test"
      label:"friend"
      # config the fields you want to export from nebula. If you want to export all properties, you can set this parameter to empty, that is `fields: []`.
      fields: [nebula-field-0, nebula-field-1, nebula-field-2]
      noFields:false  # default false, if true, just export id
      partition: 60
      # config the path to save your csv file. if your file in not in hdfs, config "file:///path/  test.csv"
      path: "hdfs://ip:port/path/friend"
      separator: ","
      header: true
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  ]
}
```

- 导出到另一个图空间:

```
# Use the command to submit the exchange job:

# spark-submit \
# --master "spark://master_ip:7077" \
# --driver-memory=2G --executor-memory=30G  \
# --total-executor-cores=60 --executor-cores=20 \
# --class com.vesoft.nebula.exchange.Exchange \
# nebula-exchange-3.0-SNAPSHOT.jar -c export_to_nebula.conf

{
  # Spark config
  spark: {
    app: {
      name: 悦数图数据库 Exchange
    }
  }

  # Nebula Graph config, just config the sink nebula information
  nebula: {
    address:{
      graph:["127.0.0.1:9669"]

      # the address of any of the meta services.
      # if your 悦数图数据库 server is in virtual network like k8s, please config the leader address of meta.
      meta:["127.0.0.1:9559"]
    }
    user: root
    pswd: nebula
    space: test

    # nebula client connection parameters
    connection {
      # socket connect & execute timeout, unit: millisecond
      timeout: 30000
    }

    error: {
      # max number of failures, if the number of failures is bigger than max, then exit the   application.
      max: 32
      # failed data will be recorded in output path, format with ngql
      output: /tmp/errors
    }

    # use google's RateLimiter to limit the requests send to 悦数图数据库
    rate: {
      # the stable throughput of RateLimiter
      limit: 1024
      # Acquires a permit from RateLimiter, unit: MILLISECONDS
      # if it can't be obtained within the specified timeout, then give up the request.
      timeout: 1000
    }
  }

  # Processing tags
  tags: [
    {
      name: tag-name-1
      type: {
        source: nebula
        sink: client
      }
      # data source nebula config
      metaAddress:"127.0.0.1:9559"
      space:"test"
      label:"person"
      # mapping the fields of the original 悦数图数据库 to the fields of the target 悦数图数据库.
      fields: [source_nebula-field-0, source_nebula-field-1, source_nebula-field-2]
      nebula.fields: [target_nebula-field-0, target_nebula-field-1, target_nebula-field-2]
      limit:10000
      vertex: _vertexId  # must be `_vertexId`
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
      batch: 2000
      partition: 60
    }
  ]

  # process edges
  edges: [
    {
      name: edge-name-1
      type: {
        source: csv
        sink: client
      }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      # data source nebula config
      metaAddress:"127.0.0.1:9559"
      space:"test"
      label:"friend"
      fields: [source_nebula-field-0, source_nebula-field-1, source_nebula-field-2]
      nebula.fields: [target_nebula-field-0, target_nebula-figld-1, target_nebula-field-2]
      limit:1000
      source: _srcId # must be `_srcId`
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
      target: _dstId # must be `_dstId`
    # udf:{
    #            separator:"_"
    #            oldColNames:[field-0,field-1,field-2]
    #            newColName:new-field
    #        }
      ranking: source_nebula-field-2
      batch: 2000
      partition: 60
    }
  ]
}
```

3. 使用如下命令导出悦数图数据库中的数据。

> **Note**
>
> Driver 和 Executor 进程的相关参数可以根据自身配置灵活修改。

```
<spark_install_path>/bin/spark-submit --master "spark://<master_ip>:7077" \
--driver-memory=2G --executor-memory=30G \
--total-executor-cores=60 --executor-cores=20 \
--class com.vesoft.nebula.exchange.Exchange nebula-exchange-x.y.z.jar_path> \
-c <conf_file_path>
```

例如导出到 CSV 文件的示例命令如下。

```
$ ./spark-submit --master "spark://192.168.10.100:7077" \
--driver-memory=2G --executor-memory=30G \
--total-executor-cores=60 --executor-cores=20 \
--class com.vesoft.nebula.exchange.Exchange ~/exchange-ent/nebula-exchange-ent-3.5.0.jar \
-c ~/exchange-ent/export_to_csv.conf
```

4. 检查导出的数据。

- 导出到 CSV 文件：

  查看目标路径下是否成功生成了 CSV 文件，并检查文件内容。

```
$ hadoop fs -ls /vertex/player
Found 11 items
-rw-r--r--   3 nebula supergroup          0 2021-11-05 07:36 /vertex/player/_SUCCESS
-rw-r--r--   3 nebula supergroup        160 2021-11-05 07:36 /vertex/player/    part-00000-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        163 2021-11-05 07:36 /vertex/player/    part-00001-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        172 2021-11-05 07:36 /vertex/player/    part-00002-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        172 2021-11-05 07:36 /vertex/player/    part-00003-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        144 2021-11-05 07:36 /vertex/player/    part-00004-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        173 2021-11-05 07:36 /vertex/player/    part-00005-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        160 2021-11-05 07:36 /vertex/player/    part-00006-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        148 2021-11-05 07:36 /vertex/player/    part-00007-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        125 2021-11-05 07:36 /vertex/player/    part-00008-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--   3 nebula supergroup        119 2021-11-05 07:36 /vertex/player/    part-00009-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
```

- 导出到另一个图空间：

  登录新的图空间，通过 `SUBMIT JOB STATS` 和 `SHOW STATS` 命令查看统计信息，确认是否导出成功。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.3.5 Exchange 常见问题

编译问题

Q：部分非 **CENTRAL** 仓库的包下载失败，报错 **`COULD NOT RESOLVE DEPENDENCIES FOR PROJECT XXX`**

请检查 Maven 安装目录下 `libexec/conf/settings.xml` 文件的 `mirror` 部分：

```
<mirror>
    <id>alimaven</id>
    <mirrorOf>central</mirrorOf>
    <name>aliyun maven</name>
    <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
</mirror>
```

检查 `mirrorOf` 的值是否配置为 `*`，如果为 `*`，请修改为 `central` 或 `*,!SparkPackagesRepo,!bintray-streamnative-maven`。

原因：Exchange 的 `pom.xml` 中有两个依赖包不在 Maven 的 central 仓库中，`pom.xml` 配置了这两个依赖所在的仓库地址。如果 maven 中配置的镜像地址对应的 `mirrorOf` 值为 `*`，那么所有依赖都会在 central 仓库下载，导致下载失败。

Q：编译 **EXCHANGE** 时无法下载 **SNAPSHOT** 包

现象：编译时提示 `Could not find artifact com.vesoft:client:jar:xxx-SNAPSHOT`。

原因：本地 maven 没有配置用于下载 SNAPSHOT 的仓库。maven 中默认的 central 仓库用于存放正式发布版本，而不是开发版本（SNAPSHOT）。

解决方案：在 maven 的 setting.xml文件的 `profiles` 作用域内中增加以下配置：

```
  <profile>
     <activation>
        <activeByDefault>true</activeByDefault>
     </activation>
     <repositories>
        <repository>
            <id>snapshots</id>
            <url>https://oss.sonatype.org/content/repositories/snapshots/</url>
            <snapshots>
               <enabled>true</enabled>
            </snapshots>
      </repository>
     </repositories>
  </profile>
```

执行问题

Q：报错 **`JAVA.LANG.CLASSNOTFOUNDEXCEPTION: COM.VESOFT.NEBULA.EXCHANGE.EXCHANGE`**

在 Yarn-Cluster 模式下提交任务，请参考如下命令，尤其是示例中的两个 `--conf`：

```
$SPARK_HOME/bin/spark-submit --class com.vesoft.nebula.exchange.Exchange \
--master yarn-cluster \
--files application.conf \
--conf spark.driver.extraClassPath=./ \
--conf spark.executor.extraClassPath=./ \
nebula-exchange-3.0.0.jar \
-c application.conf
```

Q：报错 **`METHOD NAME XXX NOT FOUND`**

一般是端口配置错误，需检查 Meta 服务、Graph 服务、Storage 服务的端口配置。

Q：报 **NOSUCHMETHOD**、**METHODNOTFOUND** 错误（**`EXCEPTION IN THREAD "MAIN" JAVA.LANG.NOSUCHMETHODERROR`** 等）

绝大多数是因为 JAR 包冲突和版本冲突导致的报错，请检查报错服务的版本，与 Exchange 中使用的版本进行对比，检查是否一致，尤其是 Spark 版本、Scala 版本、Hive 版本。

Q：**EXCHANGE** 导入 **HIVE** 数据时报错 **`EXCEPTION IN THREAD "MAIN" ORG.APACHE.SPARK.SQL.ANALYSISEXCEPTION: TABLE OR VIEW NOT FOUND`**

检查提交 exchange 任务的命令中是否遗漏参数 `-h`，检查 table 和 database 是否正确，在 spark-sql 中执行用户配置的 exec 语句，验证 exec 语句的正确性。

Copyright © Yueshu Graph Technology - All rights reserved.

Q：运行时报错 `COM.FACEBOOK.THRIFT.PROTOCOL.TPROTOCOLEXCEPTION: EXPECTED PROTOCOL ID XXX`

请检查悦数图数据库服务端口配置是否正确。

- 如果是源码、RPM 或 DEB 安装，请配置各个服务的配置文件中 `--port` 对应的端口号。
- 如果是 docker 安装，请配置 docker 映射出来的端口号，查看方式如下：

  在 `nebula-docker-compose` 目录下执行 `docker-compose ps`，例如：

```
$ docker-compose ps
              Name                            Command                  State                                                    Ports
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
nebula-docker-compose_graphd_1      /usr/local/nebula/bin/nebu ...   Up (healthy)   0.0.0.0:33205->19669/tcp, 0.0.0.0:33204->19670/tcp, 0.0.0.0:9669->9669/tcp
nebula-docker-compose_metad0_1      ./bin/nebula-metad --flagf ...   Up (healthy)   0.0.0.0:33165->19559/tcp, 0.0.0.0:33162->19560/tcp, 0.0.0.0:33167->9559/tcp, 9560/tcp
nebula-docker-compose_metad1_1      ./bin/nebula-metad --flagf ...   Up (healthy)   0.0.0.0:33166->19559/tcp, 0.0.0.0:33163->19560/tcp, 0.0.0.0:33168->9559/tcp, 9560/tcp
nebula-docker-compose_metad2_1      ./bin/nebula-metad --flagf ...   Up (healthy)   0.0.0.0:33161->19559/tcp, 0.0.0.0:33160->19560/tcp, 0.0.0.0:33164->9559/tcp, 9560/tcp
nebula-docker-compose_storaged0_1   ./bin/nebula-storaged --fl ...   Up (healthy)   0.0.0.0:33180->19779/tcp, 0.0.0.0:33178->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:33183->9779/tcp, 9780/
tcp
nebula-docker-compose_storaged1_1   ./bin/nebula-storaged --fl ...   Up (healthy)   0.0.0.0:33175->19779/tcp, 0.0.0.0:33172->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:33177->9779/tcp, 9780/
tcp
nebula-docker-compose_storaged2_1   ./bin/nebula-storaged --fl ...   Up (healthy)   0.0.0.0:33184->19779/tcp, 0.0.0.0:33181->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:33185->9779/tcp, 9780/
tcp
```

  查看 `Ports` 列，查找 docker 映射的端口号，例如：

  - Graph 服务可用的端口号是 9669。

  - Meta 服务可用的端口号有 33167、33168、33164。

  - Storage 服务可用的端口号有 33183、33177、33185。

Q：运行时报错 `EXCEPTION IN THREAD "MAIN" COM.FACEBOOK.THRIFT.PROTOCOL.TPROTOCOLEXCEPTION: THE FIELD 'CODE' HAS BEEN ASSIGNED THE INVALID VALUE -4`

检查 Exchange 版本与悦数图数据库版本是否匹配，详细信息可参考使用限制。

Q：将 **HIVE** 中的数据导入悦数图数据库时出现乱码如何解决?

如果 Hive 中数据的属性值包含中文字符，可能出现该情况。解决方案是在导入命令中的 JAR 包路径前加上以下选项：

```
--conf spark.driver.extraJavaOptions=-Dfile.encoding=utf-8
--conf spark.executor.extraJavaOptions=-Dfile.encoding=utf-8
```

即：

```
<spark_install_path>/bin/spark-submit --master "local" \
--conf spark.driver.extraJavaOptions=-Dfile.encoding=utf-8 \
--conf spark.executor.extraJavaOptions=-Dfile.encoding=utf-8 \
--class com.vesoft.nebula.exchange.Exchange \
<nebula-exchange-3.x.y.jar_path> -c <application.conf_path>
```

如果是在 YARN 中，则用以下命令：

```
<spark_install_path>/bin/spark-submit \
--class com.vesoft.nebula.exchange.Exchange \
--master yarn-cluster \
--files <application.conf_path> \
--conf spark.driver.extraClassPath=./ \
--conf spark.executor.extraClassPath=./ \
--conf spark.driver.extraJavaOptions=-Dfile.encoding=utf-8 \
--conf spark.executor.extraJavaOptions=-Dfile.encoding=utf-8 \
<nebula-exchange-3.x.y.jar_path> \
-c application.conf
```

Q：**HIVE** 数据导入时提示 **SCHEMA** 版本不一致

Spark 日志提示 `Hive Schema version 1.2.0 does not match metastore's schema version 2.1.0 Metastore is not upgraded or corrupt` 的原因是 Hive 环境中配置的 metastore schema 版本和 Spark 使用的 metastore 版本不一致。

Copyright © Yueshu Graph Technology - All rights reserved.

解决方法：

1. 将 Hive 环境中存储 Hive metastore 信息的 MySQL version 信息更新为 Spark 中使用的 metastore 版本。

   假设 Hive 在 MySQL 中存储 metastore 的数据库是 `hive`，需要按如下方式修改 `hive.VERSION` 表中的 `version` 字段：

```
update hive.VERSION set SCHEMA_VERSION="2.1.0" where VER_ID=1
```

2. 在 Hive 环境的 `hive-site.xml` 文件中增加如下配置：

```
<property>
<name>hive.metastore.schema.verification</name>
<value>false</value>
</property>
```

3. 重启 Hive。

Q: 生成 **SST** 时提示 **ORG.ROCKSDB.ROCKSDBEXCEPTION: WHILE OPEN A FILE FOR APPENDING: /PATH/SST/1-XXX.SST: NO SUCH FILE OR DIRECTORY**

排查方法：

1. 检查 `/path` 是否存在，如没有或者路径设置错误，创建或修正路径。
2. 检查 Spark 在每台机器上的当前用户对 `/path` 是否有操作权限，如没有，添加权限。

配置问题

Q：哪些配置项影响导入性能？

- batch：每次发送给悦数图数据库服务的 nGQL 语句中包含的数据条数。
- partition：Spark 数据的分区数，表示数据导入的并发数。
- nebula.rate：向悦数图数据库发送请求前先去令牌桶获取令牌。
  - limit：表示令牌桶的大小。
  - timeout：表示获取令牌的超时时间。

根据机器性能可适当调整这四项参数的值。如果在导入过程中，Storage 服务的 leader 变更，可以适当调小这四项参数的值，降低导入速度。

其他问题

Q：**EXCHANGE** 支持哪些版本的悦数图数据库？

请参见 Exchange 的使用限制。

Q：**EXCHANGE** 与 **SPARK WRITER** 有什么关系？

Exchange 是在 Spark Writer 基础上开发的 Spark 应用程序，二者均适用于在分布式环境中将集群的数据批量迁移到悦数图数据库中，但是后期的维护工作将集中在 Exchange 上。与 Spark Writer 相比，Exchange 有以下改进：

- 支持更丰富的数据源，如 MySQL、Neo4j、Hive、HBase、Kafka、Pulsar 等。
- 修复了 Spark Writer 的部分问题。例如 Spark 读取 HDFS 里的数据时，默认读取到的源数据均为 String 类型，可能与悦数图数据库定义的 Schema 不同，所以 Exchange 增加了数据类型的自动匹配和类型转换，当悦数图数据库定义的 Schema 中数据类型为非 String 类型（如 double）时，Exchange 会将 String 类型的源数据转换为对应的类型（如 double）。

Q：**EXCHANGE** 传输数据的性能如何？

Exchange 的性能测试数据和测试方法参见 NebulaGraph Exchange test result。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.4 NebulaGraph Spark Connector

NebulaGraph Spark Connector 是一个 Spark 连接器，提供通过 Spark 标准形式读写悦数图数据库数据的能力。NebulaGraph Spark Connector 由 Reader 和 Writer 两部分组成。

- Reader

  提供一个 Spark SQL 接口，用户可以使用该接口编程读取悦数图数据库图数据，单次读取一个点或 Edge type 的数据，并将读取的结果组装成 Spark 的 DataFrame。

- Writer

  提供一个 Spark SQL 接口，用户可以使用该接口编程将 DataFrame 格式的数据逐条或批量写入悦数图数据库。

更多使用说明请参见 NebulaGraph Spark Connector。

12.4.1 版本兼容性

NebulaGraph Spark Connector、悦数图数据库内核版本和 Spark 版本对应关系如下。

Spark Connector 版本	悦数图数据库版本	**Spark** 版本
nebula-spark-connector_3.0-3.0-SNAPSHOT.jar	nightly	3.x
nebula-spark-connector_2.2-3.0-SNAPSHOT.jar	nightly	2.2.x
nebula-spark-connector-3.0-SNAPSHOT.jar	nightly	2.4.x
nebula-spark-connector_2.2-3.4.0.jar	3.x	2.2.x
nebula-spark-connector-3.4.0.jar	3.x	2.4.x
nebula-spark-connector_2.2-3.3.0.jar	3.x	2.2.x
nebula-spark-connector-3.3.0.jar	3.x	2.4.x
nebula-spark-connector-3.0.0.jar	3.x	2.4.x
nebula-spark-connector-2.6.1.jar	2.6.0, 2.6.1	2.4.x
nebula-spark-connector-2.6.0.jar	2.6.0, 2.6.1	2.4.x
nebula-spark-connector-2.5.1.jar	2.5.0, 2.5.1	2.4.x
nebula-spark-connector-2.5.0.jar	2.5.0, 2.5.1	2.4.x
nebula-spark-connector-2.1.0.jar	2.0.0, 2.0.1	2.4.x
nebula-spark-connector-2.0.1.jar	2.0.0, 2.0.1	2.4.x
nebula-spark-connector-2.0.0.jar	2.0.0, 2.0.1	2.4.x

12.4.2 适用场景

NebulaGraph Spark Connector 适用于以下场景：

- 在不同的悦数图数据库集群之间迁移数据。
- 在同一个悦数图数据库集群内不同图空间之间迁移数据。
- 悦数图数据库与其他数据源之间迁移数据。
- 结合 NebulaGraph Algorithm 进行图计算。

Copyright © Yueshu Graph Technology - All rights reserved.

12.4.3 特性

NebulaGraph Spark Connector 3.4.0版本特性如下：

- 提供多种连接配置项，如超时时间、连接重试次数、执行重试次数等。
- 提供多种数据配置项，如写入数据时设置对应列为点 ID、起始点 ID、目的点 ID 或属性。
- Reader 支持无属性读取和全属性读取。
- Reader 支持将悦数图数据库数据读取成 Graphx 的 VertexRDD 和 EdgeRDD，支持非 Long 型点 ID。
- 统一了 SparkSQL 的扩展数据源，统一采用 DataSourceV2 进行悦数图数据库数据扩展。
- 支持 `insert` 、 `update` 和 `delete` 三种写入模式。 `insert` 模式会插入（覆盖）数据， `update` 模式仅会更新已存在的数据， `delete` 模式只删除数据。
- 支持与悦数图数据库之间的 SSL 加密连接。

12.4.4 更新说明

Release notes

12.4.5 获取 NebulaGraph Spark Connector

编译打包

1. 克隆仓库 `nebula-spark-connector` 。

```
$ git clone -b release-3.4 https://github.com/vesoft-inc/nebula-spark-connector.git
```

2. 进入目录 `nebula-spark-connector` 。
3. 编译打包。不同版本的 Spark 命令略有不同。

> **Note**
>
> 需已安装对应版本 Spark。

- Spark 2.4

```
$ mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true -pl nebula-spark-connector -am -Pscala-2.11 -Pspark-2.4
```

- Spark 2.2

```
$ mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true -pl nebula-spark-connector_2.2 -am -Pscala-2.11 -Pspark-2.2
```

- Spark 3.x

```
$ mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true -pl nebula-spark-connector_3.0 -am -Pscala-2.12 -Pspark-3.0
```

编译完成后，在目录的文件夹 `target` 下生成类似文件 `nebula-spark-connector-3.4.0-SHANPSHOT.jar` 。

Maven 远程仓库下载

下载地址

Copyright © Yueshu Graph Technology - All rights reserved.

12.4.6 使用方法

使用 NebulaGraph Spark Connector 读写悦数图数据库时，只需要编写以下代码即可实现。

```
# 从悦数图数据库读取点边数据。
spark.read.nebula().loadVerticesToDF()
spark.read.nebula().loadEdgesToDF()

# 将 dataframe 数据作为点和边写入悦数图数据库中。
dataframe.write.nebula().writeVertices()
dataframe.write.nebula().writeEdges()
```

`nebula()` 接收两个配置参数，包括连接配置和读写配置。

从悦数图数据库读取数据

```
val config = NebulaConnectionConfig
  .builder()
  .withMetaAddress("127.0.0.1:9559")
  .withConenctionRetry(2)
  .withExecuteRetry(2)
  .withTimeout(6000)
  .build()

val nebulaReadVertexConfig: ReadNebulaConfig = ReadNebulaConfig
  .builder()
  .withSpace("test")
  .withLabel("person")
  .withNoColumn(false)
  .withReturnCols(List("birthday"))
  .withLimit(10)
  .withPartitionNum(10)
  .build()
val vertex = spark.read.nebula(config, nebulaReadVertexConfig).loadVerticesToDF()

val nebulaReadEdgeConfig: ReadNebulaConfig = ReadNebulaConfig
  .builder()
  .withSpace("test")
  .withLabel("knows")
  .withNoColumn(false)
  .withReturnCols(List("degree"))
  .withLimit(10)
  .withPartitionNum(10)
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  .build()
val edge = spark.read.nebula(config, nebulaReadEdgeConfig).loadEdgesToDF()
```

- `NebulaConnectionConfig` 是连接悦数图数据库的配置，说明如下。

参数	是否必须	说明
`withMetaAddress`	是	所有 Meta 服务的地址，多个地址用英文逗号（,）隔开，格式为 `ip1:port1,ip2:port2,...` 。读取数据不需要配置 `withGraphAddress` 。
`withConnectionRetry`	否	NebulaGraph Java Client 连接悦数图数据库的重试次数。默认值为 `1` 。
`withExecuteRetry`	否	NebulaGraph Java Client 执行查询语句的重试次数。默认值为 `1` 。
`withTimeout`	否	NebulaGraph Java Client 请求响应的超时时间。默认值为 `6000` ，单位：毫秒（ms）。

- `ReadNebulaConfig` 是读取悦数图数据库数据的配置，说明如下。

参数	是否必须	说明
`withSpace`	是	悦数图数据库图空间名称。
`withLabel`	是	悦数图数据库图空间内的 Tag 或 Edge type 名称。
`withNoColumn`	否	是否不读取属性。默认值为 `false` ，表示读取属性。取值为 `true` 时，表示不读取属性，此时 `withReturnCols` 配置无效。
`withReturnCols`	否	配置要读取的点或边的属性集。格式为 `List(property1,property2,...)` ，默认值为 `List()` ，表示读取全部属性。
`withLimit`	否	配置 NebulaGraph Java Storage Client 一次从服务端读取的数据行数。默认值为 1000。
`withPartitionNum`	否	配置读取悦数图数据库数据时 Spark 的分区数。默认值为 100。该值的配置最好不超过图空间的的分片数量（partition_num）。

向悦数图数据库写入数据

Note

DataFrame 中的列会自动作为属性写入悦数图数据库。

```
val config = NebulaConnectionConfig
  .builder()
  .withMetaAddress("127.0.0.1:9559")
  .withGraphAddress("127.0.0.1:9669")
  .withConenctionRetry(2)
  .build()

val nebulaWriteVertexConfig: WriteNebulaVertexConfig = WriteNebulaVertexConfig
  .builder()
  .withSpace("test")
  .withTag("person")
  .withVidField("id")
  .withVidPolicy("hash")
  .withVidAsProp(true)
  .withUser("root")
  .withPasswd("nebula")
  .withBatch(512)
  .build()
df.write.nebula(config, nebulaWriteVertexConfig).writeVertices()

val nebulaWriteEdgeConfig: WriteNebulaEdgeConfig = WriteNebulaEdgeConfig
  .builder()
  .withSpace("test")
  .withEdge("friend")
  .withSrcIdField("src")
  .withSrcPolicy(null)
  .withDstIdField("dst")
  .withDstPolicy(null)
  .withRankField("degree")
  .withSrcAsProperty(true)
  .withDstAsProperty(true)
  .withRankAsProperty(true)
  .withUser("root")
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  .withPasswd("nebula")
  .withBatch(512)
  .build()
df.write.nebula(config, nebulaWriteEdgeConfig).writeEdges()
```

默认写入模式为 `insert`， 可以通过 `withWriteMode` 配置修改为 `update` 或 `delete`：

```
val config = NebulaConnectionConfig
  .builder()
  .withMetaAddress("127.0.0.1:9559")
  .withGraphAddress("127.0.0.1:9669")
  .build()
val nebulaWriteVertexConfig = WriteNebulaVertexConfig
  .builder()
  .withSpace("test")
  .withTag("person")
  .withVidField("id")
  .withVidAsProp(true)
  .withBatch(512)
  .withWriteMode(WriteMode.UPDATE)
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  .build()
df.write.nebula(config, nebulaWriteVertexConfig).writeVertices()
```

- `NebulaConnectionConfig` 是连接悦数图数据库的配置，说明如下。

参数	是否必须	说明
`withMetaAddress`	是	所有 Meta 服务的地址，多个地址用英文逗号（,）隔开，格式为 `ip1:port1,ip2:port2,...` 。
`withGraphAddress`	是	Graph 服务的地址，多个地址用英文逗号（,）隔开，格式为 `ip1:port1,ip2:port2,...` 。
`withConnectionRetry`	否	NebulaGraph Java Client 连接悦数图数据库的重试次数。默认值为 `1` 。

- `WriteNebulaVertexConfig` 是写入点的配置，说明如下。

参数	是否必须	说明
`withSpace`	是	悦数图数据库图空间名称。
`withTag`	是	写入点时需要关联的 Tag 名称。
`withVidField`	是	DataFrame 中作为点 ID 的列。
`withVidPolicy`	否	写入点 ID 时，采用的映射函数，悦数图数据库仅支持 HASH。默认不做映射。
`withVidAsProp`	否	DataFrame 中作为点 ID 的列是否也作为属性写入。默认值为 `false` 。如果配置为 `true` ，请确保 Tag 中有和 `VidField` 相同的属性名。
`withUser`	否	悦数图数据库用户名。若未开启身份验证，无需配置用户名和密码。
`withPasswd`	否	悦数图数据库用户名对应的密码。
`withBatch`	是	一次写入的数据行数，默认值为 `512` 。当 `withWriteMode` 为 `update` 时，该参数的最大值为 `512` 。
`withWriteMode`	否	写入模式。可选值为 `insert` 、 `update` 和 `delete` 。默认为 `insert` 。
`withDeleteEdge`	否	删除点时是否删除该点关联的边。默认为 `false` 。当 `withWriteMode` 为 `delete` 时生效。

Copyright © Yueshu Graph Technology - All rights reserved.

- `WriteNebulaEdgeConfig` 是写入边的配置，说明如下。

参数	是否必须	说明
`withSpace`	是	悦数图数据库图空间名称。
`withEdge`	是	写入边时需要关联的 Edge type 名称。
`withSrcIdField`	是	DataFrame 中作为起始点的列。
`withSrcPolicy`	否	写入起始点时，采用的映射函数，悦数图数据库仅支持 HASH。默认不做映射。
`withDstIdField`	是	DataFrame 中作为目的点的列。
`withDstPolicy`	否	写入目的点时，采用的映射函数，悦数图数据库仅支持 HASH。默认不做映射。
`withRankField`	否	DataFrame 中作为 rank 的列。默认不写入 rank。
`withSrcAsProperty`	否	DataFrame 中作为起始点的列是否也作为属性写入。默认值为 `false` 。如果配置为 `true` ，请确保 Edge type 中有和 `SrcIdField` 相同的属性名。
`withDstAsProperty`	否	DataFrame 中作为目的点的列是否也作为属性写入。默认值为 `false` 。如果配置为 `true` ，请确保 Edge type 中有和 `DstIdField` 相同的属性名。
`withRankAsProperty`	否	DataFrame 中作为 rank 的列是否也作为属性写入。默认值为 `false` 。如果配置为 `true` ，请确保 Edge type 中有和 `RankField` 相同的属性名。
`withUser`	否	悦数图数据库用户名。若未开启身份验证，无需配置用户名和密码。
`withPasswd`	否	悦数图数据库用户名对应的密码。
`withBatch`	是	一次写入的数据行数，默认值为 `512` 。当 `withWriteMode` 为 `update` 时，该参数的最大值为 `512` 。
`withWriteMode`	否	写入模式。可选值为 `insert` 、 `update` 和 `delete` 。默认为 `insert` 。

示例代码

详细的使用方式参见 示例代码。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

12.5 NebulaGraph Flink Connector

NebulaGraph Flink Connector 是一款帮助 Flink 用户快速访问悦数图数据库的连接器，支持从悦数图数据库图数据库中读取数据，或者将其他外部数据源读取的数据写入悦数图数据库图数据库。

更多使用说明请参见 NebulaGraph Flink Connector。

12.5.1 适用场景

NebulaGraph Flink Connector 适用于以下场景:

- 在不同的悦数图数据库集群之间迁移数据。
- 在同一个悦数图数据库集群内不同图空间之间迁移数据。
- 悦数图数据库与其他数据源之间迁移数据。

12.5.2 更新说明

Release notes

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13. 最佳实践

13.1 Compaction

本文介绍 Compaction 的相关信息。

悦数图数据库中，`Compaction` 是最重要的后台操作，对性能有极其重要的影响。

`Compaction` 操作会读取硬盘上的数据，然后重组数据结构和索引，然后再写回硬盘，可以成倍提升读取性能。将大量数据写入悦数图数据库后，为了提高读取性能，需要手动触发 `Compaction` 操作（全量 `Compaction`）。

> **Note**
>
> `Compaction` 操作会长时间占用硬盘的 IO，建议在业务低峰期（例如凌晨）执行该操作。

悦数图数据库有两种类型的 `Compaction` 操作：自动 `Compaction` 和全量 `Compaction`。

13.1.1 自动 `Compaction`

自动 `Compaction` 是在系统读取数据、写入数据或系统重启时自动触发 `Compaction` 操作，提升短时间内的读取性能。默认情况下，自动 `Compaction` 是开启状态，可能在业务高峰期触发，导致意外抢占 IO 影响业务。

13.1.2 全量 `Compaction`

全量 `Compaction` 可以对图空间进行大规模后台操作，例如合并文件、删除 TTL 过期数据等，该操作需要手动发起。使用如下语句执行全量 `Compaction` 操作：

> **Note**
>
> 建议在业务低峰期（例如凌晨）执行该操作，避免大量占用硬盘 IO 影响业务。

```
nebula> USE <your_graph_space>;
nebula> SUBMIT JOB COMPACT;
```

上述命令会返回作业的 ID，用户可以使用如下命令查看 `Compaction` 状态：

```
nebula> SHOW JOB <job_id>;
```

13.1.3 操作建议

为保证悦数图数据库的性能，请参考如下操作建议：

- 数据导入完成后，请执行 `SUBMIT JOB COMPACT`。
- 业务低峰期（例如凌晨）执行 `SUBMIT JOB COMPACT`。
- 为控制 `Compaction` 的写入速率，请在配置文件 `nebula-storaged.conf` 中设置如下参数（注：此参数限制全部写入，包括正常写入和 Compaction）：

```
# 写入速度限制为 20MB/S。
--rocksdb_rate_limit=20 (in MB/s)
```


Copyright © Yueshu Graph Technology - All rights reserved.

13.1.4 FAQ

Compaction 相关的日志在哪?

默认情况下，`/usr/local/nebula/data/storage/nebula/{1}/data/` 目录下的文件名为 `LOG` 文件，或者类似 `LOG.old.1625797988509303`，找到如下的部分。

```
** Compaction Stats [default] **
Level    Files   Size     Score Read(GB)  Rn(GB) Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) CompMergeCPU(sec) Comp(cnt) Avg(sec) KeyIn KeyDrop
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
  L0      2/0    2.46 KB   0.5      0.0     0.0      0.0       0.0      0.0       0.0   1.0      0.0      0.0      0.53              0.51         2    0.264       0      0
 Sum      2/0    2.46 KB   0.0      0.0     0.0      0.0       0.0      0.0       0.0   1.0      0.0      0.0      0.53              0.51         2    0.264       0      0
 Int      0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0      0.0       0.0   0.0      0.0      0.0      0.00              0.00         0    0.000       0      0
```

如果当前的 `L0` 文件数量较多，对读性能影响较大，可以触发 compaction。

可以同时在多个图空间执行全量 **`Compaction`** 操作吗?

可以，但是此时的硬盘 IO 会很高，可能会影响效率。

全量 **`Compaction`** 操作会耗费多长时间?

如果已经设置读写速率限制，例如 `rocksdb_rate_limit` 限制为 20MB/S 时，用户可以通过 `硬盘使用量/rocksdb_rate_limit` 预估需要耗费的时间。 如果没有设置读写速率限制，根据经验，速率大约为 50MB/S。

可以动态调整 **`rocksdb_rate_limit`** 吗?

不可以。

全量 **`Compaction`** 操作开始后可以停止吗?

不可以停止，必须等待操作完成。这是 RocksDB 的限制。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13.2 Storage 负载均衡

用户可以使用 `SUBMIT JOB BALANCE` 语句平衡分片和 Raft leader 的分布，或者清空某些 Storage 服务器方便进行维护。详情请参见 SUBMIT JOB BALANCE。

Danger

`SUBMIT JOB BALANCE` 命令通过创建和执行一组子任务来迁移数据和均衡分片分布，禁止停止集群中的任何机器或改变机器的 IP 地址，直到所有子任务完成，否则后续子任务会失败。

13.2.1 均衡分片分布

Enterpriseonly

仅企业版支持均衡分片分布。

Note

如果当前图空间已经有失败的均衡分片分布作业，无法开始新的均衡分片分布作业，只能恢复之前失败的作业。如果作业一直执行失败，可以先停止作业，再开始新的均衡分片分布作业。

`SUBMIT JOB BALANCE DATA` 语句会开始一个作业，将当前图空间的分片平均分配到所有 Storage 服务器。通过创建和执行一组子任务来迁移数据和均衡分片分布。

Copyright © Yueshu Graph Technology - All rights reserved.

示例

以横向扩容悦数图数据库为例，集群中增加新的 Storage 主机后，新主机上没有分片。

1. 执行命令 `SHOW HOSTS` 检查分片的分布。

```
nebual> SHOW HOSTS;
+----------------+------+----------+--------------+----------------------+------------------------+-------------+
| Host           | Port | Status   | Leader count | Leader distribution  | Partition distribution | Version     |
+----------------+------+----------+--------------+----------------------+------------------------+-------------+
| "192.168.8.101" | 9779 | "ONLINE" | 0            | "No valid partition"  | "No valid partition"   | "3.5.1" |
| "192.168.8.100" | 9779 | "ONLINE" | 15           | "basketballplayer:15" | "basketballplayer:15"  | "3.5.1" |
+----------------+------+----------+--------------+----------------------+------------------------+-------------+
```

2. 进入图空间 `basketballplayer`，然后执行命令 `SUBMIT JOB BALANCE DATA` 将所有分片均衡分布。

```
nebula> USE basketballplayer;
nebula> SUBMIT JOB BALANCE DATA;
+------------+
| New Job Id |
+------------+
| 25         |
+------------+
```

3. 根据返回的任务ID，执行命令 `SHOW JOB <job_id>` 检查任务状态。

```
nebula> SHOW JOB 25;
+------------------------+-------------------+------------+---------------------------+---------------------------+-------------+
| Job Id(spaceId:partId) | Command(src->dst) | Status     | Start Time                | Stop Time                 | State       |
+------------------------+-------------------+------------+---------------------------+---------------------------+-------------+
| 25                     | "DATA_BALANCE"    | "FINISHED" | 2023-01-17T06:24:35.000000 | 2023-01-17T06:24:35.000000 | "SUCCEEDED" |
| "Total:0"              | "Succeeded:0"     | "Failed:0" | "In Progress:0"           | "Invalid:0"               | ""          |
+------------------------+-------------------+------------+---------------------------+---------------------------+-------------+
```

4. 等待所有子任务完成，负载均衡进程结束，执行命令 `SHOW HOSTS` 确认分片已经均衡分布。

> **Note**

`SUBMIT JOB BALANCE DATA` 不会均衡 leader 的分布。均衡 leader 请参见均衡 leader 分布。

```
nebula> SHOW HOSTS;
+----------------+------+----------+--------------+----------------------+------------------------+-------------+
| Host           | Port | Status   | Leader count | Leader distribution  | Partition distribution | Version     |
+----------------+------+----------+--------------+----------------------+------------------------+-------------+
| "192.168.8.101" | 9779 | "ONLINE" | 7            | "basketballplayer:7" | "basketballplayer:7"   | "3.5.1" |
| "192.168.8.100" | 9779 | "ONLINE" | 8            | "basketballplayer:8" | "basketballplayer:8"   | "3.5.1" |
+----------------+------+----------+--------------+----------------------+------------------------+-------------+
```

如果有子任务失败，请执行 `RECOVER JOB <job_id>`。如果重做负载均衡仍然不能解决问题，请到悦数图数据库社区寻求帮助。

停止负载均衡作业

停止负载均衡作业，请执行命令 `STOP JOB <job_id>`。

- 如果没有正在执行的负载均衡作业，会返回错误。
- 如果有正在执行的负载均衡作业，会返回 `Job stopped`。

> **Note**

`STOP JOB <job_id>` 不会停止正在执行的子任务，而是取消所有后续子任务，状态会置为 `INVALID`，然后等待正在执行的子任执行完毕根据结果置为 `SUCCEEDED` 或 `FAILED`。用户可以执行命令 `SHOW JOB <job_id>` 检查停止的作业状态。

恢复负载均衡作业

恢复负载均衡作业，请执行命令 `RECOVER JOB <job_id>`。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

- 可以恢复执行失败的作业。
- 对于停止的作业，悦数图数据库会判断该作业的开始时间（start time）之后是否有相同类型的失败作业（failed job）或完成作业（finished job），如果有的话，无法恢复停止的作业。例如当有 `stopped job1 -> finished job2 -> stopped job3` 时，只能恢复 job3，无法恢复 job1。

迁移分片

迁移指定的 Storage 主机中的分片来缩小集群规模，可以使用命令 `SUBMIT JOB BALANCE DATA REMOVE <ip:port> [,<ip>:<port> ...]`。

例如需要迁移 `192.168.8.100:9779` 中的分片，请执行如下命令：

```
nebula> SUBMIT JOB BALANCE DATA REMOVE 192.168.8.100:9779;
nebula> SHOW HOSTS;
+-----------------+------+----------+--------------+-----------------------+------------------------+----------------------+
| Host            | Port | Status   | Leader count | Leader distribution   | Partition distribution | Version              |
+-----------------+------+----------+--------------+-----------------------+------------------------+----------------------+
| "192.168.8.101" | 9779 | "ONLINE" | 15           | "basketballplayer:15" | "basketballplayer:15"  | "3.5.1" |
| "192.168.8.100" | 9779 | "ONLINE" | 0            | "No valid partition"  | "No valid partition"   | "3.5.1" |
+-----------------+------+----------+--------------+-----------------------+------------------------+----------------------+
```

Note

该命令仅迁移分片，不会将 Storage 主机从集群中删除。删除 Storage 主机请参见管理 Storage 主机。

13.2.2 均衡 leader 分布

用户可以使用命令 `SUBMIT JOB BALANCE LEADER` 均衡分布所有图空间中的 leader。

示例

```
nebula> SUBMIT JOB BALANCE LEADER;
```

用户可以执行 `SHOW HOSTS` 检查结果。

```
nebula> SHOW HOSTS;
+-----------------+------+----------+--------------+----------------------------------+------------------------+---------+
| Host            | Port | Status   | Leader count | Leader distribution              | Partition distribution | Version |
+-----------------+------+----------+--------------+----------------------------------+------------------------+---------+
| "192.168.10.100" | 9779 | "ONLINE" | 4            | "basketballplayer:3"             | "basketballplayer:8"   | "3.5.1" |
| "192.168.10.101" | 9779 | "ONLINE" | 8            | "basketballplayer:3"             | "basketballplayer:8"   | "3.5.1" |
| "192.168.10.102" | 9779 | "ONLINE" | 3            | "basketballplayer:3"             | "basketballplayer:8"   | "3.5.1" |
| "192.168.10.103" | 9779 | "ONLINE" | 0            | "basketballplayer:2"             | "basketballplayer:7"   | "3.5.1" |
| "192.168.10.104" | 9779 | "ONLINE" | 0            | "basketballplayer:2"             | "basketballplayer:7"   | "3.5.1" |
| "192.168.10.105" | 9779 | "ONLINE" | 0            | "basketballplayer:2"             | "basketballplayer:7"   | "3.5.1" |
+-----------------+------+----------+--------------+----------------------------------+------------------------+---------+
```

Caution

在悦数图数据库 3.5.1 中，Leader 切换会导致短时的大量请求错误（Storage Error `E_RPC_FAILURE`），处理方法见 FAQ。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13.3 图建模设计

本文介绍在悦数图数据库项目中成功应用的一些图建模和系统设计的通用建议。

> **Note**
>
> 本文建议是通用的，在特定领域有例外，请结合实际业务情况进行图建模。

13.3.1 以性能为第一目标进行建模

目前悦数图数据库没有完美的建模方法，如何建模取决于想从数据中挖掘的内容。分析数据并根据业务模型创建方便直观的数据模型，测试模型并优化，逐渐适应业务。为了更好的性能，用户可以多次更改或重新设计模型。

设计和评估最重要的查询语句

在测试环节中，通常会验证各种各样的查询语句，以全面评估系统能力。但在大多数生产场景下，每个集群被频繁调用的查询语句的类型并不会太多；根据 20-80 原则，针对重要的查询语句进行建模优化。

避免全量扫描

通过属性索引或者 VID 来先定位到某个（些）点或者边，然后开始图遍历；对于有些查询，它们只给定了一个子图或者路径的（正则）模式，但无法通过属性索引或者 VID 定位到遍历起始的点边，而期望找到库中全部满足该模式的子图，这样的查询是通过全量扫描实现的，这样的性能会很差。悦数图数据库没有实现对于子图或者路径的图结构的索引。

Tag 与 **Edge type** 之间没有绑定关系

任何 Tag 可以与任何 Edge type 相关联，完全交由应用程序控制。不需要在悦数图数据库中预先定义，也没有命令获取哪些 Tag 与哪些 Edge type 相关联。

Tag/Edge type 预先定义了一组属性

建立 Tag（或者 Edge type）时，需要指定对应的属性。通常称为 Schema。

区分“经常改变的部分”和“不经常改变的部分”

改变指的是业务模型和数据模型上的改变（元信息），不是数据自身的改变。

一些图数据库产品是 schema-free 的设计，所以在数据模型上，不论是图拓扑结构还是属性，都可以非常自由。属性可以建模转变为图拓扑，反之亦然。这类系统通常对于图拓扑的访问有特别的优化。

而悦数图数据库 3.5.1 是强 Schema 的（行存型）系统，这意味着业务数据模型中的部分是“不应该经常改变的”，例如属性 Schema 应该避免改变。类似于 MySQL 中 ALTER TABLE 是应该尽量避免的操作。

而点及邻边可以非常低成本的增删，因此可以将业务模型中“经常改变的部分”建模成点或边（关系），而不是属性 Schema。

例如，在一个业务模型中，人的属性是相对固定的，例如“年龄”，“性别”，“姓名”。而“通信好友”，“出入场所”，“交易账号”，“登录设备”等是相对容易改变的。前者适合建模为属性，后者适合建模为点或边。

自环

悦数图数据库是强 Schema 类型系统，使用 `ALTER TAG` 的开销很大，而且也不支持List类型属性，当用户需要为点添加一些临时属性或者List类型的属性时，可以先创建包含所需属性的边类型，然后为点插入一条或多条指向自身的边。查询时只需要查询指向自己的边属性。如下图所示。

Copyright © Yueshu Graph Technology - All rights reserved.

示例：

如果要检索点的临时属性，请从自环边中获取。例如：

```
//创建边类型并插入自环属性。
nebula> CREATE EDGE IF NOT EXISTS temp(tmp int);
nebula> INSERT EDGE temp(tmp) VALUES "player100"->"player100"@1:(1);
nebula> INSERT EDGE temp(tmp) VALUES "player100"->"player100"@2:(2);
nebula> INSERT EDGE temp(tmp) VALUES "player100"->"player100"@3:(3);

//插入数据后，可以通过查询语句查询，例如：
nebula> GO FROM "player100" OVER temp YIELD properties(edge).tmp;
+----------------------+
| properties(EDGE).tmp |
+----------------------+
| 1                    |
| 2                    |
| 3                    |
+----------------------+

//如果需要返回结果为List，可以通过函数实现，例如：
nebula> MATCH (v1:player)-[e:temp]->() return collect(e.tmp);
+----------------+
| collect(e.tmp) |
+----------------+
| [1, 2, 3]      |
+----------------+
```

对于自环的操作没有封装任何的语法糖，操作方式与普通的边无异。

- 悦数图数据库自环小科普（2 分 54 秒）

悬挂边

悬挂边 (Dangling edge) 是指一条起点或者终点不存在于数据库中的边。

在悦数图数据库 3.5.1中，有两种情况可能会出现悬挂边。

第一种情况：在起点和终点插入之前，用 INSERT EDGE 语句插入一条边。

第二种情况：使用 DELETE VERTEX 语句删除点的时候，没有使用 `WITH EDGE` 选项。此时系统默认不删除该点关联的出边和入边，这些边将变成悬挂边。

悦数图数据库 3.5.1 的数据模型中，由于设计允许图中存在“悬挂边”；没有 openCypher 中的 MERGE 语句。对于悬挂边的保证完全依赖应用层面。用户可以使用 GO 和 LOOKUP 语句查询到悬挂边，但无法使用 MATCH 语句查询到悬挂边。

示例：

```
// 插入起点为“11”，终点为“13”并且都不存在于数据库中的悬挂边
nebula> CREATE EDGE IF NOT EXISTS e1 (name string, age int);
nebula> INSERT EDGE e1 (name, age) VALUES "11"->"13":("n1", 1);

// 使用 GO 语句查询
nebula> GO FROM "11" over e1 YIELD properties(edge);
+----------------------+
| properties(EDGE)     |
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
+----------------------+
| {age: 1, name: "n1"} |
+----------------------+

// 使用 LOOKUP 语句查询
nebula> LOOKUP ON e1 YIELD EDGE AS r;
+-------------------------------------------------------+
| r                                                     |
+-------------------------------------------------------+
| [:e2 "11"->"13" @0 {age: 1, name: "n1"}]              |
+-------------------------------------------------------+

// 使用 MATCH 查询，不能查询到悬挂边
nebula> MATCH ()-[e:e1]->() RETURN e;
+---+
| e |
+---+
+---+
Empty set (time spent 3153/3573 us)
```

- 悦数图数据库的悬挂边小科普（2 分 28 秒）

广度优先大于深度优先

- 悦数图数据库基于图拓扑结构进行深度图遍历的性能较低，广度优先遍历以及获取属性的性能较好。例如，模型 a 包括姓名、年龄、眼睛颜色三种属性，建议创建一个 Tag `person`，然后为它添加姓名、年龄、眼睛颜色的属性。如果创建一个包含眼睛颜色的 Tag 和一个 Edge type `has`，然后创建一个边用来表示人拥有的眼睛颜色，这种建模方法会降低遍历性能。
- “通过边属性获取边”的性能与“通过点属性获取点”的性能是接近的。在一些数据库中，会建议将边上的属性重新建模为中间节点的属性：例如 `(src)-[edge {P1, P2}]->(dst)`，`edge` 上有属性 `P1, P2`，会建议建模为 `(src)-[edge1]->(i_node {P1, P2})-[edge2]->(dst)`。在悦数图数据库 3.5.1 中可以直接使用 `(src)-[edge {P1, P2}]->(dst)`，减少遍历深度有助于性能。

边的方向

查询时，如果需要使用边的逆向查询，可以用如下语法：

`(dst)<-[edge]-(src)` 或者 `GO FROM dst REVERSELY;`

如果不关心边的方向，可以使用如下语法：

`(src)-[edge]-(dst)` 或者 `GO FROM src BIDIRECT;`

因此，通常同一条边没有必要反向再冗余插入一次。

合理设置 **Tag** 属性

在图建模中，请将一组类似的平级属性放入同一个 Tag，即按不同概念进行分组。

正确使用索引

使用属性索引可以通过属性查找到 VID。但是索引会导致写性能大幅下降，只有在根据点或边的属性定位点或边时才使用索引。

合理设计 **VID**

参考点 VID 一节。

长文本

为边创建属性时请勿使用长文本：这些属性会被存储 2 份，导致写入放大问题（write amplification）。此时建议将长文本放在 HBase/ES 中，将其地址存放在悦数图数据库中。

Copyright © Yueshu Graph Technology - All rights reserved.

13.3.2 关于支持动态图（时序图）

在某些场景下，图需要同时带有时序信息，以描述整个图的结构随着时间变化的情况 [1]。

悦数图数据库 3.5.1 的边可以使用 Rank 字段存放时间信息 (int64)，但是点上没有字段可以存放时间信息（存放在属性会被新写入覆盖）。一个折中的办法是在点上设计自己指向自己的自环，并将时间信息放置在自环的 Rank 上。

13.3.3 一些免费的建模工具

arrows.app

1. https://blog.twitter.com/engineering/en_us/topics/insights/2021/temporal-graph-networks ↩

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13.4 系统设计建议

13.4.1 选择 QPS 优先或时延优先

- 悦数图数据库 3.5.1 更擅长处理（互联网式的）有大量并发的小请求。也即：虽然全图很大（万亿点边），但是每个请求要访问到的子图本身并不大（几百万个点边）——单个请求时延不大；但这类请求的并发数量特别多——QPS 大。
- 但对于一些交互分析型的场景，并发请求的数量不多，而每个请求要访问的子图本身特别大（亿以上）。为降低时延，可以在应用程序中将一个大的请求，拆分为多个小请求，并发发送给多个 graphd。这样可以降低单个大请求的时延，降低单个 graphd 的内存占用。另外，也可以使用图计算功能 NebulaGraph Algorithm。

13.4.2 数据传输与优化

- 读写平衡。悦数图数据库适合读写平衡性的在线场景，也即 OLTP 型的的“并发的发生写入与读取”；而非数仓 OLAP 型的“一次写入多次读取”。
- 选择不同的写入方式。大批量的数据写入可以使用 sst 加载的方式；小批量的写入使用 `INSERT` 语句。
- 选择合适的时间运行 COMPACTION 和 BALANCE，来分别优化数据格式和存储分布。
- 悦数图数据库 3.5.1 不支持关系型数据库意义上的事务和隔离性，更接近 NoSQL。

13.4.3 查询预热与数据预热

应用端进行预热：

- Graphd 不支持预编译查询及相应生成查询计划，也不支持缓存之前的查询结果；
- Storaged 不支持预热数据，只有 RocksDB 自身的 LSM-tree 和 BloomFilter 会启动时加载到内存中。
- 点和边被访问过后，会各自缓存在 Storaged 的两种 (LRU) Cache 中。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13.5 执行计划

悦数图数据库 3.5.1 实现了基于规则的执行计划。用户无法改变执行计划，无法进行查询的预编译（及相应的计划缓存），无法通过指定索引来加速查询。

要查看执行计划及执行概要，请参考 EXPLAIN 和 PROFILE。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13.6 超级顶点（稠密点）处理

13.6.1 原理介绍

在图论中，超级顶点（稠密点）是指一个点有着极多的相邻边。相邻边可以是出边（我指向谁）或者是入边（谁指向我）。

由于幂律分布的特点，超级顶点现象非常普遍。 例如社交网络中的影响力领袖（网红大 V）、证券市场中的热门股票、银行系统中的四大行、交通网络中的枢纽站、互联网中的高流量网站等、电商网络中的爆款产品。

在悦数图数据库 3.5.1 中，一个 `点` 和其 `属性` 是一个 `Key-Value`（以该点的 `VID` 以及其他元信息作为 `Key`），其 `Out-Edge Key-Value` 和 `In-Edge Key-Value` 都存储在同一个 partition 中（具体原理详见存储架构，并且以 LSM-tree 的形式组织存放在硬盘（和缓存）中。

因此不论是 `从该点出发的有向遍历`，或者 `以该点为终点的有向遍历`，都会涉及到大量的 `顺序 IO 扫描`（最理想情况，当完成 Compact 操作之后），或者大量的 `随机 IO`（有关于 `该点` 和其 `出入边` 频繁的写入）。

经验上说，当一个点的出入度超过 1 万时，就可以视为是稠密点。需要考虑一些特殊的设计和处理。

> **Note**
>
> 悦数图数据库中没有专用的字段来记录每个点的出度和入度，也没有内置任务来进行统计，因此无法预知哪些点会是超级节点。一个折中的办法是使用 Spark 周期性地计算和统计。

重复属性索引

在属性图中，除了网络拓扑结构中的超级顶点，还有一类情况类似于超级顶点————某属性有极高重复率，也即"相同的 `点类型 Tag`，不同的 `顶点 VID`，同一属性字段，拥有相同属性值"。

悦数图数据库 3.5.1 属性索引的设计复用了存储模块 RocksDB 的功能，这种情况下的索引会被建模为 `前缀相同的 Key`。对于该属性的查找，（如果未能命中缓存，）会对应为硬盘上的“一次随机查找 + 一次前缀顺序扫描”，以找到对应的 `点 VID`（此后，通常会从该顶点开始图遍历，这样又会发生该点对应 Key-Value 的一次随机读+顺序扫描）。当重复率越高，扫描范围就越大。

关于属性索引的原理详细介绍在博客《分布式图数据库悦数图数据库的 Index 实践》。

经验上说，当重复属性值超过 1 万时，也需要特殊的设计和处理。

建议的办法

数据库端的常见办法

1. 截断：只访问一定阈值的边，超过该阈值的其他边则不返回。
2. Compact：重新组织 RocksDB 中数据的排列方式，减少随机读，增加顺序读。

Copyright © Yueshu Graph Technology - All rights reserved.

应用端的常见办法

根据业务意义，将一些超级顶点拆分：

- 删除多条边，合并为一条

  例如，一个转账场景： `（账户 A)-[转账]->（账户 B)` 。每次 `转账` 建模为 `一条 AB 之间的边` ，那么 `（账户 A)` 和 `（账户 B)` 之间会有着数万十次转账的场景。

  按日、周、或者月为粒度，合并陈旧的转账明细。也就是批量删除陈旧的边，改为少量的边“月总额”和“次数。而保留最近月的转账明细。

- 拆分相同类型的边，变为多种不同类型的边

  例如， `（机场）<-[depart]-（航班）` 场景，每个架次航班的离港，都建模为一条航班和机场之间的边。那么大型机场的离港航班会极多。

  根据不同的 `航空公司` 将 `depart` 这个 Edge type 拆分更细的 Edge type，如 `depart_ceair`, `depart_csair` 等。在查询（图遍历）时，指定离港的航空公司。

- 切分顶点本身

  例如，对于 `（人）-[借款]->（银行）` 的借款网络，某大型银行 A 的借款次数和借款人会非常的多。

  可以将该大行节点 A 拆分为多个相关联的子节点 A1、A2、A3,

  ```
  （人 1)-[借款]->（银行 A1)，（人 2)-[借款]->（银行 A2)，（人 2)-[借款]->（银行 A3);
  （银行 A1)-[属于]->（银行 A)，（银行 A2)-[属于]->（银行 A)，（银行 A3)-[属于]->（银行 A).
  ```

  这里的 A1、A2、A3 既可以是 A 真实的三个分行（例如北京、上海、浙江），也可以是三个按某种规则设立的虚拟分行，例如按借款金额划分 `A1: 1-1000, A2: 1001-10000, A3: 10000+` 。这样，查询时对于 A 的任何操作，都转变为为对于 A1、A2、A3 的三次单独操作。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

13.7 实践案例

悦数图数据库在各行各业都有应用，本文介绍部分实践案例。更多实践分享内容请参见博客。

13.7.1 业务场景

- 案例
- 经验
- 三方评测

13.7.2 内核

- MATCH 中变长 Pattern 的实现
- 如何向悦数图数据库增加一个测试用例
- 基于 BDD 理论的悦数图数据库集成测试框架重构（上）
- 基于 BDD 理论的悦数图数据库集成测试框架重构（下）
- 解析悦数图数据库子图设计及实践
- 基于全文搜索引擎的文本搜索
- 实操 | LDBC 数据导入及 nGQL 实践

13.7.3 周边工具

- 基于悦数图数据库Importer 批量导入工具性能验证方案总结
- 详解悦数图数据库 3.0 性能报告
- 悦数图数据库 支持 JDBC 协议
- Nebula·利器 | Norm 知乎开源的 ORM 工具
- 基于悦数图数据库的 Betweenness Centrality 算法
- 无依赖单机尝鲜 NebulaGraph Exchange 的 SST 导入
- logrotate 在悦数图数据库的日志滚动实践

视频

- Nebula 高性能图 schema 设计 by 青藤云安全（51 分 30 秒）

- 同花顺图数据库选型：消息面、基本面、技术面（21 分 53 秒）

- Nebula 在 Akulaku 智能风控的实践（40 分 03 秒）

- 从零到一：如何使用悦数图数据库构建一个企业股权图谱系统（09 分 34 秒）

- 美团图数据库平台建设及业务实践（上）（14 分 36 秒）

Copyright © Yueshu Graph Technology - All rights reserved.

- 美团图数据库平台建设及业务实践（下）（21 分 33 秒）
- 信息图谱在携程酒店的应用（39 分 06 秒）
- OPPO 图平台建设（43 分 09 秒）
- 悦数图数据库在网易游戏业务中的实践（47 分 40 秒）
- BIGO 数据管理与应用实践（53 分 47 秒）
- 悦数图数据库保险反欺诈解决方案 Demo 分享（39 分 54 秒）
- 悦数图数据库人际关系查询之疫情防控场景 Demo 分享（1 时 01 分 55 秒）
- 悦数图数据库在携程金融风控的应用（29 分 06 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

14. 客户端

14.1 客户端介绍

悦数图数据库提供多种类型客户端，便于用户连接、管理悦数图数据库图数据库。

- NebulaGraph Console：原生 CLI 客户端
- NebulaGraph CPP：C++ 客户端
- NebulaGraph Java：Java 客户端
- NebulaGraph Python：Python 客户端
- NebulaGraph Go：Go 客户端

Note

目前仅 NebulaGraph Java 客户端支持线程安全（thread-safe）。

Caution

以下客户端工具也可用于连接和管理悦数图数据库。他们由非常酷的社区用户提供和维护，欢迎大家参与测试和贡献。

- NebulaGraph PHP
- NebulaGraph Node
- NebulaGraph .net
- NebulaGraph JDBC
- NebulaGraph Carina（Python ORM）
- NORM（Golang ORM）
- Graph-Ocean（Java ORM）
- NebulaGraph Ngbatis（MyBatis 风格的 Java ORM）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

14.2 NebulaGraph Console

NebulaGraph Console 是悦数图数据库的原生命令行客户端，用于连接悦数图数据库集群并执行查询，同时支持管理参数、导出命令的执行结果、导入测试数据集等功能。

14.2.1 版本对照表

参见Github。

14.2.2 获取 NebulaGraph Console

NebulaGraph Console 的获取方式如下：

- 从 GitHub 发布页下载二进制文件。
- 编译源码获取二进制文件。编译方法参见 Install from source code。

14.2.3 功能说明

连接悦数图数据库

运行二进制文件 `nebula-console` 连接悦数图数据库的命令语法如下：

```
<path_of_console> -addr <ip> -port <port> -u <username> -p <password>
```

`path_of_console` 是 NebulaGraph Console 二进制文件的存储路径。

常用参数的说明如下。

参数	说明
`-h/-help`	显示帮助菜单。
`-addr/-address`	设置要连接的 Graph 服务的 IP 地址。默认地址为 127.0.0.1。
`-P/-port`	设置要连接的 Graph 服务的端口。默认端口为 9669。
`-u/-user`	设置悦数图数据库账号的用户名。未启用身份认证时，可以使用任意已存在的用户名（默认为 `root`）。
`-p/-password`	设置用户名对应的密码。未启用身份认证时，密码可以填写任意字符。
`-t/-timeout`	设置整数类型的连接超时时间。单位为毫秒，默认值为 120。
`-e/-eval`	设置字符串类型的 nGQL 语句。连接成功后会执行一次该语句并返回结果，然后自动断开连接。
`-f/-file`	设置存储 nGQL 语句的文件的路径。连接成功后会执行该文件内的 nGQL 语句并返回结果，执行完毕后自动断开连接。
`-enable_ssl`	连接悦数图数据库时使用 SSL 加密。
`-ssl_root_ca_path`	指定 CA 证书的存储路径。
`-ssl_cert_path`	指定 CRT 证书的存储路径。
`-ssl_private_key_path`	指定私钥文件的存储路径。

更多参数参见项目仓库。

例如，要连接到部署在 192.168.10.8 上的 Graph 服务，运行以下命令。

```
./nebula-console -addr 192.168.10.8 -port 9669 -u Joe -p Joespassword
```


Copyright © Yueshu Graph Technology - All rights reserved.

管理参数

NebulaGraph Console 可以保存参数，用于参数化查询。

Note

- VID 不支持参数化查询。
- `SAMPLE` 子句中不支持参数化查询。
- 会话释放后，参数不会保留。

- 保存参数命令如下：

```
nebula> :param <param_name> => <param_value>;
```

示例：

```
nebula> :param p1 => "Tim Duncan";
nebula> MATCH (v:player{name:$p1})-[:follow]->(n)  RETURN v,n;
+--------------------------------------------------+-----------------------------------------------------+
| v                                                | n                                                   |
+--------------------------------------------------+-----------------------------------------------------+
| ("player100" :player{age: 42, name: "Tim Duncan"}) | ("player125" :player{age: 41, name: "Manu Ginobili"}) |
| ("player100" :player{age: 42, name: "Tim Duncan"}) | ("player101" :player{age: 36, name: "Tony Parker"})   |
+--------------------------------------------------+-----------------------------------------------------+

nebula> :param p2 => {"a":3,"b":false,"c":"Tim Duncan"};
nebula> RETURN $p2.b AS b;
+-------+
| b     |
+-------+
| false |
+-------+
```

- 查看当前保存的所有参数，命令如下：

```
nebula> :params;
```

- 查看指定参数，命令如下：

```
nebula> :params <param_name>;
```

- 删除指定参数，命令如下：

```
nebula> :param <param_name> =>;
```

导出执行结果

导出命令执行的返回结果，可以保存为 CSV 文件、 DOT 文件或者 Profile/Explain 结果。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

- 文件保存在当前工作目录中，即 Linux 命令 `pwd` 显示的目录。
- 命令只对下一条查询语句生效。
- DOT 文件的内容可以复制后在 GraphvizOnline 网页中粘贴，生成可视化的执行计划图。

- 导出 CSV 文件命令如下：

```
nebula> :CSV <file_name.csv>;
```

- 导出 DOT 文件命令如下：

```
nebula> :dot <file_name.dot>;
```

示例：

```
nebula> :dot a.dot;
nebula> PROFILE FORMAT="dot" GO FROM "player100" OVER follow;
```

- 导出 PROFILE/EXPLAIN 结果到文件命令如下：

```
nebula> :profile <file_name>;
```

或者

```
nebula> :explain <file_name>;
```

Note

相比于 Studio 中的截图、CSV 文件，因为保有更多信息量和拥有更好的可读性，经由此命令输出的文本文件内容是首推的在 GitHub issue、论坛中报告执行计划、图查询调优的方式。

示例：

```
nebula> :profile profile.log
nebula> PROFILE GO FROM "player102" OVER serve YIELD dst(edge);
nebula> :profile profile.dot
nebula> PROFILE FORMAT="dot" GO FROM "player102" OVER serve YIELD dst(edge);
nebula> :explain explain.log
nebula> EXPLAIN GO FROM "player102" OVER serve YIELD dst(edge);
```

加载测试数据集

测试数据集名称为 basketballplayer，详细 Schema 信息和数据信息请使用相关 `SHOW` 命令查看。

加载测试数据集命令如下：

```
nebula> :play basketballplayer;
```

重复执行语句

重复执行下一个命令 N 次，然后打印平均执行时间。命令如下：

```
nebula> :repeat N;
```

示例：

Copyright © Yueshu Graph Technology - All rights reserved.

```
nebula> :repeat 3;
nebula> GO FROM "player100" OVER follow YIELD dst(edge);
+-------------+
| dst(EDGE)   |
+-------------+
| "player101" |
| "player125" |
+-------------+
Got 2 rows (time spent 2602/3214 us)

Fri, 20 Aug 2021 06:36:05 UTC

+-------------+
| dst(EDGE)   |
+-------------+
| "player101" |
| "player125" |
+-------------+
Got 2 rows (time spent 583/849 us)

Fri, 20 Aug 2021 06:36:05 UTC

+-------------+
| dst(EDGE)   |
+-------------+
| "player101" |
| "player125" |
+-------------+
Got 2 rows (time spent 496/671 us)

Fri, 20 Aug 2021 06:36:05 UTC

Executed 3 times, (total time spent 3681/4734 us), (average time spent 1227/1578 us)
```

睡眠

睡眠 N 秒。常用于修改 Schema 的操作中，因为修改 Schema 是异步实现的，需要在下一个心跳周期才同步数据。命令如下：

```
nebula> :sleep N;
```

断开连接

用户可以使用 `:EXIT` 或者 `:QUIT` 从悦数图数据库断开连接。为方便使用，NebulaGraph Console 支持使用不带冒号（`:`）的小写命令，例如 `quit` 。

示例：

```
nebula> :QUIT;

Bye root!
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

14.3 NebulaGraph CPP

NebulaGraph CPP 是一款 C++ 语言的客户端，可以连接、管理悦数图数据库图数据库。

14.3.1 使用限制

请确保已安装 C++ 且 GCC 版本为 4.8 及以上。

14.3.2 版本对照表

参见Github。

14.3.3 安装 NebulaGraph CPP

本文介绍通过编译方式安装 NebulaGraph CPP。

前提条件

- 准备正确的编译环境，详情请参见软硬件要求和安装三方库依赖包。
- 确保已安装 C++ 且 GCC 版本为：{10.1.0 | 9.3.0 | 9.2.0 | 9.1.0 | 8.3.0 | 7.5.0 | 7.1.0}。详情请参见 gcc_preset_versions 参数。

安装步骤

1. 克隆 NebulaGraph CPP 源码到机器。

- （推荐）如果需要安装指定版本的 NebulaGraph CPP，请使用选项 `--branch` 指定分支。例如安装 v3.4.0发布版本，请执行如下命令：

```
$ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula-cpp.git
```

- 如果需要安装日常开发版本，请执行如下命令下载 `master` 分支的源码：

```
$ git clone https://github.com/vesoft-inc/nebula-cpp.git
```

2. 进入目录 nebula-cpp。

```
$ cd nebula-cpp
```

3. 创建目录 build 并进入该目录。

```
$ mkdir build && cd build
```

4. 使用 CMake 生成 makefile 文件。

> **Note**
>
> 默认安装路径为 `/usr/local/nebula`，如果需要修改路径，请在下方命令内增加参数 `-DCMAKE_INSTALL_PREFIX=<installation_path>`。

```
$ cmake -DCMAKE_BUILD_TYPE=Release ..
```

> **Note**
>
> 如果 g++ 不支持 C++11，请添加选项 `-DDISABLE_CXX11_ABI=ON`。

5. 编译 NebulaGraph CPP。

Copyright © Yueshu Graph Technology - All rights reserved.

为了适当地加快编译速度，可以使用选项 `-j` 并行编译。并行数量 `N` 建议为\(\min(\text{CPU}核数，\frac{内存 (GB)}{2})\)。

```
$ make -j{N}
```

6. 安装 NebulaGraph CPP。

```
$ sudo make install
```

7. 更新动态链接库。

```
$ sudo ldconfig
```

14.3.4 使用方法

将 CPP 文件编译为可执行文件即可。接下来以 `SessionExample.cpp` 为例，介绍如何操作。

1. 使用示例代码创建 `SessionExample.cpp` 文件。
2. 编译文件，命令如下：

```
$ LIBRARY_PATH=<library_folder_path>:$LIBRARY_PATH g++ -std=c++11 SessionExample.cpp -I<include_folder_path> -lnebula_graph_client -o session_example
```

- `library_folder_path`： 悦数图数据库动态库文件存储路径，默认为 `/usr/local/nebula/lib64` 。
- `include_folder_path`： 悦数图数据库头文件存储路径，默认为 `/usr/local/nebula/include` 。

示例：

```
$ LIBRARY_PATH=/usr/local/nebula/lib64:$LIBRARY_PATH g++ -std=c++11 SessionExample.cpp -I/usr/local/nebula/include -lnebula_graph_client -o session_example
```

14.3.5 核心代码

NebulaGraph CPP 客户端提供 Session Pool 和 Connection Pool 两种方式连接悦数图数据库。使用 Connection Pool 需要用户自行管理 Session 实例。

- Session Pool

  详细示例请参见 SessionPoolExample。

- Connection Pool

  详细示例请参见 SessionExample。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

14.4 NebulaGraph Java

NebulaGraph Java 是一款 Java 语言的客户端，可以连接、管理悦数图数据库图数据库。

14.4.1 前提条件

已安装 Java，版本为 8.0 及以上。

14.4.2 版本对照表

参见Github。

14.4.3 下载 NebulaGraph Java

- （推荐）如果需要使用指定版本的 NebulaGraph Java，请使用选项 `--branch` 指定分支。例如使用 v3.5.0发布版本，请执行如下命令：

```
$ git clone --branch release-3.5 https://github.com/vesoft-inc/nebula-java.git
```

- 如果需要安装日常开发版本，请执行如下命令下载 `master` 分支的源码：

```
$ git clone https://github.com/vesoft-inc/nebula-java.git
```

14.4.4 使用方法

Note

建议一个线程使用一个会话，如果多个线程使用同一个会话，会降低效率。

使用 IDEA 等工具导入 Maven 项目，请在 `pom.xml` 中添加如下依赖：

Note

`3.0.0-SNAPSHOT` 为日常研发版本，可能存在未知问题，建议使用 release 版本号替换 `3.0.0-SNAPSHOT` 。

```
<dependency>
  <groupId>com.vesoft</groupId>
  <artifactId>client</artifactId>
  <version>3.0.0-SNAPSHOT</version>
</dependency>
```

如果无法下载日常研发版本的依赖，请在 `pom.xml` 中添加如下内容（release 版本不需要添加）：

```
<repositories>
  <repository>
    <id>snapshots</id>
    <url>https://oss.sonatype.org/content/repositories/snapshots/</url>
  </repository>
</repositories>
```

如果没有 Maven 管理项目，请手动下载 JAR 包进行安装。

Copyright © Yueshu Graph Technology - All rights reserved.

核心代码

NebulaGraph Java 客户端提供 Connection Pool 和 Session Pool 两种使用方式，使用 Connection Pool 需要用户自行管理 Session 实例。

- Session Pool

  详细示例请参见 GraphSessionPoolExample。

- Connection Pool

  详细示例请参见 GraphClientExample。

14.4.5 Java 相关库

以下由非常酷的社区用户提供和维护，欢迎大家参与测试和贡献。

- java-jdbc
- java-orm
- java-springboot demo
- ngbatis

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

14.5 NebulaGraph Python

NebulaGraph Python 是一款 Python 语言的客户端，可以连接、管理悦数图数据库图数据库。

14.5.1 前提条件

已安装 Python，版本为 3.6 及以上。

14.5.2 版本对照表

参见Github。

14.5.3 安装 NebulaGraph Python

pip 安装

```
$ pip install nebula3-python==<version>
```

克隆源码安装

1. 克隆 NebulaGraph Python 源码到机器。

- （推荐）如果需要安装指定版本的 NebulaGraph Python，请使用选项 `--branch` 指定分支。例如安装 v3.4.0发布版本，请执行如下命令：

```
$ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula-python.git
```

- 如果需要安装日常开发版本，请执行如下命令下载 `master` 分支的源码：

```
$ git clone https://github.com/vesoft-inc/nebula-python.git
```

2. 进入目录 nebula-python。

```
$ cd nebula-python
```

3. 执行如下命令安装。

```
$ pip install .
```

14.5.4 核心代码

NebulaGraph Python 客户端提供 Connection Pool 和 Session Pool 两种使用方式，使用 Connection Pool 需要用户自行管理 Session 实例。

- Session Pool

  详细示例请参见 SessinPoolExample.py。

  使用限制请参见 Example of using session pool。

- Connection Pool

  详细示例请参见 Example。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

14.6 NebulaGraph Go

NebulaGraph Go 是一款 Go 语言的客户端，可以连接、管理悦数图数据库图数据库。

14.6.1 前提条件

已安装 Go，版本为 1.13 及以上。

14.6.2 版本对照表

参见Github。

14.6.3 下载 NebulaGraph Go

- （推荐）如果需要下载指定版本的 NebulaGraph Go，请使用选项 `--branch` 指定分支。例如安装 v3.5.0发布版本，请执行如下命令：

```
$ git clone --branch release-3.5 https://github.com/vesoft-inc/nebula-go.git
```

- 如果需要安装日常开发版本，请执行如下命令下载 `master` 分支的源码：

```
$ git clone https://github.com/vesoft-inc/nebula-go.git
```

14.6.4 安装或更新

安装或更新的命令如下：

```
$ go get -u -v github.com/vesoft-inc/nebula-go@<tag>
```

`tag`： 指定分支。例如 `master` 或 `release-3.5` 。

14.6.5 核心代码

NebulaGraph GO 客户端提供 Connection Pool 和 Session Pool 两种使用方式，使用 Connection Pool 需要用户自行管理 Session 实例。

- Session Pool

  详细示例请参见 session_pool_example.go。

  使用限制请参见 Usage example。

- Connection Pool

  详细示例请参见 graph_client_basic_example 和 graph_client_goroutines_example。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

15. 悦数图数据库（公有云）

15.1 什么是悦数图数据库（公有云）

悦数图数据库（公有云）是一套集成了悦数图数据库和数据服务的云上服务，支持一键部署悦数图数据库和相关可视化产品。用户可以在几分钟内创建一个图数据库，并快速扩展计算、存储等资源。

Note

Cloud 当前支持的内核版本是 3.4.0，暂不支持 3.5.1。

15.1.1 产品功能

- 即买即用。用户可以根据自己的业务发展随时创建或停止悦数图数据库实例，业务提供充分的灵活性。
- 集成可视化图数据库管理工具悦数运维监控、悦数图探索。可以使用它们导入图数据集、执行 nGQL 语句查询、探索图数据、监控数据等。即使没有图数据库操作经验，也可以快速成为图专家。
- 用户可直接使用 Nebula Console 等客户端连接到云上数据库，而不需要先访问云平台。
- 权限控制。用户可以为指定图空间增加不同角色权限的数据库用户，保证业务数据安全。
- 官方支持。用户可以快速创建工单，咨询使用过程中遇到的问题。

15.1.2 产品优点

- 灵活与弹性：悦数图数据库（公有云）最大的优势在于灵活与弹性，无需自购软硬件即可快速（5-10分钟）搭建悦数图数据库。
- 高可用性：悦数图数据库（公有云）上创建的悦数图数据库集群采用 Raft 协议，结合云厂商提供的同城多机房实现数据冗余备份，保证云计算基础框架的高可用性、数据的可靠性以及云服务器的高可用性。
- 操作易用：悦数图数据库（公有云）提供 Web 页面对悦数图数据库集群进行在线管理，同时提供配套的可视化软件，降低用户使用门槛，快速上手。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

15.2 悦数图数据库（阿里云版）

15.2.1 创建服务实例

在使用悦数图数据库（阿里云版）前，要先在阿里云控制台创建服务实例。

账号与权限

- 准备阿里云账号。
- 如果使用 RAM 用户创建实例，需为其添加以下权限：
- AliyunECSFullAccess
- AliyunVPCFullAccess
- AliyunROSFullAccess
- AliyunCloudMonitorFullAccess
- AliyunComputeNestUserFullAccess

资源与费用

悦数图数据库（阿里云版）支持免费试用和付费使用，二者的详细说明如下表。

项目	免费试用版	付费版
云资源归属	阿里云官方账号	用户账号
云资源费用	阿里云承担	用户承担
云资源使用时长	30 天（不可续期）	用户创建实例时选择
悦数图数据库许可证费用	试用期内免费	用户承担
悦数图数据库服务许可证有效期	30 天	用户创建实例时选择
许可证过期后数据是否保留	否（因云资源会同时到期）	是
云资源到期后数据是否保留	否	否
数据盘快照费用	阿里云承担	用户承担（可关闭功能）

数据备份

在创建服务实例时，系统会默认开启数据盘自动快照备份，用于周期性备份悦数图数据库数据。

创建付费版服务实例前需开通快照。免费试用实例使用的是阿里云官方账号下的资源，已开通快照，无需用户手动开通。

默认的快照备份策略为每天 00:00 创建快照。

创建快照会产生费用，详情参见快照计费。

如需删除快照、修改或删除自动快照策略，参见快照文档。

Copyright © Yueshu Graph Technology - All rights reserved.

套餐版本

悦数图数据库（阿里云版）支持如下套餐版本。

套餐版本	说明
基础版	将所有悦数图数据库服务节点部署在 1 台 ECS 服务器上。
标准版	将 Graph 服务和 Storage 服务分别部署在不同的 ECS 服务器上，每个服务都是单节点（1 台 ECS）。将悦数图探索等生态工具混合部署在 1 台 ECS 服务器上。
高可用版	将 Graph 服务和 Storage 服务分别部署在不同的 ECS 服务器上，每个服务都包含 3 节点（3 台 ECS）。将悦数图探索等生态工具混合部署在 1 台 ECS 服务器上。

付费版和免费试用版服务实例支持的套餐版本有所不同，详情参见创建实例页面。

创建付费版服务实例

1. 登录阿里云控制台。
2. 打开云市场的 悦数图数据库产品页。
3. 选择套餐版本和购买时长（即悦数图数据库服务的许可证有效期），并单击立即购买。
4. 在创建服务实例页，保持选择模板处的选择不变。如需切换部署架构，可改变选中的选项，重新选择模板。

Note

改变模板会改变之前选择的套餐版本，软件费用（悦数图数据库许可证费用）和创建服务实例需要的资源也会改变。

5. （可选）设置服务实例名称。默认值为服务实例 ID。
6. 选择要创建服务实例的地域。
7. 在付费模式设置区域，指定 ECS 服务器的付费方式。默认为按量付费。

- 按量付费：按照计费周期计费，在每个结算周期生成账单并从账户中扣除相应费用。详情参见按量付费。
- 包年包月：先付费后使用。详情参见包年包月。选择包年包月模式需要指定购买时长周期和购买时长。购买时长周期当前仅支持 **Month**，即按月购买。

8. 在 悦数图数据库 配置区域，完成数据盘和 ECS 服务器密码设置。
9. 在基础设施配置区域，完成以下设置。

a. 选择专有网络 **VPC** 实例 **ID**。

如果下拉列表为空，先单击其右侧的新建专有网络，完成专有网络创建。

b. 选择交换机可用区。

c. 选择业务网络交换机的实例 **ID**。

如果下拉列表为空，先单击其右侧的新建交换机，完成交换机创建。

10. 完成权限确认，并选中我同意授权服务商（杭州悦数科技有限公司）获取上述权限以提供代运维服务。
11. 在页面底部，单击下一步：确认订单。
12. 在服务条款区域，勾选我已阅读并同意《商品在线协议》《云市场平台服务协议》《计算巢服务协议》。
13. 在页面底部，单击立即支付。
14. 在支付页面，选择支付方式，之后单击页面底部的支付。
15. 在支付完成页面，单击管理控制台。
16. 在已购买的服务页面，单击要创建的实例右侧的立即进入计算巢部署或查看部署详情。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

如果已购买的服务列表中未显示要创建的实例，刷新页面。

17. 在服务实例详情页查看实例的状态，确保状态为部署中。

    部署的平均耗时为 10 分钟。完成后实例的状态变为已部署。

创建免费版试用版服务实例

Danger

免费试用实例不可在生产环境中使用。

1. 登录阿里云计算巢推荐服务页面，搜索 悦数图数据库。
2. 在 悦数图数据库集群版卡片上，单击免费试用。
3. 系统会为首次创建悦数图数据库服务实例的用户弹出申请对话框。在申请权限对话框中填写申请人信息。带有红色星号（*）的为必填项。

Note

申请通过后需从第 1 步重新开始。

4. （可选）设置服务实例名称。默认值为服务实例 ID。
5. 选择要创建实例的地域。
6. 在 悦数图数据库配置区域，完成数据盘和 ECS 服务器密码设置。
7. 在基础设施配置区域，选择交换机可用区。
8. 完成权限确认，并选中我同意授权服务商（杭州悦数科技有限公司）获取上述权限以提供代运维服务。
9. 在页面底部，单击下一步：确认订单。
10. 在服务条款区域，勾选我已阅读并同意《计算巢服务协议》。
11. 在页面底部，单击开始免费试用。
12. 在提交成功页面，单击去列表查看。
13. 在实例列表中查看目标实例的状态，确保状态为部署中。

    部署的平均耗时为 10 分钟。完成后实例的状态变为已部署。

常见问题

Q：服务实例的状态显示为部署失败怎么处理?

1. 如果使用 RAM 账号创建的实例，确认为该账号授予了本文前提条件中指定的权限。
2. 如果权限符合要求，删除创建失败的实例，尝试重新创建。
3. 如果仍然创建失败，到 悦数图数据库论坛寻求帮助。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

15.2.2 使用服务实例

创建好服务实例后，即可查看服务信息，并开始使用悦数图数据库服务。

查看服务实例

在计算巢控制台可以查看服务实例的详细信息，详情参见查看服务实例。

查看连接信息

连接服务前，需要在阿里云计算巢的实例详情中查看各服务的连接地址。查看方式如下：

1. 登录服务实例管理页面。

> **Note**
>
> 如果要寻找的是免费试用版服务实例，需先在服务实例管理页面单击试用服务标签。

2. 在实例列表中，单击目标实例的服务实例**ID**，或其右侧操作列的详情。
3. 在概览标签页的基本信息区域，可以查看 Graph 服务、Storage 服务，以及悦数图探索、悦数运维监控等周边工具的连接信息。

连接悦数图数据库

悦数图数据库（阿里云版）提供多种连接方式。

使用客户端直连悦数图数据库

获取到连接地址后，即可使用悦数图数据库的各类客户端连接服务。

连接方式：

直连 Graph 服务的私网或公网 IP 地址。详细连接方式参见客户端文档。

如需通过公网连接悦数图数据库数据库：

1. 到 Graph 服务部署的 ECS 实例详情中找到其公网 IP 地址。
2. （仅付费版需执行该步骤）在悦数图数据库服务实例加入的安全组中，添加安全组规则，允许从指定的公网 IP 地址访问 TCP 9669 端口。

> **Note**
>
> 免费试用版默认允许从公网访问 Graph 服务的 9669 端口。

3. 使用 Graph 服务的公网 IP 地址和 9669 端口访问悦数图数据库。

> **Caution**
>
> 建议仅通过私网 IP 连接，提高网络和数据的安全性。

使用 EXPLORER 连接悦数图数据库

使用图探索工具可以快速连接悦数图数据库，与图数据进行可视化交互。

Copyright © Yueshu Graph Technology - All rights reserved.

连接方式:

1. 在浏览器中使用 **Explorer** 服务访问地址打开 Nebula Explorer。
2. 使用 Explorer 连接悦数图数据库。详情参见连接数据库。

监控悦数图数据库

在悦数图数据库（阿里云版）中部署悦数运维监控后，可以实时监控悦数图数据库服务的状态。详情参见 悦数运维监控文档。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

15.2.3 扩缩容服务

悦数图数据库（阿里云版）的弹性扩缩容功能支持根据需求扩缩容服务节点，灵活调整集群规模。

使用限制

- 弹性扩缩容功能支持 Storage 扩容，不支持 Storage 缩容。
- 使用弹性扩缩容功能缩容 Graph 服务时，需先使用集群中悦数运维监控服务的扩缩容功能删除不再需要的 Graph 服务节点。扩容 Graph 服务时无此限制。

操作步骤

1. 登录阿里云计算巢的服务实例管理页面。
2. 在服务实例列表中，单击目标实例的 ID 或其右侧操作列的详情。

> **Note**
>
> 免费试用版服务实例列表在试用服务标签页中。

3. 在服务实例详情页面，单击运维管理标签。
4. 在应用运维区域，单击弹性扩缩容。
5. 在弹性扩缩容对话框，完成如下操作。

a. 确认将要调整的资源类型为 **ESS**。

b. 选择伸缩组 **ID**。

- 如需扩缩容 Storage 服务，选择名称以 **ng-storage** 开头的伸缩组。
- 如需扩缩容 Graph 服务，选择名称以 **ng-graph** 开头的伸缩组。

c. 选择伸缩规则的调整方式。

- 如果选择扩容或缩容，下方的伸缩组 **ECS** 实例调整的数量指定的是要增加或减少的节点数量。
- 如果选择调整至指定数量，下方的伸缩组 **ECS** 实例调整的数量指定的是节点调整之后剩余的数量，而不是增加或减少的数量。

d. 输入伸缩组 **ECS** 实例调整的数量。

e. 单击下一步。

f. 检查任务信息，确认无误后单击创建。

6. 等候数分钟，直到运维管理标签页中的执行状态显示成功。
7. 单击概览标签页，查看服务资源区域的 ECS 实例数量，如果符合预期的调整效果，则扩缩容完成。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

15.2.4 删除服务实例

如果不再需要某个实例，可手动删除该实例。

删除方式参见删除服务实例。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16. 运维监控

16.1 什么是悦数运维监控

悦数运维监控是一款用于监控和管理悦数图数据库多集群中机器和服务状态的可视化工具。

16.1.1 产品功能

- 创建指定版本的悦数图数据库集群，支持批量导入节点、一键添加服务等功能。
- 支持在可视化界面进行集群导入、数据平衡、扩容缩容等操作。
- 支持管理多集群并可以查看集群的操作记录。
- 支持在可视化页面进行服务启动、停止、重启操作。
- 支持快速更新集群中 Storage 及 Graph 服务的配置。
- 支持全局调整监控数据的页面更新频率。
- 监控集群中所有服务的信息，包括服务 IP 地址、版本和监控指标（例如查询数量、查询延迟、心跳延迟等）。
- 监控集群中所有机器的状态，包括 CPU、内存、负载、磁盘和流量。
- 监控集群本身的信息，包括集群的服务信息、分区信息、配置和长时任务。
- 支持根据监控信息设置告警通知。

16.1.2 适用场景

- 针对大规模集群，需要可视化的运维监控平台。
- 需要方便快捷地监测关键指标，集中呈现业务的多个重点信息，保证业务正常运行。
- 需要多维度（例如时间段、聚合规则、指标）监控集群。
- 故障发生后，需要复盘问题，确认故障发生时间、异常现象。

16.1.3 注意事项

- 监控数据默认保留 14 天，即只能查询最近 14 天内任意时间段的监控数据。
- 只支持 2.5.0 及以上版本的悦数图数据库。
- 建议使用最新版本的 Chrome 访问悦数运维监控。
- 建议使用官方提供的安装包进行集群创建或导入。

Note

监控服务由 prometheus 提供，更新频率和保留时间等都可以自行修改。详情请参见 prometheus 官方文档。

Copyright © Yueshu Graph Technology - All rights reserved.

16.1.4 版本兼容性

悦数图数据库的版本和悦数运维监控的版本对应关系如下。

悦数图数据库版本	悦数运维监控版本
3.5.x	3.5.x
3.4.0 ~ 3.4.1	3.5.x、3.4.2、3.4.1、3.4.0、3.2.4、3.2.3、3.2.2、3.2.1、3.2.0
3.3.0	3.2.4、3.2.3、3.2.2、3.2.1、3.2.0
2.5.0 ~ 3.2.0	3.1.2、3.1.1、3.1.0
2.5.x ~ 3.1.0	3.0.4
2.5.1 ~ 3.0.0	1.1.0
2.0.1 ~ 2.6.1	1.0.2
2.0.1 ~ 2.6.1	1.0.1
2.0.1 ~ 2.6.1	1.0.0

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.2 部署悦数运维监控

本文将详细介绍如何安装并部署悦数运维监控。

16.2.1 前提条件

在部署悦数运维监控之前，用户需要确认以下信息：

- 选择并下载符合版本的悦数运维监控。关于悦数运维监控和悦数图数据库的版本对应关系，参见版本兼容性。
- 悦数运维监控支持 SQLite 和 MySQL 存储悦数运维监控元数据。如果用户使用 MySQL，需准备版本为 5.7 以上的 MySQL 环境，创建名称为 `dashboard` 的数据库，字符集为 `utf8` 。

  Note

  悦数运维监控内置了 SQLite 数据库。如果使用 SQLite 存储悦数运维监控元数据，则无需准备 SQLite 环境。

- 确保在安装开始前，以下端口处于未被使用状态。

端口号	说明
7005	悦数运维监控提供 web 服务的端口。
9091	Prometheus 服务的端口。
9200	nebula-stats-exporter 服务的端口。
9093	Alertmanager 服务的端口。接收 Prometheus 告警，发送告警通知给悦数运维监控。
9100	node-exporter 的服务端口。在集群创建后，自动部署在目标机器内，收集集群中机器的资源信息，包括 CPU、内存、负载、磁盘和流量。

16.2.2 使用 TAR 包部署

安装

1. 根据需要下载 TAR 包，建议选择最新版本。
2. 使用 `tar -xzvf` 解压 TAR 包。

```
tar -xzvf yueshu-dashboard-<version>.linux-amd64.tar.gz -C <install_path>
```

例如：

```
tar -xzvf yueshu-dashboard-3.5.1.linux-amd64.tar.gz -C /usr/local/
```

3. （可选）进入解压后的文件夹，在 `etc` 目录内修改 `config.yaml` 文件，设置相关配置。

```
Name: dashboard-api
Host: 0.0.0.0    # 指定能访问悦数运维监控的地址。
Port: 7005  # 访问悦数运维监控的默认端口。
MaxBytes: 1073741824 # Http 可接受请求的最大 ContentLength，默认为 1048576。取值范围：0 ~ 8388608。
Timeout: 60000  # 访问超时时间。
Log:     # 悦数运维监控运行日志设置。
  KeepDays: 7    # 日志保留天数。
  Mode: file   # 日志保存方式，支持 console 和 file。console 表示服务日志会记录在 webserver.log里；file 表示服务日志会分别记录在 access.log、error.log、severe.log、slow.log 和 stat.log 里。
  Encoding: plain # 日志编码方式，支持 plain 和 json。
Database:
  Dialect: sqlite  # 使用的数据库类型，目前支持 SQLite 和 MySQL，默认使用 SQLite。
  AutoMigrate: true # 自动创建数据库表，默认为 true。
  Host: 127.0.0.1 # 连接的 MySQL 数据库的 IP 地址。
  Port: 3306  # 连接的 MySQL 数据库的端口号。
  Username: root # 登录 MySQL 的账户。
  Password: nebula # 登录 MySQL 的密码。
  Name: dashboard # 对应的数据库名称。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# exporter 端口信息
Exporter:
  NodePort: 9100 # node-exporter 服务的端口。
  NebulaPort: 9200 # nebula-stats-exporter 服务的端口。

# 服务信息
Proxy:
  PrometheusAddr: 127.0.0.1:9091 # prometheus 服务的 IP 地址和端口。
  AlertmanagerAddr: 127.0.0.1:9093 # alertmanager 服务的 IP 地址和端口。

# 邀请 LDAP 账号的发件人邮箱信息
Mail:
  Host: smtp.office365.com #  SMTP 服务器地址。
  Port: 587 # SMTP 服务器的端口号。
  Username: "" # SMTP 服务器的用户名。
  Password: "" # SMTP 服务器的密码。

# 慢查询设置
SlowQuery:
  Enable: true # 慢查询数据轮询开关。
  MessageStore: 14 # 慢查询数据存储时长。单位：天。
  ScrapeInterval: 2m # 拉取慢查询数据的间隔时间，例如:1s、10s、2m、3h。

# 系统信息
System:
  WebAddress: http://127.0.0.1:7005 # 外网访问悦数运维监控的地址，也可以设置为域名。用于一些接口回调，例如邮件被邀请人可以使用此链接访问悦数运维监控。
  MessageStore: 90 # 保存告警消息的天数，默认 90 天。

CloudProvider: "" # 云提供商名称。

LicenseManagerURL: "" # License Manager 所在的主机 IP，端口默认为9119。
```

4. 启动悦数运维监控。

   可以使用以下命令一键启动悦数运维监控。

```
cd /usr/local/yueshu-dashboard/scripts
sudo ./dashboard.service start all
```

   或是执行以下命令，分别启动 Prometheus、nebula-dashboard-server、nebula-stats-exporter 和 Alertmanager 依赖服务以启动悦数运维监控。

```
cd scripts
sudo ./dashboard.service start prometheus # 启动 Prometheus 服务
sudo ./dashboard.service start webserver #启动 nebula-dashboard-server 服务
sudo ./dashboard.service start exporter #启动 nebula-stats-exporter 服务
sudo ./dashboard.service start alertmanager #启动 Alertmanager 服务
```

Note

在启动悦数运维监控后，如再修改配置文件的话，在修改后，可在 `scripts` 目录下执行 `dashboard.service restart all` 同步修改至悦数运维监控客户端页面中。

16.2.3 使用 RPM 包部署

前提条件

使用的 Linux 发行版为 CentOS ，已安装 lsof。

安装

1. 下载 RPM 包。
2. 使用 `sudo rpm -ivh <package_name>` 命令安装 RPM 包。

   例如，安装悦数运维监控需要运行以下命令，默认安装路径为 `/usr/local/yueshu-dashboard`：

```
sudo rpm -ivh yueshu-dashboard-<version>.x86_64.rpm
```

   也可以使用以下命令安装到指定路径：

```
sudo rpm -ivh yueshu-dashboard-xxx.rpm --prefix=<path>
```

3. （可选）进入解压后的文件夹，在 `etc` 目录内修改 `config.yaml` 文件，设置相关配置。

Copyright © Yueshu Graph Technology - All rights reserved.

```
Name: dashboard-api
Host: 0.0.0.0    # 指定能访问悦数运维监控的地址。
Port: 7005  # 访问悦数运维监控的默认端口。
MaxBytes: 1073741824 # Http 可接受请求的最大 ContentLength，默认为 1048576。取值范围：0 ~ 8388608。
Timeout: 60000  # 访问超时时间。
Log:     # 悦数运维监控运行日志设置。
  KeepDays: 7    # 日志保留天数。
  Mode: file   # 日志保存方式，支持 console 和 file。console 表示服务日志会记录在 webserver.log里；file 表示服务日志会分别记录在 access.log、error.log、severe.log、slow.log 和 stat.log 里。
  Encoding: plain # 日志编码方式，支持 plain 和 json。
Database:
  Dialect: sqlite  # 使用的数据库类型，目前支持 SQLite 和 MySQL，默认使用 SQLite。
  AutoMigrate: true # 自动创建数据库表，默认为 true。
  Host: 127.0.0.1 # 连接的 MySQL 数据库的 IP 地址。
  Port: 3306  # 连接的 MySQL 数据库的端口号。
  Username: root # 登录 MySQL 的账户。
  Password: nebula # 登录 MySQL 的密码。
  Name: dashboard # 对应的数据库名称。

# exporter 端口信息
Exporter:
  NodePort: 9100 # node-exporter 服务的端口。
  NebulaPort: 9200 # nebula-stats-exporter 服务的端口。

# 服务信息
Proxy:
  PrometheusAddr: 127.0.0.1:9091 # prometheus 服务的 IP 地址和端口。
  AlertmanagerAddr: 127.0.0.1:9093 # alertmanager 服务的 IP 地址和端口。

# 邀请 LDAP 账号的发件人邮箱信息
Mail:
  Host: smtp.office365.com #  SMTP 服务器地址。
  Port: 587 # SMTP 服务器的端口号。
  Username: "" # SMTP 服务器的用户名。
  Password: "" # SMTP 服务器的密码。

# 慢查询设置
SlowQuery:
  Enable: true # 慢查询数据轮询开关。
  MessageStore: 14 # 慢查询数据存储时长。单位：天。
  ScrapeInterval: 2m # 拉取慢查询数据的间隔时间，例如:1s、10s、2m、3h。

# 系统信息
System:
  WebAddress: http://127.0.0.1:7005 # 外网访问悦数运维监控的地址，也可以设置为域名。用于一些接口回调，例如邮件被邀请人可以使用此链接访问悦数运维监控。
  MessageStore: 90 # 保存告警消息的天数，默认 90 天。

CloudProvider: "" # 云提供商名称。

LicenseManagerURL: "" # License Manager 所在的主机 IP，端口默认为9119。
```

4. 使用以下命令查看所有服务状态、启动服务。

```
sudo systemctl list-dependencies nebula-dashboard.target  # 查看状态
sudo systemctl start nebula-dashboard.target   # 启动服务
```

用户也可以执行以下命令查看、启停单一服务。

```
sudo systemctl {status|stop|start}  {nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service}
```

卸载

执行以下命令卸载通过 RPM 包部署的悦数运维监控。

```
sudo rpm -e <package_name>
```

16.2.4 使用 DEB 包部署

安装

1. 下载 DEB 包。
2. 使用 `sudo dpkg -i <package_name>` 命令安装 DEB 包。

例如，安装悦数运维监控需要运行以下命令，默认安装路径为 `/usr/local/yueshu-dashboard`：

```
sudo dpkg -i yueshu-dashboard-3.5.1.ubuntu1804.amd64.deb
```


Copyright © Yueshu Graph Technology - All rights reserved.

Note

使用 DEB 包安装悦数运维监控时不支持自定义安装路径。

3. （可选）进入解压后的文件夹，在 `etc` 目录内修改 `config.yaml` 文件，设置相关配置。

```
Name: dashboard-api
Host: 0.0.0.0    # 指定能访问悦数运维监控的地址。
Port: 7005  # 访问悦数运维监控的默认端口。
MaxBytes: 1073741824 # Http 可接受请求的最大 ContentLength，默认为 1048576。取值范围：0 ~ 8388608。
Timeout: 60000  # 访问超时时间。
Log:     # 悦数运维监控运行日志设置。
  KeepDays: 7    # 日志保留天数。
  Mode: file   # 日志保存方式，支持 console 和 file。console 表示服务日志会记录在 webserver.log里；file 表示服务日志会分别记录在 access.log、error.log、severe.log、slow.log 和 stat.log 里。
  Encoding: plain # 日志编码方式，支持 plain 和 json。
Database:
  Dialect: sqlite  # 使用的数据库类型，目前支持 SQLite 和 MySQL，默认使用 SQLite。
  AutoMigrate: true # 自动创建数据库表，默认为 true。
  Host: 127.0.0.1 # 连接的 MySQL 数据库的 IP 地址。
  Port: 3306  # 连接的 MySQL 数据库的端口号。
  Username: root # 登录 MySQL 的账户。
  Password: nebula # 登录 MySQL 的密码。
  Name: dashboard # 对应的数据库名称。

# exporter 端口信息
Exporter:
  NodePort: 9100 # node-exporter 服务的端口。
  NebulaPort: 9200 # nebula-stats-exporter 服务的端口。

# 服务信息
Proxy:
  PrometheusAddr: 127.0.0.1:9091 # prometheus 服务的 IP 地址和端口。
  AlertmanagerAddr: 127.0.0.1:9093 # alertmanager 服务的 IP 地址和端口。

# 邀请 LDAP 账号的发件人邮箱信息
Mail:
  Host: smtp.office365.com #  SMTP 服务器地址。
  Port: 587 # SMTP 服务器的端口号。
  Username: "" # SMTP 服务器的用户名。
  Password: "" # SMTP 服务器的密码。

# 慢查询设置
SlowQuery:
  Enable: true # 慢查询数据轮询开关。
  MessageStore: 14 # 慢查询数据存储时长。单位：天。
  ScrapeInterval: 2m # 拉取慢查询数据的间隔时间，例如:1s、10s、2m、3h。

# 系统信息
System:
  WebAddress: http://127.0.0.1:7005 # 外网访问悦数运维监控的地址，也可以设置为域名。用于一些接口回调，例如邮件被邀请人可以使用此链接访问悦数运维监控。
  MessageStore: 90 # 保存告警消息的天数，默认 90 天。

CloudProvider: "" # 云提供商名称。

LicenseManagerURL: "" # License Manager 所在的主机 IP，端口默认为9119。
```

4. 使用以下命令查看所有服务状态、启动服务。

```
sudo systemctl list-dependencies nebula-dashboard.target  # 查看状态
sudo systemctl start nebula-dashboard.target   # 启动服务
```

用户也可以执行以下命令查看、启停单一服务。

```
sudo {systemctl status|stop|start}  {nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service}
```

卸载

执行以下命令卸载通过 DEB 包部署的悦数运维监控。

```
sudo dpkg -r <package_name>
```


Copyright © Yueshu Graph Technology - All rights reserved.

16.2.5 管理悦数运维监控服务

以下内容介绍了两种管理悦数运维监控服务的方式。需要注意的是，两种方式不能混合使用。例如，不能先使用 `dashboard.service` 脚本启动，再用 `systemctl` 命令停止。

- 悦数运维监控使用脚本 `dashboard.service` 管理服务，包括启动、重启、停止和查看状态。

```
sudo <dashboard_path>/scripts/dashboard.service
[-v] [-h] [-version]
<start|restart|stop|status>  <prometheus|webserver|exporter|gateway|all>
```

参数	说明
`dashboard_path`	悦数运维监控安装路径。
`-v`	显示详细调试信息。
`-h`	显示帮助信息。
`-version`	显示悦数运维监控版本。
`start`	启动服务。
`restart`	重启服务。
`stop`	停止服务。
`status`	查看服务状态。
`prometheus`	管理 prometheus 服务。
`webserver`	管理 webserver 服务。
`exporter`	管理 exporter 服务。
`gateway`	管理 gateway 服务。
`all`	管理所有服务。

例如， Dashboard 的安装在当前目录下，用户可以用以下命令管理服务：

```
sudo /dashboard/scripts/dashboard.service start all # 启动悦数运维监控所有服务。
sudo /dashboard/scripts/dashboard.service stop all # 停止悦数运维监控所有服务。
sudo /dashboard/scripts/dashboard.service status all # 查看悦数运维监控所有服务状态。
sudo /dashboard/scripts/dashboard.service restart all # 重启悦数运维监控所有服务状态。
```

- 使用 RPM 包或 DEB 包安装的用户，还可以使用 systemd 管理服务。通过 `systemctl` 命令进行服务启动、查看、重启和停止。

```
sudo systemctl start nebula-dashboard.target # 启动服务
sudo systemctl status nebula-dashboard.target # 查看服务状态
sudo systemctl restart <nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service> # 分别重启服务
sudo systemctl stop <nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service> # 分别停止服务
```

例如，关闭 Prometheus 服务：

```
sudo systemctl stop nbd-prometheus.service
```

16.2.6 查看日志

- 使用脚本 `dashboard.service` 管理服务的用户可以在 `logs` 目录内查看悦数运维监控的日志。

  例如：

```
cat logs/prometheus.log
```


Copyright © Yueshu Graph Technology - All rights reserved.

日志文件说明如下。

日志文件名	说明
`alertmanager.log`	Alertmanager 服务日志。
`nebula-stats-exporter.log`	nebula-stats-exporter 服务日志。
`prometheus.log`	Prometheus 服务日志。
`br`	备份恢复日志。
`webserver.log`	悦数运维监控服务日志。悦数运维监控的配置 `Log.Mode` 为 `console` 时生效。
`access.log`	访问日志。悦数运维监控的配置 `Log.Mode` 为 `file` 时生效。
`error.log`	错误日志。悦数运维监控的配置 `Log.Mode` 为 `file` 时生效。
`severe.log`	严重错误日志。悦数运维监控的配置 `Log.Mode` 为 `file` 时生效。
`slow.log`	慢日志。悦数运维监控的配置 `Log.Mode` 为 `file` 时生效。
`stat.log`	统计日志。悦数运维监控的配置 `Log.Mode` 为 `file` 时生效。

- 使用 systemd 管理服务的用户，可以通过 `journalctl` 查看悦数运维监控中各个服务的日志。

```
journalctl -u {nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service} -b
```

例如，查看 Prometheus 服务的日志：

```
journalctl -u nbd-prometheus.service -b
```

16.2.7 后续操作

连接悦数运维监控

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.3 连接悦数运维监控

悦数运维监控部署完成后，可以通过浏览器登录使用悦数运维监控。

16.3.1 前提条件

- 悦数运维监控相关服务已经启动。详情请参见部署悦数运维监控。
- 建议使用 Chrome 89 及以上的版本的 Chrome 浏览器，否则可能有兼容问题。

16.3.2 连接悦数运维监控

1. 确认悦数运维监控服务所在机器的 IP 地址，在浏览器中输入 `http://<ip_address>:7005` 打开登录页面。

   如在浏览器窗口中看到以下登录界面，则表示已经成功部署并启动了悦数运维监控。

Note

首次登录悦数运维监控的时候，页面显示最终用户许可协议的内容，请仔细阅读并单击同意。

2. 使用初始化用户名 `nebula` 和密码 `nebula` 登录悦数运维监控。

Note

登录悦数运维监控后可创建 LDAP 账户、OAuth2.0 账户和普通账户，然后使用不同的账户登录悦数运维监控。关于账户的详情，参见权限管理。

3. 设置悦数 License 以使用悦数运维监控。详情请参见下文激活悦数运维监控。

Copyright © Yueshu Graph Technology - All rights reserved.

16.3.3 激活悦数运维监控

在使用悦数运维监控前，需配置 LM 以校验悦数 License 的有效性从而激活悦数运维监控。

用户可使用以下两种方式配置 LM 以正常使用悦数运维监控：

- 通过悦数运维监控配置 LM：

  悦数运维监控目前支持可视化一键部署 LM。用户登录悦数运维监控后，按照以下步骤完成 LM 的部署后开始使用悦数运维监控。

  1. 在 LM 介绍页面单击开始：页面介绍了 LM 的相关概念和整体流程，即安装 LM、生成 License Key 和运行 LM。单击开始继续。
  2. 填写配置信息及安装 LM：需要填写主机 IP、SSH 相关信息、安装位置和 LM 端口号等。填写完成后单击安装 **LM** 即可进行安装。
  3. 查看 LMID 和获取 License Key：成功安装 LM 后，会生成 LMID，用户可复制 LMID 并提供给悦数销售人员，悦数会为用户生成 License Key。用户需要在界面中填写 License Key，然后单击确认完成 License 的配置。

  > **Note**
  >
  > 如果用户已经有 License Key，可点击右上角设置 **LM URL**直接填写 LM URL，然后单击确认完成 License 的配置。

- 通过安装包配置 LM：

a. 安装、启动 LM 并在 LM 中加载 License Key。详情请参见使用 License Manager。

b. 在悦数运维监控的安装路径下，进入 `etc` 目录，修改 `config.yaml` 文件，设置 `LicenseManagerURL` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.xxx:9119`。用户还可以登录悦数运维监控后，在 LM 服务管理页面的设置 **LM URL**中，设置 `LicenseManagerURL` 的值。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.4 创建及导入集群

16.4.1 创建集群

本文介绍如何通过悦数运维监控创建集群。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

按以下方式创建集群:

Copyright © Yueshu Graph Technology - All rights reserved.

1. 在悦数运维监控页面顶部，单击集群管理。
2. 在集群管理页面，单击创建集群标签。

Copyright © Yueshu Graph Technology - All rights reserved.

3. 在创建集群页面，完成以下配置：

- 输入集群名称，最大可输入 15 个字符。
- 选择悦数图数据库安装版本。

> **Note**
>
> 悦数运维监控默认内置企业版的悦数图数据库安装包。如果用户需要安装其他版本的悦数图数据库，可以在安装包管理页面下载或上传，详情参见安装包管理。

> **Note**
>
> 创建 3.5.0 以下版本的集群时，需要手动上传 License 证书。

Copyright © Yueshu Graph Technology - All rights reserved.

- 添加节点信息，提供两种方式：
- 添加节点，输入待添加节点的 Host 信息、SSH 端口号、SSH 用户、认证方式、安装包等。

  关于认证方式的说明如下：
- SSH 密码：输入 SSH 用户对应的密码。
- SSH 密钥：单击上传，选择节点的私钥文件。需要提前在待添加节点上生成密钥文件，并将私钥发送给当前电脑（非悦数运维监控机器）。如果设置了短密码（passphrase），也需要填写。

- 批量导入节点，批量添加多个节点的相关信息。批量导入需要先选择安装包后单击下载 **CSV** 模版。按照模版填写后上传，尽量确保节点信息正确，否则容易造成上传失败。

4. 勾选节点并在节点右上方，单击需要添加的服务。创建集群需要给节点添加 3 种类型的服务。如果不熟悉悦数图数库架构，建议单击自动添加服务按钮。

Copyright © Yueshu Graph Technology - All rights reserved.

5. （可选）在下方的服务中，选择编辑 Meta、Graph、Storage 服务的端口号、HTTP 端口号，单击确认保存。
6. 单击创建集群，确定配置信息无误且节点无冲突后，单击确认。
7. 在集群管理页面中的列表中出现状态为 `installing` 的集群，需等待 3-10 分钟，状态变为 `healthy` 即集群创建成功。如果服务状态为 `unhealthy` ，表示集群服务中存在非正常运行的服务，请单击详情进行查看。

后续操作

成功创建集群后，用户可以对集群进行操作，详情见集群操作。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.4.2 导入集群

本文介绍如何通过悦数运维监控导入集群。当前版本仅支持官方下载的 DEB、RPM 包部署的集群和悦数运维监控创建的集群导入，暂不支持导入使用 Docker 和 Kubernetes 方式部署的集群。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

Caution

在同一集群下，服务需要统一版本。不支持在同一集群中导入不同版本的悦数图数据库实例。

1. 在各个服务的配置文件中，将 `<meta|graph|storage>_server_addrs` 和 `local_ip` 中的IP地址改为本机的IP地址，然后启动服务。

   具体步骤，参见配置管理和管理服务。

2. 在集群管理页面，单击导入集群标签。
3. 在导入集群页面，输入连接悦数图数据库的信息：

- Graphd Host：<其中一个 Graphd 进程的虚机 IP>:<端口号>。本示例设置为 `192.168.8.157:9669` 。
- 用户名：连接悦数图数据库的账号，本示例设置为 `vesoft` 。
- 密码：连接悦数图数据库的密码，本示例设置为 `nebula` 。

Note

因为悦数图数据库默认不启用身份验证，所以，一般情况下用户可以使用 `root` 账号和任意密码连接悦数图数据库。 当悦数图数据库启用了身份验证后，用户只能使用指定的账号和密码连接悦数图数据库。关于悦数图数据库的身份验证功能，参考 悦数图数据库用户手册。

4. 在连接服务节点页面完成以下配置：

- 输入集群的名称，最大可输入 15 个字符，本示例设置为 `create_1027` 。
- 对节点进行授权，授权需输入每个节点的 SSH 用户名及密码，以及选择是否使用 `sudo` 执行命令。

Notice

如果用户 SSH 账号没有操作悦数图数据库的权限，但是可以免密执行 `sudo` 命令，需要选择使用 **sudo**。

- 批量授权需要上传 CSV 文件。请根据下载的 CSV 文件，编辑每个节点授权信息，尽量确保节点信息正确，否则容易造成上传失败。
- 页面中节点状态变为已授权，则该节点授权成功。

5. 确保所有节点都授权成功，单击导入集群。

Copyright © Yueshu Graph Technology - All rights reserved.

后续操作

成功导入集群后，用户可以对集群进行操作，详情见总览。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5 集群管理

16.5.1 集群总览

本文主要介绍悦数运维监控的集群总览页面。

悦数运维监控的集群总览页面可选择以下部分进行展示：

- 集群信息概览
- 集群拓扑
- 监控大屏

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 在目标集群右侧详情，即可进入指定集群的集群总览页面。

集群信息概览

悦数运维监控的集群信息概览页面为集群总览的默认页面，分为集群关键信息和集群监控面板两部分。

集群关键信息

在集群关键信息中，展示如下信息：

- 集群健康分：按百分制展示集群的健康程度。每 5 分钟刷新一次，若出现 emergency 级别的告警则立刻刷新。计算公式可以在集群诊断页面进行配置。
- 监控中的节点和服务：展示在线节点数/总节点数以及在线服务数/总服务数。不同类型的服务分别展示。单击 进入服务管理页面。
- 慢查询数量：集群中的慢查询数量。
- 新打开 session 数量：5 分钟内建立的 session 总数。
- 最近备份时间：集群中最近一次备份的时间。单击 进入备份恢复页面。
- 总存储使用量：集群中所有节点的存储使用量之和与总空间的比例。单击 显示具体节点的存储使用量。

集群监控面板

在集群监控面板中，可选择不同时间段的监控数据进行查看。支持选择时间段，也可以快捷选择最近 5 分钟、 1 小时、6 小时、12 小时、1 天和 3 天。监控面板展示如下信息：

- QPS：每秒查询次数。
- 查询延迟（P95）：查询响应耗时从小到大排列，顺序处于 95% 位置的数值。
- CPU使用率：CPU 已使用百分比。
- 内存使用率：内存已使用百分比。
- 打开会话数：当前打开的会话数量。
- 慢查询数：慢查询次数。

Copyright © Yueshu Graph Technology - All rights reserved.

单击每个子面板右上方的 可快速跳转至该监控页面，以查看详情信息。

集群拓扑

悦数运维监控的集群拓扑页面以拓扑图的形式展示集群中的节点和服务分布及状态。单击页面上方的集群拓扑进入集群拓扑页面。

可视化监控大屏

悦数运维监控的可视化监控大屏帮助用户一目了然地把握集群的健康状态、服务及节点信息。

单击页面右上方的切换到监控大屏进入监控大屏页面。

Copyright © Yueshu Graph Technology - All rights reserved.

大屏区域	显示信息
中间区域上方	1. 集群的健康度，为集群的健康打分。关于打分规则，参见下文 Note 说明。 2. 运行的节点信息和数量、集群中运行中的服务个数和异常服务的个数。 3. 当前时间的节点 CPU 和内存使用率。 4. 告警信息。系统按照告警消息的严重级别（emergency>critical>warning）展示最近触发的 5 条告警消息。更多信息，参见监控告警。
中间区域下方	在不同时间段 4 个 Graph 服务指标的监控信息。4 个指标分别是： 1. num_active_sessions 2. num_slow_queries 3. num_active_queries 4. num_query_errors
左侧区域	1. 集群的每秒查询率 QPS (Query Per Second)。 2. 在不同时间段两个 Storage 服务指标的监控信息。两个指标分别是 add_edges_latency_us、add_vertices_latency_us。
右侧区域	在不同时间段节点相关指标的信息，指标包括： 1. cpu_utilization 2. memory_utilization 3. load_1m 4. disk_readbytes 5. disk_writebytes

有关监控指标的具体信息，参见监控指标说明。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

集群的健康打分规则如下：

- 最高 100 分，最低 13 分。
- 当 100≥健康度≥80，分数为蓝色；当 80＞健康度≥60，分数为黄色；当健康度＜60，分数为红色。
- 算法：(1-异常服务数/总服务数)*100%。
- 除第一个出现的 `emergency` 级别的告警扣 40 分外，其余 `emergency` 级别及其他级别的告警每个扣 10 分。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.2 集群监控

本文主要介绍悦数运维监控集群管理中的集群监控以及自定义监控功能。

在悦数运维监控顶部导航栏，单击集群管理，在目标集群右侧单击详情，在左侧导航栏点击集群监控即可进入。

默认集群监控

悦数运维监控的默认集群监控页面展示了 8 个监控指标面板的情况，不支持修改或删除指标面板，用户可以通过下文介绍的自定义集群监控功能在新建的 Dashboard 中添加、修改和删除监控指标面板。

默认展示的 8 个监控指标如下：

- CPU Utilization：CPU 已使用百分比。
- Memory Utilization：内存已使用百分比。
- Disk Usage：磁盘使用情况，显示实例、磁盘名称、挂载点和已使用情况。
- Load 5：最近 5 分钟系统平均负载。
- QPS：每秒查询次数。
- Num Queries & Slow Queries：查询次数和慢查询次数。
- Num Opened Sessions：已打开的会话数。
- Add Edge Latency & Add Vertex Latency：添加边的延迟时间和添加顶点的延迟时间。

单击每个面板右上方的 可分别进入监控指标的子页面单独筛选时间段进行查看。

自定义集群监控

创建集群监控

1. 单击新建 **Dashboard**，在弹出的对话框中输入自定义 Dashboard 名称，单击确定即可创建一个新的集群监控。
2. 单击右上方的添加面板或正中的新建面板添加监控指标面板，设置参数后单击确认。参数说明如下：

Copyright © Yueshu Graph Technology - All rights reserved.

- 面板名称：自定义面板名称。
- 面板类型：分为节点线性图面板、服务线性图面板和磁盘直方图面板。不同的面板类型决定了后续可供选择的指标。
- 指标：根据面板类型选择不同的指标。例如，选择节点线性图面板，可以选择 CPU、Memory、Load、Disk 和 Network In/Out 等相关指标。支持指标多选。监控指标详情请参见监控指标说明。
- 监控指标类型：可选择百分比、字节、字节/秒、io/秒、数值等。
- 展示顺序：面板展示顺序，数字越小越靠前，最小值为1。默认根据创建时间排序。

Note

Graph 服务支持一系列基于图空间的监控指标。当选中 Graph 服务相关指标时，可在增加的图空间下拉框中选择对应的图空间。

Caution

使用图空间指标前，用户需要在 Graph 服务中，设置 `enable_space_level_metrics` 为 `true` 。具体操作，参见更新配置。

管理监控指标面板

单击每个面板右上方的 按钮可修改或删除该监控指标面板。

可以单击面板右上方的 进入监控指标的子页面进行查看。

Copyright © Yueshu Graph Technology - All rights reserved.

其他操作

页面上方的部分用于进行监控筛选，具体功能如下：

- 时间段选择：默认选择 6 小时的监控数据进行查看。支持根据日期选择时间段，也可以快捷选择最近 5 分钟、 1 小时、6 小时、12 小时、1 天和 3 天。
- 节点选择：默认勾选显示所有节点的监控数据，可以在节点框内调整。
- 服务选择：默认勾选显示所有服务的监控数据，可以在服务框内调整。
- 数据更新：页面的监控数据默认不自动更新，即默认状态为 `Off`，可以在下拉框中设置页面更新频率，例如每 5 秒、30 秒或 5 分钟等。单击 按钮可以手动更新。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.3 集群操作

节点管理

节点管理页面展示所有的节点详情信息，包括节点名称、Host 及 SSH 用户名称、CPU 核等信息，并且支持添加节点、管理节点服务等操作。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群操作->节点管理。

添加节点

单击添加节点，输入待添加节点的 Host 信息、SSH 端口号、SSH 用户、认证方式、安装包等，单击确认。

关于认证方式的说明如下：

- SSH 密码：输入 SSH 用户对应的密码。
- SSH 密钥：单击上传，选择节点的私钥文件。需要提前在待添加节点上生成密钥文件，并将私钥发送给当前电脑（非悦数运维监控机器）。如果设置了短密码（passphrase），也需要填写。

> **Note**
>
> 节点添加后，数据不会自动负载均衡。请在信息总览页面选择图空间执行 `Balance Data` 和 `Balance Leader` 操作。

节点其他操作

在节点列表中，单击 按钮，查看对应节点的服务名、服务类型、服务状态、运行路径等信息。

- 单击节点监控可快速跳转至节点监控页面，详情信息见集群监控。
- 单击服务管理可快速跳转至服务管理页面。
- 单击编辑节点可修改节点设置。
- 当节点上无服务时，可以删除节点。删除服务请参见下文扩缩容部分。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

扩缩容

在扩缩容页面，用户可以快速添加节点、批量导入节点，并对已存在的节点添加 **Graph** 服务及 **Storage** 服务。

> **Enterpriseonly**
>
> 仅企业版悦数图数据库支持扩缩容。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群操作->扩缩容。

操作说明

添加节点

具体设置参见节点管理。

> **Note**
>
> 节点添加后，数据不会自动负载均衡。请在信息总览页面选择图空间执行 `Balance Data` 和 `Balance Leader` 操作。

批量导入节点

下载 CSV 模板并填写，然后上传 CSV 文件并选择安装包。单击确认即可批量导入节点。

修改服务

1. 在节点列表内勾选节点，然后在列表右上角单击要添加的服务，或者在列表内服务类型列单击要删除的服务标签上的**X**。
2. 在下方的服务显示区域确认修改的服务。可以单击编辑修改服务的端口。

> **Note**
>
> 绿色表示即将新增的服务，红色表示即将移除的服务。

3. 在页面下方单击确定。

> **Caution**
>
> - 目前仅支持对 graphd 和 storaged 进行扩缩容，不支持移除或添加 metad 服务。扩缩容集群时，建议提前备份数据，以便故障发生时回滚数据。关于扩缩容，详情参见如何扩缩容。
> - 集群中每种类型的服务至少要有一个，且单个节点上仅能部署一个同类型服务。
> - 移除 Storage 服务时需要迁移节点上存储的数据。请在信息总览页面执行 `Balance Data Remove` 操作。

在该示例中，新增了节点为 `192.168.8.143` 和 `192.168.8.167` 的 Storage 服务，删除了节点为 `192.168.8.169` 的 Graph 服务。虚线框并置灰服务名表示移除的服务，实线框表示添加的服务。

Copyright © Yueshu Graph Technology - All rights reserved.

重置

单击重置按钮，可以取消节点或整个扩缩容页面内所有未执行的操作，恢复到初始状态。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

服务管理

在服务管理页面，用户可以查看服务的 Host、路径、状态等信息，还可以启动、停止、强杀、重启服务，并且可以方便快捷地查看日志文件内容。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群操作->服务管理。

操作说明

Danger

单击停止/重启，会立即中断进行中的任务，可能会导致数据不一致，请在业务低峰期执行该操作。

- 找到目标服务，在操作列进行相关操作。
- 勾选多个服务，在上方进行批量操作。
- 单击 图标，可快速查看服务监控信息。
- 进行数据同步时，可以在依赖服务页面查看及管理相关的服务。关于数据同步的详情，参见集群间数据同步。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

更新配置

在更新配置页面，用户可以查看及更新服务的配置文件。

注意事项

配置更新后需要在服务页面重启对应服务。具体操作参见服务管理。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群操作->更新配置。

修改配置

1. 在上方选择需要修改配置的服务类型。
2. 查找需要修改的配置，在操作列单击编辑。
3. 在弹出的会话框中，可以单个修改配置值，也可以在上方统一修改，修改后需要单击应用到所有服务。

4. 修改完成后单击确认。

新增配置

如果需要调整配置文件中没有的参数，需要先新增配置。

1. 在左上方单击新增配置。
2. 填写参数名称，即设置项，然后填写设置值，并将设置值应用到所有服务，也可以在下方调整单个服务的取值。
3. 单击确认。

删除配置

Note

删除配置并重启服务后，对应的配置会恢复为默认值。

1. 在上方选择需要删除配置的服务类型。
2. 查找需要删除的配置，在操作列单击删除并确定。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

成员管理

成员管理页面默认只包含集群创建者账户（`owner` 角色）。`owner` 角色账户可以添加、删除集群管理员（`operator` 角色）。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群操作->成员管理。

操作说明

- 添加集群管理员：单击左上方搜索框，在下拉列表中，选择需要添加的账户，然后单击右侧的添加。

  Note

  集群管理员必需是用户创建的悦数运维监控账户。关于如何创建账户，参见权限管理。

- 删除集群管理员：在集群管理员右侧操作列单击 ，然后单击确认。
- 转移 `owner` 角色：在 `owner` 角色右侧操作列，单击转移，选择要转移的目标账户，然后单击确认。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

版本升级

悦数运维监控支持升级现有悦数图数据库集群的版本。

Caution

- 升级过程中会替换二进制文件，升级速度较快，但是仍会停机重启集群。
- 不支持自动回滚，升级失败时需要手动重新升级。
- 升级不支持暂停、取消。

Note

- 仅支持升级版本不低于 **3.0.0** 的悦数图数据库集群，且最高升级至 **3.2.1** 版本。如需升级至 **3.3.0**，请参见手动升级文档。
- 不支持跨大版本升级集群。
- 支持社区版升级为企业版或企业版升级为社区版。
- 支持升级为当前大版本内的任意小版本（包括比当前版本低的小版本）。

操作说明

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 在目标集群右侧，单击详情。
3. 在左侧导航栏，单击集群操作->版本升级。
4. 在版本升级页面确认当前悦数图数据库版本，选择待升级目标版本。

Note

如果没有找到合适的版本，单击包管理中心，下载或上传所需的版本安装包。详情参见安装包管理。

Note

升级 3.5.0 以下版本的企业版集群时，需要单击上传证书上传悦数运维监控 License。

5. 单击下一步，进行升级检查，然后再单击下一步。

   集群在升级期间将会停机，升级完成后会自动重启服务。升级前您可以通过查看诊断报告了解集群线上运行情况，从而帮助您判断当前集群停机升级是否合适。

6. 再次确认升级信息，包括集群名称、当前悦数图数据库版本和待升级悦数图数据库版本，然后单击升级。

   用户可以在在任务中心查看升级任务信息，类型为 `version update` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

使用悦数运维监控备份和恢复悦数图数据库数据

为了防止操作失误或者系统故障导致数据丢失，我们需要经常对数据进行备份，悦数图数据库提供Backup&Restore工具可帮助用户备份和恢复图数据。悦数运维监控集成了Backup&Restore能力并提供简单的 UI，用户只需几步即可进行数据备份和恢复操作。本文介绍如何使用悦数运维监控备份和恢复悦数图数据库数据。

使用限制

- 目前全量备份支持将数据备份至兼容 S3 协议的云存储中（例如阿里云对象存储 OSS、MinIO、Ceph RGW 等）以及备份至本地，增量备份仅支持将数据备份至兼容 S3 协议的云存储中。

  Note

  如需增量备份至本地，请参见备份与恢复。

- 不支持图空间级别数据的备份和恢复。
- 备份数据只能被恢复至原集群，不能跨集群恢复数据。
- 不支持断点续传备份和恢复数据。
- 目前仅支持查看备份和恢复操作产生的日志。

前提条件

- 使用悦数运维监控创建悦数图数据库集群。
- 如选择备份至兼容 S3 协议的云存储中，则需要先开通对应的云存储服务并创建存储桶。具体操作，参见对应云存储服务的文档。

操作步骤

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 在目标集群右侧详情。
3. 在左侧导航栏，单击集群操作->备份恢复。

全量备份数据

备份至兼容 S3 协议的云存储

Copyright © Yueshu Graph Technology - All rights reserved.

用户通过创建备份文件的方式将备份的数据存至云存储服务中，操作如下：

Copyright © Yueshu Graph Technology - All rights reserved.

1. 在备份恢复页面，单击备份列表。
2. 在页面右上角，单击 **S3** 配置。
3. 填写对应的云存储的配置信息，并单击提交。

配置项	说明
s3.access_key	Access Key ID，用于标识用户，例如，`AKIAI44QH8DHBxxxx`。
s3.endpoint	表示云存储服务对外服务的访问域名 URL，例如`https://s3.us-east-2.amazonaws.com`。不支持带`bucket_name`的 URL，例如`https://{bucket_name}.s3.us-west-2.amazonaws.com`。
s3.region	数据中心所在的物理位置，例如`us-east-1`。
s3.secret_key	Access Key Secret，用于加密签名字符串和用来验证签名字符串的密钥，例如`je7MtGbClwBF/2Zp9Utk/h3yCoxxxx`。
storage path	数据的存储路径，路径必须以`s3`开头，例如`s3://br-test/backup/`。

Copyright © Yueshu Graph Technology - All rights reserved.

以下列举阿里云对象存储 OSS 和 Amazon S3 的示例配置信息：

- Amazon S3 的示例配置如下：

* s3.access_key：	LTAI5tEwhrcm[illegible]
* s3.endpoint：	https://s3.us-west-2.amazonaws.com/
* s3.region：	us-west-2
* s3.secret_key：	MfjNFKNf56Y[illegible]
* storage path：	s3://br-test/

- 阿里云对象存储 OSS 的示例配置如下：

* s3.access_key：	LTAI5tK[illegible]
* s3.endpoint：	https://oss-cn-hangzhou.aliyuncs.com
* s3.region：	oss-cn-hangzhou
* s3.secret_key：	7dI9I9I[illegible]
* storage path：	s3://br-test/

Caution

备份数据至阿里云 OSS 时，需将备份路径中的 `oss` 替换为 `s3`，例如将 `oss://br-test/` 改成 `s3://br-test/`。

4. 在页面右上角，单击创建备份。
5. 在创建备份页面，备份方式选择全量。
6. 单击检查环境检查相关配置是否正常运行，然后单击提交。

Copyright © Yueshu Graph Technology - All rights reserved.

环境检查包括：

- 悦数图数据库需正常运行。
- 集群中至少有一个空间。
- S3 设置验证：登录云存储服务的密钥未过期。
- 业务流量检查：仅检查用户业务的 QPS 是否为 0，当 QPS 不为 0 时，页面提示用户在业务流量低峰期进行备份操作。

Note

如果悦数图数据库运行异常或者云存储服务的登录信息有误，用户将无法提交备份。

7. 在备份列表中查看创建的备份文件。

Note

在上一次备份操作未完成前，不能再次执行备份操作。

8. 检查存储服务中是否有上述创建的备份文件。 成功创建的备份会被存储至上述用户设置的存储路径中，例如 `s3://br-test` 。

- Amazon S3:

- 阿里云 OSS:

Danger

请勿修改生成备份文件信息，包括名称、保存路径等，否则将无法恢复数据。

备份至本地

Copyright © Yueshu Graph Technology - All rights reserved.

用户通过创建备份文件的方式将备份的数据存至本地，操作如下：

1. 在页面右上角，单击创建备份。
2. 在创建备份页面，备份方式选择全量。
3. 在备份类型中，选择 **Local**。
4. 输入本地存储路径。
5. 单击检查环境检查相关配置是否正常运行，然后单击提交。

   环境检查包括：

- 悦数图数据库需正常运行。
- 集群中至少有一个空间。
- 业务流量检查：仅检查用户业务的 QPS 是否为 0，当 QPS 不为 0 时，页面提示用户在业务流量低峰期进行备份操作。

6. 在备份列表中查看创建的备份文件。

> **Note**
>
> 在上一次备份操作未完成前，不能再次执行备份操作。

7. 检查存储服务中是否有上述创建的备份文件。成功创建的备份会被存储至上述用户设置的存储路径中。

> **Danger**
>
> 请勿修改生成备份文件信息，包括名称、保存路径等，否则将无法恢复数据。

增量备份数据

用户可以基于已存在的备份文件，进行增量备份。增量备份仅备份上一次备份后变化或者修改过的数据。

1. 在备份恢复页面，单击备份列表。
2. 在页面右上角，单击创建备份。
3. 在创建备份页面，备份方式选择增量备份。
4. 选择基础备份。

> **Note**
>
> 系统将基于这个基础备份文件进行增量备份，请确保基础备份中的数据无修改，否则可能造成增量备份失败。

5. 单击检查环境检查相关配置是否正常运行，然后单击提交。

   环境检查包括：

- 悦数图数据库需正常运行。
- 集群中至少有一个空间。
- S3 设置验证：登录云存储服务的密钥未过期。
- 业务流量检查：仅检查用户业务的 QPS 是否为 0，当 QPS 不为 0 时，页面提示用户在业务流量低峰期进行备份操作。

> **Note**
>
> 如果悦数图数据库运行异常或者云存储服务的登录信息有误，用户将无法提交备份。

Copyright © Yueshu Graph Technology - All rights reserved.

恢复数据

用户将备份数据存储至云存储服务后，可以恢复备份的数据至图数据库中。

Caution

- 恢复数据前，请确认保存至云存储服务中的备份文件的名称、存储路径等无改动，否则数据恢复会失败。
- 恢复数据的过程中，集群中的所有数据会被移除，且替换为备份文件中的数据。
- 恢复过程为离线执行，用户在恢复数据的过程中无法进行其他操作。

按照以下步骤执行恢复操作：

1. 在备份恢复页面，单击备份列表。
2. 在目标备份文件右侧，单击恢复。
3. 单击检查环境，环境检查通过后，单击提交。

环境检查包括：

- 悦数图数据库需正常运行。
- 登录云存储服务的密钥未过期。
- 无业务流量的进入。

4. 在恢复列表页面，查看恢复记录。

- 恢复记录不可被删除。
- 恢复列表中默认显示 30 天内创建的恢复记录
- 恢复记录展示记录 ID、备份名称、恢复状态、恢复时间、恢复的图空间、存储路径、操作人、及查看恢复操作产生的日志。
- 恢复状态包括 `running` 、 `success` 、 `failed` 。

Note

在上一次恢复操作未完成前，不能再次执行恢复操作。

- 用户可以通过设置筛选条件（创建时间和状态）或者搜索备份名称查看恢复记录。

查看备份/恢复进度

备份或恢复开始后，可以在对应任务的操作列单击查看日志，查看备份或恢复的进度。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.4 分析诊断

慢查询分析

在 DBA 的日常工作中，需要对集群的查询语句执行情况进行分析和治理，悦数运维监控支持查看慢查询的情况，包括语句、耗时、类型、执行计划等。

功能介绍

- 展示当前执行中的慢查询信息。可以根据关键词、图空间筛选慢查询。
- 展示慢查询历史记录。可以根据关键词、图空间、语句类型、时间范围筛选记录。

前提条件

悦数图数据库的 Graph 配置文件中，已设置 `enable_record_slow_query = true` 。详情参见 Graph 服务配置。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧的详情。
3. 在左侧导航栏，单击分析诊断->慢查询分析。

查看当前慢查询

单击当前查询标签，会展示正在执行的慢查询。参数说明如下。

参数	说明
语句	慢查询的语句。
持续时间（μs）	慢查询已经执行的时长。
开始时间	慢查询开始执行的时间。
状态	慢查询的状态。包括 `running` 和 `killing` 。
用户名	执行查询的用户名。
主机名	用户连接的服务器地址和端口。
操作	支持终止慢查询。

查看慢查询历史记录

单击历史查询标签，会展示慢查询历史记录。参数说明如下。

参数	说明
语句	慢查询的语句。
累计耗时（μs）	慢查询执行的时长。
语句类型	慢查询语句的类型，包括 `DDL` 、 `DQL` 、 `DML` 、 `DCL` 、 `UTIL` 和 `UNKNOWN` 。
图空间	执行慢查询的图空间名称。
记录时间	将语句作为慢查询记录到内存中的时间。
操作	支持查看执行计划，方便 DBA 根据执行计划优化慢查询语句。

Note

关闭慢查询记录功能不会清空慢查询历史记录。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

集群诊断

集群诊断是在指定的时间范围内，定位及分析当前集群出现的问题，并将诊断结果和集群相关的监控信息生成网页版的诊断报告。

功能介绍

- 通过诊断报告可以排查出当前集群出现的问题，并可以对问题进行解决。
- 快速了解集群内各个节点以及服务、Session、服务配置的基本情况。
- 根据诊断的报告信息，做出运维建议和集群预警。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击分析诊断->集群诊断。

创建诊断

1. 选择诊断的时间范围。支持自定义诊断时间和通过时间区间设置集群诊断的范围，包括：`1 小时`、`6 小时`、`12 小时`、`1 天`、`3 天`、`7 天`、`14 天`。

Caution

集群诊断是对集群历史数据的诊断，因此用户设置的诊断范围的结束时间不能超过当前时间。

2. 确认诊断公式的配置。用户可以自行调整公式。

公式中的 weight 函数为：`weight(value>conditionValue , weightValue)`，其中的蓝色字体内容支持修改。函数值的计算公式为 `(value - conditionValue * timeRange)/((maxValue - conditionValue) * timeRange) * weightValue`。

- value：当前指标值。
- conditionValue：满足条件的最低值。
- timeRange：时间范围天数。对于 avg 类型的指标，该参数值为 1。
- maxValue：最大指标值。百分比类型指标最大值为 100，其余类型最大值为 conditionValue 的 2 倍。

例如 `weight(hit_memory_times > 10 * days, 10)`，当时间范围选择 2 天、`hit_memory_times = 40` 时，计算公式为 `(40 - 10 * 2)/((10*2 - 10) * 2) * 10 = 10`。

3. 单击开始诊断。

4. 当诊断状态变为成功时，表示诊断已经完成，可以查看诊断报告。

查看诊断报告

在诊断列表中，单击目标诊断右侧的详情可以直接查看诊断报告。

Copyright © Yueshu Graph Technology - All rights reserved.

集群诊断报告中包含以下内容:

- 基本信息
- 展示需要重点关注的信息，例如紧急告警、普通告警、提示等。
- 展示集群健康分数、CPU 最大平均使用率、内存最大平均使用率等。
- 节点信息
- 展示各节点的 Host、实例数量、架构、系统。
- 展示各节点的 CPU、内存、硬盘、网络流量情况。
- 服务信息
- 展示各服务的在线情况、Session 信息。
- 展示 Graph、Meta、Storage 服务稳定性信息。
- 配置信息
- 展示配置变更信息。

详情页面可以下载 HTML 或 PDF 格式的诊断报告。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.5 集群信息

信息总览

用户可以在信息总览页面查看悦数图数据库内核相关信息，包括 Storage 服务信息、Storage Leader 分布、悦数图数据库各个服务的版本信息及所在节点信息、分片的分布情况及详细信息。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群信息->信息总览。

Note

在查看集群信息之前，用户需要选择任意一个在线的 Graph 服务地址，输入登录悦数图数据库的账号（非悦数运维监控登录账号）和对应密码。

Copyright © Yueshu Graph Technology - All rights reserved.

STORAGE LEADER 分布

显示 Leader 数量及 Leader 的分布。

- 单击右上角的 **Balance Leader** 按钮可以快速在悦数图数据库集群中均衡分布 Leader。关于 Leader 的详细信息，参见 Storage 服务。
- 单击右上角的详情，查看 Leader 分布的详细信息。

版本

显示悦数图数据库所有服务版本及服务地址。单击右上角的详情，查看更多信息。

Copyright © Yueshu Graph Technology - All rights reserved.

服务信息

展示 Storage 服务的基本信息。参数说明如下：

参数	说明
`Host`	主机地址
`Port`	主机端口号
`Status`	主机状态
`Git Info Sha`	版本 Commit ID
`Leader Count`	Leader 总数
`Partition Distribution`	分片分布
`Leader Distribution`	Leader 分布

单击右上角的详情，查看更多信息。

PARTITION 分布

左上方选择指定图空间：

- 查看指定图空间的分片分布情况。显示所有 Storage 服务的 IP 地址、端口，及对应 Storage 服务中的分片数量。
- 单击 **Balance Data** 均衡分布当前图空间中的所有分片。
- 单击 **Balance Data Remove** 迁移指定的 Storage 服务中的所有分片至其他 Storage 服务中，操作前系统会先引导用户选择 Storage 服务所在的节点 IP。

单击右上角的详情，查看更多信息。

分片信息

显示分片信息。用户需要在左上角选择图空间，查看分片信息。参数说明如下：

参数	说明
`Partition ID`	分片序号。
`Leader`	分片的 leader 副本的 IP 地址和端口。
`Peers`	分片所有副本的 IP 地址和端口。
`Losts`	分片的故障副本的 IP 地址和端口。

单击右上角的详情，查看更多信息，通过右上角的输入框，输入分片 ID，筛选展示的数据。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

集群作业管理

悦数运维监控可以管理指定图空间中的作业，包括查看、停止、恢复图空间内的作业，并支持查看单个作业的详情。

> **Note**
>
> 如何执行作业，请参见作业管理。

前提条件

悦数图数据库集群版本需要为企业版 3.4.0 及以上或悦数图数据库社区版 3.3.0 及以上。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群信息->作业管理。
4. 选择任意一个在线的 Graph 服务地址，输入登录悦数图数据库的账号（非悦数运维监控登录账号）和对应密码。
5. 在左上角选择目标图空间。

查看作业

选择图空间后，页面会默认展示所有未过期的作业信息。用户可以通过上方筛选框快速查找作业。方法如下：

- 选择作业状态进行筛选。状态包括 `QUEUE` 、 `RUNNING` 、 `FINISHED` 、 `FAILED` 、 `STOPPED` 、 `SUCCEEDED` 。状态说明请参见作业管理。
- 选择时间进行筛选。
- 选择 `作业命令` 或 `作业 ID` 筛选。选择后输入要搜索的内容。
- 页面的作业数据默认不自动更新，可以调整更新频率让页面自动更新，也可以单击 按钮手动更新。
- 在目标作业右侧的 `操作` 列单击 `详情` ，可以查看更多详细信息，包括 `任务 ID` 、 `Host` 、 `错误码` 等。

停止作业

在目标作业右侧的 `操作` 列单击 `停止作业` ，可以停止未完成的作业，点击后该作业状态变为 `STOPPED` 。

重新执行作业

在目标作业右侧的 `操作` 列单击 `重新执行` ，可以重新执行状态为 `FAILED` 、 `STOPPED` 的作业，点击后该作业状态变为 `RUNNING` 。

> **Note**
>
> - 如果有多个状态为 `STOPPED` 的 `BALANCE DATA` 作业，只能重新执行最新的一个。
> - 已完成的作业无法重新执行。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

审计日志

悦数图数据库的审计日志功能可以将 Graph 服务接受到的所有操作进行分类存储，悦数运维监控可以快速查看审计日志。

> **Enterpriseonly**
>
> 仅企业版悦数图数据库支持审计日志。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群信息->审计日志。

> **Note**

- 首次使用审计日志，需要根据页面提示跳转到更新配置页面开启审计日志。开启后需要重启 Graph 服务。
- 审计日志参数的说明请参见设置审计日志。

查看审计日志

在页面上方可以筛选服务或搜索日志名称，在操作列单击查看日志。

- 支持一键复制窗口内所有日志。
- 支持复制日志文件路径。
- 支持以行尾模式和区间模式查看日志。设置后需要单击刷新。
- 支持通过关键字（至少 3 个字符）搜索日志。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

运行日志

运行日志通常提供给 DBA 或开发人员查看，当系统出现故障，DBA 或开发人员可以根据运行日志定位问题。悦数运维监控可以快速查看运行日志。

入口

1. 在悦数运维监控顶部导航栏，单击集群管理。
2. 单击目标集群右侧详情。
3. 在左侧导航栏，单击集群信息->运行日志。

Note

运行日志参数的说明请参见运行日志。

查看运行日志

在页面上方可以筛选服务或搜索日志名称，在操作列单击查看日志。

- 支持一键复制窗口内所有日志。
- 支持复制日志文件路径。
- 支持行尾模式和区间模式查看日志。设置后需要单击刷新。
- 支持通过关键字（至少 3 个字符）搜索日志。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.6 通知

通知是监控悦数图数据库集群的监控指标进行报警的服务。用户可以查看告警信息，设置告警规则和告警接收方。

在悦数运维监控顶部导航栏，单击集群管理，在目标集群右侧单击详情，在左侧导航栏的通知下包含告警信息、告警规则和接收人设置。

告警信息

告警信息会以弹框的形式显示，有查看和静默按钮：

- 单击查看，会跳转至通知->告警信息页面，可以查看详细的告警信息。
- 单击静默，仅对该用户静默该告警信息，对应的告警规则在 2 小时内不会再次触发。

在告警信息页面可进行如下操作：

- 支持在搜索框中根据告警的名称搜索目标告警信息。
- 支持通过日期筛选告警信息。同时支持不同时间段的告警信息筛选，时间段包括 1 小时、6 小时、12 小时、1 天、3 天、7 天和 14 天。
- 支持根据信息的严重级别、类型、状态，筛选告警信息。用户可单击重置置空所有的筛选项。
- 用户可标记告警信息的处理状态，告警信息处理状态默认为 `unsolved`，用户可手动标记为 `处理中` 和 `已解决` 状态。

用户不可删除告警信息，但可在 `yueshu-dashboard/config/config.yaml` 文件中，通过修改 `messageStore` 参数的值来设置保存告警信息的天数。悦数运维监控默认保存 90 天内的信息。具体操作，参见部署悦数运维监控。

告警规则

对集群监控指标进行告警前，需要设置告警规则。用户可以自定义告警规则或者使用悦数运维监控内置的规则。

Copyright © Yueshu Graph Technology - All rights reserved.

创建自定义规则

用户创建自定义规则的步骤如下：

1. 在悦数运维监控顶部导航栏中，单击集群管理，然后在目标集群右侧，单击详情。
2. 在集群管理页面左侧导航栏中，单击通知->告警规则。
3. 在告警规则页面，单击自定义规则，然后单击右侧的创建规则。
4. 设置创建规则的所需的信息。

a. 在基本信息配置页中，设置告警名称、级别和频率。

参数	说明
告警名称	为告警规则设置名称。名称只能包含小写字母、数字及连字符（`-`），且必须以小写字母或数字开头及结尾；最长 253 个字符。
严重级别	为告警规则设置级别：分为 `emergency` 、 `critical` 和 `warning` 三个级别。
警报频率	设置每隔多长时间告警一次，单位分（Min）。

b. 在条件配置页中，设置指标类型、规则和告警持续时间。

参数	说明
指标类型	包括节点和服务（graphd、storaged、metad）两种指标类型。
指标规则	单击添加条件为节点或者服务设置指标规则。支持添加复合条件（`AND` 关系）。指标的详细介绍请参见监控指标。
持续时间	设置警报持续多长时间后触发告警。单位分（Min）。

c. 在规则总览页面查看整体的规则情况，确认无误。

d. 在消息设置配置页中，可看到系统自动生成的总规则及消息，然后单击提交。

> **Note**
>
> 不建议新手用户修改消息内容，否则创建规则时，可能会报错。

查看自定义规则

在自定义规则列表页，用户可进行以下操作：

- 搜索告警规则，并根据规则的严重级别、类型、指标和状态，筛选规则。
- 单击重置置空所有的筛选项。
- 打开和关闭自定义告警规则，开启状态显示 **active**，关闭状态显示 **disable**。

修改自定义规则

用户可在自定义规则列表中，选中目标规则，单击编辑图标 修改规则。

删除自定义规则

用户可在自定义规则列表中，选中目标规则，单击删除图标 删除规则。

内置规则

内置规则是悦数运维监控为用户提供的默认规则。用户可打开和关闭内置规则，开启状态显示 **active**，关闭状态显示 **disable**。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

内置规则不可被修改和删除。

静默告警规则

单击 可以静默该告警规则 2 小时。如果想中途取消静默，可以单击 。

告警接收人

用户可为目标集群设置接收告警消息的邮箱，并且查看 Webhook 的地址及 Webhook 的开启状态。有关 Webhook 的介绍，参见通知设置。

1. 在悦数运维监控顶部导航栏中，单击集群管理，然后在目标集群右侧，单击详情。
2. 在集群管理页面左侧导航栏中，单击通知->接收人设置。
3. 在接收人设置页面：

- 单击 **Mail**，填入为目标集群接收告警消息的邮箱地址，然后单击添加。
- 单击 **Webhook**，查看 Webhook 的地址及 Webhook 的开启状态。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.7 数据同步

悦数运维监控的数据同步功能用于在集群间实现数据同步。

单击悦数运维监控顶部导航栏的集群管理，在目标集群右侧单击详情，再点击左侧导航栏的数据同步即可进入。

> **Note**
>
> 关于数据同步功能的详细信息，请参见集群间数据同步。

准备工作

在服务配置中完成当前集群 Listener 的部署，并重启 Meta Listener 和 Storage Listener。

操作步骤

单击页面中的新建同步，并按照下列步骤进行操作：

1. 选择图空间：勾选主集群中需要同步的图空间，修改或使用默认图空间名称。选择完毕后单击下一步。
2. 选择从集群：在下拉框中选择数据同步从集群，单击下一步。
3. 开始同步。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.8 操作记录

本文主要介绍悦数运维监控的操作记录。

在悦数运维监控顶部导航栏，单击集群管理，单击目标集群右侧详情，然后在左侧导航栏，单击操作记录进入操作记录页面。

在操作页面可以看到 1 小时、6 小时、12 小时、1 天、3 天、7 天和 14 天的操作记录，操作人、操作、时间、集群的信息。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.5.9 其他设置

本文主要介绍悦数运维监控的其他设置。

在悦数运维监控顶部导航栏，单击集群管理，单击目标集群右侧详情，然后在左侧导航栏，单击其他设置进入其他设置页面。

- 集群信息：显示集群名称、创建时间、创建用户信息、集群所有者名称。
- 解除绑定：解除绑定后从平台移除该集群的信息，对集群的服务和悦数图数据库数据不做任何操作。

  Note

  在执行该操作时，用户需要输入集群名称确定解除绑定。

- 删除集群：删除集群会停止集群服务同时保留悦数图数据库数据，并解除与平台的信息绑定，请谨慎操作。

  Note

  在执行该操作时，用户需要输入集群名称确定删除集群。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.6 权限管理

用户可以使用不同账户登录悦数运维监控，不同账户拥有不同权限。本文介绍账户类型、账户角色及角色权限等。

Note

使用 LDAP 账户 或 OAuth2.0 账户前，需要配置相关的协议。详情参见单点登录。

16.6.1 账户类型

悦数运维监控的初始化账户名为 `nebula` 和密码为 `nebula` 。使用初始化账户名和密码登录悦数运维监控后，即可创建不同类型的账户：LDAP 账户、OAuth2.0 账户和普通账户。

LDAP 账户

悦数运维监控通过对接 LDAP（Lightweight Directory Access Protocol），使得用户可以使用企业内部账户体系登录悦数运维监控。

OAuth2.0 账户

Caution

该功能仍处于测试中，后续会继续进行调整优化。

悦数运维监控基于 OAuth2.0 协议，使用 access_token 来授权第三方应用访问受保护的信息。

普通账户

悦数运维监控支持用户自行创建本地账号。

16.6.2 账户角色

用户可以为账户设置不同的角色，不同的角色拥有的权限不同。悦数运维监控中的账户角色有两种类型：系统角色（`admin` 和 `user`）和集群角色（`owner` 和 `operator`）。

系统角色和集群角色的关系及介绍如下。

Copyright © Yueshu Graph Technology - All rights reserved.

系统角色：

角色名	权限	说明
admin	1. 创建账户。 2. 修改已有账户的角色。 3. 进行平台设置、系统级别告警设置。 4. 删除账户。	1. 系统中可以有多个 `admin` 角色，即系统管理员角色。 2. 默认 `admin` 是所有集群的 `operator` ，即可以看到所有集群。 3. `admin` 可以指定 `user` 为集群的 `operator` 。 4. 默认显示在集群成员列表中。 `owner` 无法将 `admin` 成员移除，除非将 `admin` 成员转换成 `user` ，系统将从集群中自动移除 `admin` 成员。
user	1. 系统维度只有只读权限。 2. `admin` 新建一个 `user` 角色的账户后，若没有分配对应的集群，则该账户无法查看任何集群。 3. 可以创建集群，成为该集群的 `owner` 。	1. 普通用户角色。 2. 系统中可以有多个 `user` 。

集群角色：

角色名	权限	说明
`operator`	1. 扩缩容集群。 2. 设置集群告警。 3. 进行节点管理。 4. 进行服务管理。	1. 集群操作者角色。 2. 一个集群可以有多个 `operator` 。
`owner`	1. 拥有 `operator` 的所有权限。 2. 解绑和删除集群。 3. 添加和移除 `operator` 账户。 4. 转移 `owner` 角色。	1. 集群所有者角色。 2. 一个集群只能有一个 `owner` 。

Copyright © Yueshu Graph Technology - All rights reserved.

16.6.3 创建账户

角色为 `admin` 的账户可以创建账户。步骤如下：

1. 在悦数运维监控页面的顶部，单击权限管理，然后单击创建。
2. 选择创建方式并填入信息，然后单击确定。

- 邀请（LDAP 或 OAuth2.0 账户）：设置被邀请人的账号类型、企业邮箱和角色。被邀请人单击邮件中的 **Accept** 按钮激活账户后，然后单击登录自动跳转至悦数运维监控登录页面。被邀请人使用企业邮箱账号和邮箱密码即可登录悦数运维监控。

> **Note**
>
> 开启 LDAP 后还支持自动注册。在登录页面选择 LDAP 方式输入未注册账号登录时，悦数运维监控会自动注册该账号，但是角色权限为 `user` 。

- 创建账户（普通账户）：设置新账户的登录名称、密码、角色。关于角色的详情，参见上文。

16.6.4 查看账户

已创建的账户会显示在权限管理页面中。

- 用户可以查看账户的用户名、账号类型、角色、关联集群、创建时间。
- 账号类型：包括 **ldap**、 **oauth2.0** 和 **platform**。**platform** 表示普通账户。
- 角色：显示角色名，包括 **admin** 和 **user**。关于角色的详情，参见上文。
- 关联集群：显示该账户可以操作的所有集群。如果集群为该账户所创建，相关集群附带 `owner` 标签。

- 用户可以在搜索框中搜索账户，也可以通过选择关联集群来筛选账户。

16.6.5 其他操作

- 在权限管理页面的操作列下，单击 编辑账户信息。
- 在权限管理页面的操作列下，单击 删除账户。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.7 任务中心

悦数运维监控支持展示正在运行的任务的进度和已结束的历史任务的信息。

16.7.1 注意事项

- 不支持取消进行中的任务。
- 不支持删除历史任务。

16.7.2 任务类型

目前任务中心支持的任务类型包括:

- **install**：创建集群。
- **scale**：扩缩容。
- **version update**：版本升级。
- **package upload**：上传安装包。
- **package download**：下载安装包。
- **package deploy**：添加节点时部署安装包。

16.7.3 入口

在悦数运维监控顶部导航栏中，单击任务中心可查看相关任务信息。

16.7.4 运行中的任务

单击运行中任务页签，用户可以查看运行中的任务进度。

- 单击任务名称可以查看正在进行中的任务的 ID、节点名称、类型、创建时间、操作人。
- 单击任务信息可以查看任务详情。

16.7.5 历史任务

单击历史任务页签，用户可以查看所有结束的任务。

- 用户可通过任务状态、类型、时间段筛选历史任务。
- 在目标历史任务右侧，单击任务信息查看历史任务的详情；单击任务日志查看任务执行日志。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.8 许可证管理

悦数运维监控目前支持可视化一键部署许可证管理工具 (License Manager，LM)。在悦数图数据库及附属软件中配置 LM 后，可通过 LM 校验许可证的有效性，从而保证图数据库及附属软件的正常使用。

悦数运维监控的许可证管理页面展示许可证信息、许可证使用情况和产品列表。通过页面上方的按钮，还可以编辑、更换许可证。

16.8.1 使用说明

在许可证管理页面查看 License 相关信息前，请确保已经激活悦数运维监控。详情请参见激活悦数运维监控。

16.8.2 入口

在悦数运维监控顶部导航栏中，单击许可证管理。

16.8.3 许可证信息

在许可证管理页面，可以查看：

- LM 的运行状态。当 LM 正常运行时，页面上方显示运行中；当 LM 未运行时，页面上方显示已退出。
- 许可证的相关信息，包括许可证的状态、许可证的有效期、LM 的 ID 和 LM 的访问链接。
- 许可证的使用情况，包括配额类型、配额总数和剩余配额。
- 使用配额的产品列表，包括产品名称、产品状态、产品使用的配额类型和配额数量。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.9 平台设置

16.9.1 系统设置

系统设置页面可以修改页面标题、Logo 图片和封面图，以及切换语言和开关帮助提示。

入口

1. 在悦数运维监控顶部导航栏，单击平台设置。
2. 在左侧导航栏单击系统设置。

操作说明

- 修改页面标题、Logo 图片和封面图后单击保存。
- 更换显示语言，目前仅支持中文和英文。
- 开启和关闭帮助提示，提示示例如下：

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.9.2 通知设置

通知设置页面可以设置通知邮箱和 Webhook。

入口

1. 在悦数运维监控顶部导航栏，单击平台设置。
2. 在左侧导航栏单击通知设置。

操作说明

邮箱

悦数运维监控支持通过邮箱发送和接收所有集群的告警消息。

- 用户需设置以下参数来发送告警消息。

参数	说明
SMTP 服务器地址	用户邮箱对应的 SMTP 服务器地址。
端口号	邮箱对应的 SMTP 服务器的端口号。
Use SSL	勾选后启用 SSL 进行数据的加密传输。
SMTP 用户名	SMTP 服务器的用户名。
SMTP 密码	SMTP 服务器的密码。
发件人邮箱	发送告警消息的邮箱地址。

- 用户需设置接收人来接收告警消息。

参数	说明
接收人	接收告警消息的邮箱地址。该邮箱将接收悦数运维监控上所有集群的告警消息。

WEBHOOK

悦数运维监控支持用户配置 Webhook 将所有集群的告警消息接入第三方项目中。

在平台设置页面的左侧导航栏中，单击通知设置**->Webhook**，填入接收告警消息的 **Webhook URL** 和**Webhook**请求体**body**。用户也可在页面右上方开启和关闭 Webhook 功能。

Copyright © Yueshu Graph Technology - All rights reserved.

支持的变量如下。

变量	说明
`${cluster.name}`	集群名称。
`${cluster.id}`	集群 ID。
`${cluster.version}`	集群版本。
`${cluster.status}`	集群当前状态。
`${cluster.owner}`	拥有集群 Owner 角色权限的账号名称。
`${alert.labels.instance}`	服务的 IP 地址和端口。
`${alert.labels.instanceName}`	服务名称。例如 `192.168.10.100-graphd-9669` 。
`${alert.labels.alertname}`	告警名称。
`${alert.labels.severity}`	告警严重级别。
`${alert.labels.componentType}`	告警服务类型。
`${alert.annotations.summary}`	告警摘要。
`${alert.annotations.description}`	告警消息描述。

关于告警的更多信息，请参见通知。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.9.3 单点登录

悦数运维监控支持普通账户、LDAP 账户和 OAuth2.0 账户。本文介绍如何在悦数运维监控上配置 LDAP 和 OAuth2.0 协议。

Note

- 配置完成后，请参见权限管理创建账户和邀请激活。
- 可以在左侧导航栏里通过开关快速开启或关闭 LDAP 或 OAuth2.0 登录方式。

LDAP 设置

入口

1. 在悦数运维监控顶部导航栏，单击平台设置。
2. 在左侧导航栏单击单点登录**->LDAP**。

设置说明

参数	示例	说明
`LDAP Server Address`	`ldap://192.168.10.100`	LDAP 服务器地址。
`Bind DN`	`cn=admin,dc=vesoft,dc=com`	LDAP 服务器登录用户名。
`Password`	`123456`	LDAP 服务器登录密码。
`Base DN`	`dc=vesoft,dc=com`	查询用户数据的路径。
`User Filter`	`&(objectClass=*)`	查询条件。
`Email Key`	`mail`	LDAP 数据库存放邮箱信息的字段名。

使用说明

开启 LDAP 后有 2 种 LDAP 账号的注册方法：

- 邮箱邀请：在权限管理页面创建账号时可以通过邮箱邀请别人进行注册，优点是可以设置账号的角色权限。
- 自动注册：在登录页面选择 LDAP 方式输入未注册账号登录时，悦数运维监控会自动注册该账号，但是角色权限为 `user` 。

OAuth2.0 设置

Caution

该功能仍处于测试中，后续会继续进行调整优化。

入口

1. 在悦数运维监控顶部导航栏，单击平台设置。
2. 在左侧导航栏单击单点登录**->OAuth2.0**。

Copyright © Yueshu Graph Technology - All rights reserved.

设置说明

参数	示例	说明
`ClientID`	`4953xxx-mmnoge13xx.apps.googleusercontent.com`	应用的 ClientId。
`ClientSecret`	`GOCxxx-xaytomFexxx`	应用的 ClientSecret。
`RedirectURL`	`http://dashboard.yueshu-inc.com/login`	重定向到悦数运维监控的 URL。
`AuthURL`	`https://accounts.google.com/o/oauth2/auth`	认证 URL。
`TokenURL`	`https://oauth2.googleapis.com/token`	获取 access_token 的URL。
`UserInfoURL`	`https://www.googleapis.com/oauth2/v1/userinfo`	获取用户信息的 URL。
`Username Key`	`email`	用户名字段。
`Organization`	`yueshu company`	组织名称。
`OAuth权限范围`	`email`	OAuth 的权限范围。权限范围需要是厂商 OAuth2.0 平台配置的 scope 的子集，否则请求会失败。请求的 scope 需要能获取到 `Username Key` 的值。

使用说明

开启 OAuth2.0 后，在权限管理页面创建账号时可以通过邮箱邀请别人进行注册。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.9.4 安装包管理

悦数运维监控支持管理悦数图数据库安装包，可以下载社区版的各个版本，或者手动上传安装包。

注意事项

只有 `admin` 角色的用户可以管理安装包。

入口

1. 在悦数运维监控顶部导航栏，单击平台设置。
2. 在左侧导航栏单击安装包管理。

操作说明

查看安装包

界面右侧会显示已有安装包的列表，展示安装包名称、版本、大小和创建时间。

用户可以通过右上角的搜索框进行筛选。

下载安装包

1. 单击下载安装包，选择要下载的安装包。说明如下：

> **Note**
>
> 下载已存在的安装包，会提示覆盖原有安装包。

- 版本：支持 v2.5 以上的稳定版本。
- 平台：支持 CentOS 7/8、Ubuntu 1604/1804/2004。
- 安装包类型：支持 RPM、DEB、tar.gz。

2. 单击下载。

下载任务可以在任务中心查看任务信息，类型为 `package download` 。如果任务状态为 `success` ，返回安装包管理页面，即可查看到新下载的安装包。

上传安装包

如果在下载安装包的列表里没有需要的安装包，用户可以手动上传安装包，例如上传企业版安装包。

单击上传安装包，选择本地要上传的安装包。类型可以是 RPM、DEB 或 tar.gz。查看界面上方的上传进度，等待上传完成。

上传任务可以在任务中心查看任务信息，类型为 `package upload` 。如果任务状态为 `success` ，返回安装包管理页面，即可查看到新上传的安装包。

删除安装包

找到需要删除的安装包，在目标安装包的操作列，单击删除并确认。

常见问题

上传安装包时报错 `REQUEST ENTITY TOO LARGE` 如何处理?

如果用户使用 Nginx 做反向代理，默认限制上传文件大小为 1 MB，请在 `nginx.conf` 中的 `http{}` 段中添加 `client_max_body_size 200m;` ，即允许上传最大 200 MB 的文件。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.10 监控指标说明

本文介绍悦数运维监控中展示的悦数图数据库监控指标。

16.10.1 机器

Note

- 以下机器指标适用于 Linux 操作系统。
- 磁盘容量和流量的默认单位为字节（Byte），页面显示时单位会随着数据量级而变化，例如流量低于 1 KB/s 时单位为 Bytes/s。
- 对于所有版本的悦数运维监控，机器的 Buff 和 Cache 所占的内存没有被计算在内存使用率中。

CPU

参数	说明
`cpu_utilization`	CPU 已使用百分比。
`cpu_idle`	CPU 空闲百分比。
`cpu_wait`	等待 IO 操作的 CPU 百分比。
`cpu_user`	用户空间（非悦数图数据库图空间）占用的 CPU 百分比。
`cpu_system`	内核空间（非悦数图数据库内核空间）占用的 CPU 百分比。

内存

参数	说明
`memory_utilization`	内存已使用百分比。
`memory_used`	已使用内存（不包括缓存）。
`memory_free`	空闲内存。

负载

参数	说明
`load_1m`	最近 1 分钟系统平均负载。
`load_5m`	最近 5 分钟系统平均负载。
`load_15m`	最近 15 分钟系统平均负载。

Copyright © Yueshu Graph Technology - All rights reserved.

磁盘

参数	说明
`disk_used_percentage`	磁盘使用率。
`disk_used`	磁盘已使用存储空间。
`disk_free`	磁盘剩余存储空间。
`disk_readbytes`	磁盘每秒读取的字节数。
`disk_writebytes`	磁盘每秒写入的字节数。
`disk_readiops`	磁盘每秒的读请求数量。
`disk_writeiops`	磁盘每秒的写请求数量。
`inode_utilization`	inode 已使用百分比。

流量

参数	说明
`network_in_rate`	网卡每秒接收的字节数。
`network_out_rate`	网卡每秒发送的字节数。
`network_in_errs`	网卡每秒接收错误的字节数。
`network_out_errs`	网卡每秒发送错误的字节数。
`network_in_packets`	网卡每秒接收的数据包数量。
`network_out_packets`	网卡每秒发送的数据包数量。

16.10.2 服务

周期

指标统计的时间范围，当前支持 5 秒、60 秒、600 秒和 3600 秒，分别表示最近 5 秒、最近 1 分钟、最近 10 分钟和最近 1 小时。

聚合方式

参数	说明
`rate`	周期内平均每秒操作的速率。
`sum`	周期内操作的总和。
`avg`	周期内响应平均耗时。
`P75`	周期内响应耗时从小到大排列，顺序处于 75% 位置的分位数值。
`P95`	周期内响应耗时从小到大排列，顺序处于 95% 位置的分位数值。
`P99`	周期内响应耗时从小到大排列，顺序处于 99% 位置的分位数值。
`P999`	周期内响应耗时从小到大排列，顺序处于 99.9% 位置的分位数值。

Note

以下为悦数运维监控获取内核的全量指标，但悦数运维监控仅展示重要的指标。

Copyright © Yueshu Graph Technology - All rights reserved.

Graph

参数	说明
`num_active_queries`	活跃的查询语句数的变化数。 计算公式：时间范围内开始执行的语句数减去执行完毕的语句数。
`num_active_sessions`	活跃的会话数的变化数。 计算公式：时间范围内登录的会话数减去登出的会话数。 例如查询num_active_sessions.sum.5，过去 5 秒中登录了 10 个会话数，登出了 30 个会话数，那么该指标值就是 `-20` （10-30）。
`num_aggregate_executors`	聚合（Aggregate）算子执行时间。
`num_auth_failed_sessions_bad_username_password`	因用户名密码错误导验证失败的会话数量。
`num_auth_failed_sessions_out_of_max_allowed`	因为超过 `FLAG_OUT_OF_MAX_ALLOWED_CONNECTIONS` 参数导致的验证登录的失败的会话数量。
`num_auth_failed_sessions`	登录验证失败的会话数量。
`num_indexscan_executors`	索引扫描（IndexScan）算子执行时间。
`num_killed_queries`	被终止的查询数量。
`num_opened_sessions`	服务端建立过的会话数量。
`num_queries`	查询次数。
`num_query_errors_leader_changes`	因查询错误而导致的 Leader 变更的次数。
`num_query_errors`	查询错误次数。
`num_reclaimed_expired_sessions`	服务端主动回收的过期的会话数量。
`num_rpc_sent_to_metad_failed`	Graphd 服务发给 Metad 的 RPC 请求失败的数量。
`num_rpc_sent_to_metad`	Graphd 服务发给 Metad 服务的 RPC 请求数量。
`num_rpc_sent_to_storaged_failed`	Graphd 服务发给 Storaged 服务的 RPC 请求失败的数量。
`num_rpc_sent_to_storaged`	Graphd 服务发给 Storaged 服务的 RPC 请求数量。
`num_sentences`	Graphd 服务接收的语句数。
`num_slow_queries`	慢查询次数。
`num_sort_executors`	排序（Sort）算子执行时间。
`optimizer_latency_us`	优化器阶段延迟时间。
`query_latency_us`	查询延迟时间。
`slow_query_latency_us`	慢查询延迟时间。
`num_queries_hit_memory_watermark`	达到内存水位线的语句的数量。

Copyright © Yueshu Graph Technology - All rights reserved.

Meta

参数	说明
`commit_log_latency_us`	Raft 协议中 Commit 日志的延迟时间。
`commit_snapshot_latency_us`	Raft 协议中 Commit 快照的延迟时间。
`heartbeat_latency_us`	心跳延迟时间。
`num_heartbeats`	心跳次数。
`num_raft_votes`	Raft 协议中投票的次数。
`transfer_leader_latency_us`	Raft 协议中转移 Leader 的延迟时间。
`num_agent_heartbeats`	AgentHBProcessor 心跳次数。
`agent_heartbeat_latency_us`	AgentHBProcessor 延迟时间。
`replicate_log_latency_us`	Raft 复制日志至大多数节点的延迟。
`num_send_snapshot`	Raft 发送快照至其他节点的次数。
`append_log_latency_us`	Raft 复制日志到单个节点的延迟。
`append_wal_latency_us`	Raft 写入单条 WAL 的延迟。
`num_grant_votes`	Raft 投票给其他节点的次数。
`num_start_elect`	Raft 发起投票的次数。

Copyright © Yueshu Graph Technology - All rights reserved.

Storage

Copyright © Yueshu Graph Technology - All rights reserved.

参数	说明
`add_edges_latency_us`	添加边的延迟时间。
`add_vertices_latency_us`	添加点的延迟时间。
`commit_log_latency_us`	Raft 协议中 Commit 日志的延迟时间。
`commit_snapshot_latency_us`	Raft 协议中 Commit 快照的延迟时间。
`delete_edges_latency_us`	删除边的延迟时间。
`delete_vertices_latency_us`	删除点的延迟时间。
`get_neighbors_latency_us`	查询邻居延迟时间。
`get_dst_by_src_latency_us`	通过起始点获取终点的延迟时间。
`num_get_prop`	GetPropProcessor 执行的次数。
`num_get_neighbors_errors`	GetNeighborsProcessor 执行出错的次数。
`num_get_dst_by_src_errors`	GetDstBySrcProcessor 执行出错的次数。
`get_prop_latency_us`	GetPropProcessor 执行的延迟时间。
`num_edges_deleted`	删除的边数量。
`num_edges_inserted`	插入的边数量。
`num_raft_votes`	Raft 协议中投票的次数。
`num_rpc_sent_to_metad_failed`	Storaged 服务发给 Metad 服务的 RPC 请求失败的数量。
`num_rpc_sent_to_metad`	Storaged 服务发给 Metad 服务的 RPC 请求数量。
`num_tags_deleted`	删除的 Tag 数量。
`num_vertices_deleted`	删除的点数量。
`num_vertices_inserted`	插入的点数量。
`transfer_leader_latency_us`	Raft 协议中转移 Leader 的延迟时间。
`lookup_latency_us`	LookupProcessor 执行的延迟时间。
`num_lookup_errors`	LookupProcessor 执行时出错的次数。
`num_scan_vertex`	ScanVertexProcessor 执行的次数。
`num_scan_vertex_errors`	ScanVertexProcessor 执行时出错的次数。
`update_edge_latency_us`	UpdateEdgeProcessor 执行的延迟时间。
`num_update_vertex`	UpdateVertexProcessor 执行的次数。
`num_update_vertex_errors`	UpdateVertexProcessor 执行时出错的次数。
`kv_get_latency_us`	Getprocessor 的延迟时间。
`kv_put_latency_us`	PutProcessor 的延迟时间。
`kv_remove_latency_us`	RemoveProcessor 的延迟时间。
`num_kv_get_errors`	GetProcessor 执行出错次数。
`num_kv_get`	GetProcessor 执行次数。
`num_kv_put_errors`	PutProcessor 执行出错次数。
`num_kv_put`	PutProcessor 执行次数。

Copyright © Yueshu Graph Technology - All rights reserved.

参数	说明
`num_kv_remove_errors`	RemoveProcessor 执行出错次数。
`num_kv_remove`	RemoveProcessor 执行次数。
`forward_tranx_latency_us`	传输延迟时间。
`scan_edge_latency_us`	ScanEdgeProcessor 执行的延迟时间。
`num_scan_edge_errors`	ScanEdgeProcessor 执行时出错的次数。
`num_scan_edge`	ScanEdgeProcessor 执行的次数。
`scan_vertex_latency_us`	ScanVertexProcessor 执行的延迟时间。
`num_add_edges`	添加边的次数。
`num_add_edges_errors`	添加边时出错的次数。
`num_add_vertices`	添加点的次数。
`num_start_elect`	Raft 发起投票的次数
`num_add_vertices_errors`	添加点时出错的次数。
`num_delete_vertices_errors`	删除点时出错的次数。
`append_log_latency_us`	Raft 复制日志到单个节点的延迟。
`num_grant_votes`	Raft 投票给其他节点的次数。
`replicate_log_latency_us`	Raft 复制日志到大多数节点的延迟。
`num_delete_tags`	删除 Tag 的次数。
`num_delete_tags_errors`	删除 Tag 时出错的次数。
`num_delete_edges`	删除边的次数。
`num_delete_edges_errors`	删除边时出错的次数。
`num_send_snapshot`	发送快照的次数。
`update_vertex_latency_us`	UpdateVertexProcessor 执行的延迟时间。
`append_wal_latency_us`	Raft 写入单条 WAL 的延迟。
`num_update_edge`	UpdateEdgeProcessor 执行的次数。
`delete_tags_latency_us`	删除 Tag 的延迟时间。
`num_update_edge_errors`	UpdateEdgeProcessor 执行时出错的次数。
`num_get_neighbors`	GetNeighborsProcessor 执行的次数。
`num_get_dst_by_src`	GetDstBySrcProcessor 执行的次数。
`num_get_prop_errors`	GetPropProcessor 执行时出错的次数。
`num_delete_vertices`	删除点的次数。
`num_lookup`	LookupProcessor 执行的次数。
`num_sync_data`	Storage 同步 Drainer 数据的次数。
`num_sync_data_errors`	Storage 同步 Drainer 数据出错的次数。
`sync_data_latency_us`	Storage 同步 Drainer 数据的延迟时间。

Copyright © Yueshu Graph Technology - All rights reserved.

图空间级别监控指标

Note

图空间级别监控指标是动态创建的，只有当图空间内触发该行为时，对应的指标才会创建，用户才能查询到。

参数	说明
`num_active_queries`	当前正在执行的查询数。
`num_queries`	查询次数。
`num_sentences`	Graphd 服务接收的语句数。
`optimizer_latency_us`	优化器阶段延迟时间。
`query_latency_us`	查询延迟时间。
`num_slow_queries`	慢查询次数。
`num_query_errors`	查询报错语句数量。
`num_query_errors_leader_changes`	因查询错误而导致的 Leader 变更的次数。
`num_killed_queries`	被终止的查询数量。
`num_aggregate_executors`	聚合（Aggregate）算子执行时间。
`num_sort_executors`	排序（Sort）算子执行时间。
`num_indexscan_executors`	索引扫描（IndexScan）算子执行时间。
`num_auth_failed_sessions_bad_username_password`	因用户名密码错误导验证失败的会话数量。
`num_auth_failed_sessions`	登录验证失败的会话数量。
`num_opened_sessions`	服务端建立过的会话数量。
`num_queries_hit_memory_watermark`	达到内存水位线的语句的数量。
`num_reclaimed_expired_sessions`	服务端主动回收的过期的会话数量。
`num_rpc_sent_to_metad_failed`	Graphd 服务发给 Metad 的 RPC 请求失败的数量。
`num_rpc_sent_to_metad`	Graphd 服务发给 Metad 服务的 RPC 请求数量。
`num_rpc_sent_to_storaged_failed`	Graphd 服务发给 Storaged 服务的 RPC 请求失败的数量。
`num_rpc_sent_to_storaged`	Graphd 服务发给 Storaged 服务的 RPC 请求数量。
`slow_query_latency_us`	慢查询延迟时间。

Copyright © Yueshu Graph Technology - All rights reserved.

单进程指标

Graph、Meta 和 Storage 服务都有自身的单进程指标。

参数	说明
`context_switches_total`	上下文切换数量。
`cpu_seconds_total`	基于用户时间和系统时间的 CPU 使用量。
`memory_bytes_gauge`	使用的内存字节数。
`open_filedesc_gauge`	文件描述符的数量。
`read_bytes_total`	读取的字节数量。
`write_bytes_total`	写入的字节数量。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

16.11 常见问题 FAQ

本文列出了使用悦数运维监控时可能遇到的常见问题，用户可以使用文档中心或者浏览器的搜索功能查找相应问题。

16.11.1 什么是集群、节点和服务?

- 集群：悦数运维监控集群特指由多个悦数图数据库服务所在的节点组成的一组系统。
- 节点：特指承载运行悦数图数据库服务的物理或虚拟机。
- 服务：特指悦数图数据库服务，包括 Metad，Storaged 和 Graphd。

16.11.2 什么是集群的状态?

集群的状态有以下几种：

- installing：集群正在创建中，过程大约需要等待 3～10 分钟。
- healthy：集群中所有服务正常运行。
- unhealthy：集群服务中存在非正常运行的服务。

16.11.3 为什么要授权节点?

由于集群的管理操作需要获取对应节点的 SSH 信息，因此悦数运维监控在执行操作前，需要先拥有最少可执行权限的 SSH 账号密码信息。

16.11.4 什么是扩缩容?

悦数图数据库是分布式图数据库，可以支持运行时动态扩展和缩减服务。也可以通过悦数运维监控可视化的动态扩展或缩减 Storage 和 Graph 服务（不可以扩缩容 Metad）。

16.11.5 为什么不能操作 Meta 服务?

Meta 服务主要存储悦数图数据库数据库中的元数据。一旦 Meta 服务出现问题，整个集群会面临瘫痪风险。且 Meta 处理的数据量较少，一般不会出现扩缩容场景，因此我们直接在悦数运维监控中禁用了操作 Meta，防止出现用户误操作导致集群不可用的情况。

16.11.6 扩缩容之后对数据有什么影响?

- 扩容 Storage：悦数运维监控会在指定的机器上创建并运行 Storage 服务，对已有数据不会造成影响，可以在左侧导航栏集群信息->信息总览中的 `Storage Leader 分布` 和 `Partition` 分布区域，根据自身需求选择进行 `Balance Leader` 或者 `Balance Data` 操作。
- 缩容 Storage：如果 Storage 上有分片的话，无法直接缩容。用户需要先在左侧导航栏集群信息->信息总览中的 `Partition 分布` 区域单击 `Balance Data Remove`，将指定的 Storage 服务中的所有分片迁移至其他 Storage 服务中。
- 扩缩容 Graph 对数据不会有影响。

16.11.7 为什么无法启动悦数运维监控?

- 确保已加载 License Key。
- 确保 License 未过期。

用户也可以在悦数运维监控目录中执行 `cat logs/webserver.log` 查看各个模块的启动信息。如果满足以上条件，仍无法启动悦数运维监控，请前往 悦数图数据库官方论坛咨询。

16.11.8 为什么导入集群时，提示 “ssh 连接失败”?

如果在为节点授权时，节点**Host** 显示为 `127.0.0.1` 且悦数运维监控和悦数图数据库没有被部署在同一台机器上，系统会提示 “ssh 连接失败”。用户需要在悦数图数据库所有服务的配置文件中，修改各服务的 Host IP 为真实的机器 IP。更多信息，参见配置管理。

Copyright © Yueshu Graph Technology - All rights reserved.

如果用户导入的是 Docker 部署的集群，也会显示 “ssh 连接失败”。悦数运维监控不支持导入使用 Docker 部署的集群。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17. 图探索

17.1 什么是悦数图探索

悦数图探索是一款可以通过 Web 访问的可视化图探索工具，搭配悦数图数据库内核使用，用于与图数据进行可视化交互。即使没有图数据操作经验，用户也可以快速成为图专家。

17.1.1 适用场景

如果有以下任一需求，都可以使用悦数图探索：

- 从复杂关系中快速查找友邻关系、分析可疑目标，需要以可视化的方式展示图数据。
- 针对大规模数据集，需要以可视化的方式对数据进行过滤、分析和图探索。

17.1.2 产品优点

- 易于安装：简单步骤即可完成部署。
- 易于使用：可视化构造查询场景，无需构思 nGQL 语句，轻松实现图探索。
- 灵活性强：支持通过 VID、Tag、Subgraph 等方式查询数据。
- 探索拓展：支持对多个点进行拓展操作、查询多个点的共同邻居、查询起点到终点之间的路径等操作。
- 多样展示：支持修改画布中点的颜色和图标，突出关键节点；也支持使用不同布局模式展示数据。
- 数据存储：支持保存和导入画布数据。
- 内联框架：支持在第三方页面中嵌入悦数图探索画布。

17.1.3 身份验证

悦数图数据库默认不启用身份验证，一般情况下用户可以使用 `root` 账号和任意密码登录悦数图探索。

悦数图数据库启用了身份验证后，用户只能使用指定的账号和密码登录悦数图探索。

关于悦数图数据库的身份验证功能，参考 悦数图数据库用户手册。

17.1.4 版本兼容性

悦数图数据库的版本和悦数图探索版本对应关系如下。

悦数图数据库版本	悦数图探索版本
3.5.x	3.5.1、3.5.0、3.4.0
3.4.0 ~ 3.4.1	3.5.1、3.5.0、3.4.0、3.2.1、3.2.0
3.3.0	3.2.1、3.2.0
3.1.0 ~ 3.2.x	3.1.0
3.0.0 ~ 3.1.0	3.0.0
2.5.x ~ 3.0.0	2.2.0
2.6.x	2.1.0
2.5.x	2.0.0

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.2 安装与登录

17.2.1 部署悦数图探索

本文介绍如何在本地通过 RPM、DEB 和 TAR 包部署悦数图探索。

前提条件

在部署悦数图探索之前，用户需要确认以下信息：

- 已在 LM 中加载 License Key。
- 悦数图数据库服务已经部署并启动。详细信息参考 悦数图数据库安装部署。
- 以下端口未被使用。

端口号	说明
7002	悦数图探索提供的 web 服务

Caution

悦数图探索默认使用的端口号为 7002，用户可以在安装目录下的 `conf/app.conf` 文件中修改 `httpport`，并重启服务。

- 使用的 Linux 发行版为 CentOS。

注意事项

悦数图探索从 3.2.0 版本开始内置了 Dag Controller 安装包，用于提供图计算服务。用户可以自行决定是否启动 Dag Controller 服务。如果没有启动 Dag Controller 服务，悦数图探索中的 **Workflow** 菜单将显示为灰色无法点击。

此外，如果需要使用 **Workflow** 进行复杂图计算，还需在安装悦数图探索后配置 NFS 或 HDFS，namenode 默认使用 8020 端口，datanode 默认使用 50010 端口。详情参见 **Workflow** 的资源配置。

RPM 部署

安装

1. 根据需要下载 RPM 包，建议选择最新版本。
2. 使用 `sudo rpm -i <rpm>` 命令安装 RPM 包。

   例如，安装悦数图探索需要运行以下命令，默认安装路径为 `/usr/local/yueshu-explorer`：

```
sudo rpm -i yueshu-explorer-<version>.x86_64.rpm
```

   也可以使用 `--prefix` 选项安装到指定路径：

```
sudo rpm -i yueshu-explorer-<version>.x86_64.rpm --prefix=<path>
```

3. 进入解压后的文件夹，在 `config` 目录内修改 `app-config.yaml` 文件，设置 `LicenseManagerURL` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。
4. （可选）配置 Dag Controller。参见下文 配置 **Dag Controller** 部分。
5. 进入 `yueshu-explorer` 文件夹，执行以下命令启动服务。

```
cd yueshu-explorer

# 启动悦数图探索。
sudo ./scripts/start.sh
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
# （可选）启动 Dag Controller。
sudo ./dag-ctrl/scripts/start.sh
```

启停服务

支持使用 systemctl 服务控制项目启停。

```
systemctl status yueshu-explorer #查看服务状态
systemctl stop yueshu-explorer #停止服务
systemctl start yueshu-explorer #启动服务
```

也可以在安装目录下使用以下命令，手动启动或停止服务。

```
sudo ./scripts/start.sh #启动悦数图探索服务
sudo ./scripts/stop.sh #停止悦数图探索服务
sudo ./dag-ctrl/scripts/start.sh #启动 Dag Controller 服务
sudo ./dag-ctrl/scripts/stop.sh #停止 Dag Controller 服务
```

卸载

使用以下的命令卸载悦数图探索。

```
sudo rpm -e yueshu-explorer-<version>.x86_64
```

DEB 部署

安装

1. 下载 DEB 包。
2. 使用 `sudo dpkg -i <package_name>` 命令安装 DEB 包。

   例如，安装悦数图探索需要运行以下命令，默认安装路径为 `/usr/local/yueshu-explorer`：

```
sudo dpkg -i yueshu-explorer-3.5.1.x86_64.deb
```

> **Note**
>
> 使用 DEB 包安装悦数图探索时不支持自定义安装路径。

3. 进入解压后的文件夹，在 `config` 目录内修改 `app-config.yaml` 文件，设置 `LicenseManagerURL` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。
4. （可选）配置 Dag Controller。参见下文 配置 **Dag Controller** 部分。
5. 进入 `yueshu-explorer` 文件夹，执行以下命令启动服务。

```
cd yueshu-explorer

# 启动悦数图探索。
sudo ./lib/start.sh

# （可选）启动 Dag Controller。
sudo ./dag-ctrl/scripts/start.sh
```

查看服务状态

```
sudo systemctl status yueshu-explorer.service
```

停止服务

```
sudo systemctl stop yueshu-explorer.service
```

卸载

使用以下的命令卸载悦数图探索。

Copyright © Yueshu Graph Technology - All rights reserved.

```
sudo dpkg -r yueshu-explorer
```

TAR 包部署

安装及部署

1. 根据需要下载 TAR 包，建议选择最新版本。
2. 使用 `tar -xvf` 解压 tar 包。

```
tar -xvf yueshu-explorer-<version>.tar.gz
```

3. 进入解压后的文件夹，在 `config` 目录内修改 `app-config.yaml` 文件，设置 `LicenseManagerURL` 的值为 LM 所在的主机 IP 和端口号 `9119`，例如 `192.168.8.100:9119`。
4. （可选）配置 Dag Controller。参见下文 配置 **Dag Controller** 部分。
5. 进入 `yueshu-explorer` 文件夹，执行以下命令启动服务。

```
cd yueshu-explorer

# 启动悦数图探索和 Dag Controller。
sudo ./scripts/start.sh

# 单独启动悦数图探索。
sudo nohup ./yueshu-explorer-server > explorer.log 2>&1 &
```

停止服务

用户可以采用 `kill pid` 的方式来关停服务：

```
kill $(lsof -t -i :7002)
```

配置 Dag Controller

Dag Controller 是一款任务编排调度工具，可以编排调度有向无环图（DAG）类型的作业，该作业由多个任务组成，且任务之间存在先后关系，组成一个有向无环图（DAG）。

Dag Controller 可以结合悦数图计算进行复杂的图计算。例如 Dag Controller 将算法调用请求发送给悦数图计算，悦数图计算保存结果至悦数图数据库或 HDFS，Dag Controller 再将上次的计算结果作为下一个算法任务的输入创建新的任务。

配置步骤

1. 配置 Dag Controller 机器 SSH 免密登录悦数图计算机器，以及悦数图计算集群内所有节点间的 SSH 相互免密登录。

   例如机器 A（Dag Controller）通过 SSH 免密登录至悦数图计算集群 B 中的机器 B-1。请在机器 A 上执行如下命令：

```
//执行后按提示生成密钥，默认按回车即可。
$ ssh-keygen -t rsa

//将机器 A 的公钥文件安装到机器 B-1 对应的用户下，即可从机器 A 免密登录机器 B-1。
$ ssh-copy-id -i ~/.ssh/id_rsa.pub <B_user>@<B_IP>
```

   按同样方法设置 A 免密登录机器 B-2、B-3 等，以及集群 B 内所有机器的互相免密登录。

2. 在 Dag Controller 机器上执行 `eval $(ssh-agent)` 启动 ssh-agent，然后执行 `ssh-add ~/.ssh/id_rsa` 将私钥交给 ssh-agent 管理。

   Note

   ssh-agent是密钥管理器，用来管理多个密钥，并为其他需要使用 SSH 密钥对的程序提供代理。

3. 配置 `dag-ctrl-api.yaml` 文件，路径为 `dag-ctrl/etc/dag-ctrl-api.yaml`。配置悦数图计算机器的用户名及端口，如果有多台机器，请确保使用相同用户名和端口。

```
# 配置名称。
Name: task-api

Host: 0.0.0.0     # Dag Controller 服务的 IP。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```yaml
Port: 9002        # Dag Controller 服务的端口。
Timeout: 60000    # HTTP 接口请求的超时时间。

Log:              # 日志打印相关参数。详情参见 https://go-zero.dev/cn/docs/blog/tool/logx/
  Mode: file      # 保存模式。
  KeepDays: 7     # 保存时长。
  Path: logs      # 保存目录。
  Level: info     # 保存级别。
  Compress: false  # 是否压缩。

# 悦数图计算机器的用户名以及 SSH 端口。
SSH:
 UserName: vesoft
 Port: 22

# 任务和作业的并行线程池大小。
JobPool:
 Sleep: 3    # 3 秒检查一次有没有未执行的作业。
 Size: 3    # 同时可以执行 3 个作业。
TaskPool:
 CheckStatusSleep: 1    # 1 秒检查一次任务状态。
 Size: 10    #同时可以执行 10 个任务。
Dag:
 VarDataListMaxSize: 100    # 如果读取 HDFS 的列，则限制为每次 100 条数据。

# 其他
Debug:
  Enable: false  #是否开启 Debug。

# 悦数图探索和 Dag Controller 通信的密钥。无需修改。
RsaPriKey: |
  -----BEGIN RSA PRIVATE KEY-----
  MIICXAIBAAKBgQDcR0keIMmmV...
  -----END RSA PRIVATE KEY-----
RsaPubKey: |
  -----BEGIN RSA PUBLIC KEY-----
  MIGJAoGBANxHSR4gyaZX7uet7...
  -----END RSA PUBLIC KEY-----
```

4. 配置 `tasks.yaml` 文件，路径为 `dag-ctrl/etc/tasks.yaml` 。只需配置算法文件的具体路径（`exec_file` 参数）。当前所有 `exec_file` 参数都配置为 `run_algo.sh` 文件的路径。

> **Note**

- 算法文件在悦数图计算安装路径下的 `scripts` 目录内。
- 如果有多台机器，请确保算法文件路径一致。
- 其它参数是算法的执行参数，后续在可视化工作流页面配置。

```yaml
exec_file: /home/xxx/yueshu-analytics/scripts/run_algo.sh
```

后续操作

连接悦数图探索

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.2.2 连接数据库

在成功启动悦数图探索后，用户需要配置连接悦数图数据库。默认情况下可以直接连接数据库。为保证数据安全，还支持 OAuth2.0 认证，认证通过后才可以连接数据库。

前提条件

在连接悦数图数据库前，用户需要确认以下信息：

- 已经安装部署了悦数图探索。详细信息，参见部署悦数图探索。
- 悦数图数据库 的 Graph 服务本机 IP 地址以及服务所用端口。默认端口为 `9669` 。

-悦数图数据库登录账号信息，包括用户名和密码。

- 建议使用 Chrome 89 及以上的版本的 Chrome 浏览器，否则可能有兼容问题。

OAuth2.0 认证设置

Caution

当前 OAuth2.0 认证设置为 Beta 功能，后续可能会有一定优化调整。

Note

如果想直接连接数据库，请参见后文连接数据库部分。

如果需要开启 OAuth2.0 认证，需要在悦数图探索安装目录内修改配置文件。路径为 `config/app-config.yaml` 。

OAuth 部分的配置说明如下。

参数	示例	说明
`Enable`	`false`	是否开启 OAuth2.0 认证。
`ClientID`	`4953xxx-mmnoge13xx.apps.googleusercontent.com`	应用的 ClientId。
`ClientSecret`	`GOCxxx-xaytomFexxx`	应用的 ClientSecret。
`RedirectURL`	`http://dashboard.vesoft-inc.com/login`	重定向到悦数运维监控的 URL。
`AuthURL`	`https://accounts.google.com/o/oauth2/auth`	认证 URL。
`TokenURL`	`https://oauth2.googleapis.com/token`	获取 access_token 的 URL。
`UserInfoURL`	`https://www.googleapis.com/oauth2/v1/userinfo`	获取用户信息的 URL。
`UsernameKey`	`email`	用户名字段。
`Organization`	`vesoft company`	组织名称。
`TokenName`	`oauth_token`	Cookie 里的 Token 名称。
`Scope`	`email`	OAuth 的权限范围。权限范围需要是厂商 OAuth2.0 平台配置的 scope 的子集，否则请求会失败。请求的 scope 需要能获取到 `UsernameKey` 的值。
`AvatarKey`	`picture`	用户信息里的 Avatar Key。

Copyright © Yueshu Graph Technology - All rights reserved.

配置完成后重启悦数图探索服务，登录页面会先展示 OAuth 认证页面，通过后才能继续连接数据库。

Copyright © Yueshu Graph Technology - All rights reserved.

连接数据库

按以下步骤连接悦数图数据库:

Copyright © Yueshu Graph Technology - All rights reserved.

1. 在浏览器地址栏输入 `http://<ip_address>:7002` 。

   在浏览器窗口中看到以下登录界面表示已经成功部署并启动了悦数图探索。

Note

首次登录悦数图探索的时候，页面显示最终用户许可协议的内容，请仔细阅读并单击同意。

2. 在悦数图探索的配置数据库页面上，输入以下信息：

- **Graphd IP** 地址：填写悦数图数据库的 Graph 服务本机 IP 地址。例如 `192.168.10.100` 。

Note

- 即使悦数图数据库与悦数图探索部署在同一台机器上，用户也必须填写这台机器的本机 IP 地址，而不是 `127.0.0.1` 或者 `localhost` 。
- 在新的标签页连接另一个悦数图数据库时，会覆盖旧标签页的会话。如果需要同时登录多个悦数图数据库，可以用不同的浏览器或者无痕模式。

- **Port**：Graphd 服务的端口。默认为 `9669` 。
- 用户名和密码：根据悦数图数据库的身份验证设置填写登录账号和密码。
- 如果未启用身份验证，可以填写默认用户名 `root` 和任意密码。
- 如果已启用身份验证，但是未创建账号信息，用户只能以 GOD 角色登录，必须填写用户名 `root` 和密码 `nebula` 。
- 如果已启用身份验证，同时又创建了不同的用户并分配了角色，不同角色的用户使用自己的账号和密码登录。

3. 完成设置后，点击登录按钮。

Note

一次连接会话持续 30 分钟。如果超过 30 分钟没有操作，会话即断开，用户需要重新登录数据库。

首次登录会显示欢迎页，根据使用流程展示相关功能，并且支持自动下载并导入测试数据集。

Copyright © Yueshu Graph Technology - All rights reserved.

想要再次访问欢迎页，在 下选择新手引导。

断开连接

在页面右上角，选择 图标 > 清空连接。

如果浏览器上显示配置数据库页面，表示悦数图探索已经成功断开了与悦数图数据库的连接。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.3 页面概览

本文介绍悦数图探索的基本组成，帮忙用户了解悦数图探索的各功能。

悦数图探索由顶部导航栏、左侧导航栏、和画布三大模块组成。

17.3.1 顶部导航栏

图标/元素	说明
悦数图探索	可视化探索及分析数据。详情参见开始探索、节点筛选、探索拓展、图计算。
Visual Query	可视化构造场景进行数据查询。详情参见可视化查询。
Workflow	可视化构造自定义工作流进行复杂图计算。需要在 里开启工作流才能显示 Workflow 页面。详情参见工作流简介。
	在画板上自行设计 Schema，可以直观展示点边关系。详情参见Schema 草图。
	对悦数图数据库图空间进行管理。详情参见创建 Schema。
	将数据批量导入悦数图数据库。详情参见导入数据。
	对悦数图数据库内的数据使用 nGQL 进行查询操作。详情参见控制台。
	nGQL 语句的模板列表。详情参见查询语句模板。
	数据库用户管理。详情参见数据库用户管理。
	全局设置。可以设置悦数图探索页面的语言、开启 Beta 功能、画布查询结果最大数量等。
	帮助页面，指导和帮忙用户使用悦数图数据库。
	反馈页面。可以反馈故障、提交建议、参与调研或联系悦数图数据库团队。
	显示连接信息和版本信息。支持修改密码和退出登录。

Copyright © Yueshu Graph Technology - All rights reserved.

17.3.2 左侧导航栏

Note

登录悦数图探索后，用户需选择并单击一个目标图空间，然后才可解锁左侧导航栏的查询及分析功能。详情参见选择图空间。

用户可以通过单击悦数图探索页面左侧的图标进行图数据导入、图数据分析与探索等操作。左侧导航栏的图标及描述如下：

图标	说明
	输入 VID、Tag 和子图等信息后，匹配的数据会显示到画布上。更多信息，参见查询方式。
	对画布中显示的点进行过滤。更多信息，参见节点筛选。
	选择页面上的节点并进行自定义拓展，包括方向、步数、过滤条件等。更多信息，参见探索拓展。
	选择页面上至少两个点并查看它们的共同邻居。更多信息，参见探索拓展。
	查询起点到终点之间的所有路径、最短路径和非循环路径。更多信息，参见探索拓展。
	选择是否显画布中的点或边的属性值。更多信息，参见探索拓展。
	基于画布中的点边进行图计算。更多信息，参见图计算。
	基于画布中已聚合的边进行属性计算。更多信息，参见属性计算。
	查看历史快照信息。更多信息，参见画布快照。
	查看所有图空间，单击图空间可创建相应图空间的画布。更多信息，参见选择图空间。

17.3.3 画布

Note

登录悦数图探索后，用户需选择并单击一个目标图空间，才可进入画布页面。详情参见选择图空间。

图数据可视化地展示在画布中。悦数图探索的画布由以下部分组成：

- 顶部页签
- 可视化模式
- 数据存储
- 搜索框
- 多布局
- 缩略图
- 数据概览

更多信息，参见画布操作。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.4 数据库管理

17.4.1 Schema 草图

悦数图探索支持 Schema 草图功能。用户可以在画板上自行设计 Schema，可以直观展示点边关系，设计完成后可以将 Schema 应用到指定图空间。

功能说明

- 图形化设计 Schema。
- 应用 Schema 到指定图空间。
- 导出 Schema 为 PNG 格式图片。

入口

在顶部导航栏里，单击 图标。

设计 **Schema**

以 `basketballplayer` 的 Schema 为例，说明如何设计 Schema。

1. 在页面左上角单击新建。
2. 在画布下方选择合适颜色的 Tag 标签，按住左键拖拽至画布中，创建一个 Tag。
3. 单击 Tag，在右侧填写标签名称 `player` 、描述，并且添加属性 `name` 和 `age` 。
4. 再次创建一个 Tag，标签名称为 `team` ，属性为 `name` 。
5. 从 Tag `player` 上的锚点连接至 Tag `team` 的锚点，单击生成的边，在右侧填写边类型名称 `serve` ，并且添加属性 `start_year` 和 `end_year` 。
6. 从 Tag `player` 上的锚点连接至自身另一个锚点，单击生成的边，在右侧填写边类型名称 `follow` ，并且添加属性 `degree` 。
7. 设计完成后，在页面上方单击 修改草图名称，然后在右上角单击 保存草图。

Copyright © Yueshu Graph Technology - All rights reserved.

应用 Schema

1. 在页面左侧的草图列表内选择需要导入的 Schema 草图，然后在右上角单击应用到图空间。
2. 选择将 Schema 导入新建图空间或者已创建的图空间，单击确认。

Note

- 创建图空间的参数说明参见CREATE SPACE。
- 如果图空间中已有重名 Schema，导入操作会失败，并提示修改名称或更换图空间。

修改 Schema

在页面左侧的草图列表内选择需要修改的 Schema 草图，修改完成后在右上角单击 保存。

Note

已应用 Schema 的图空间不会同步修改。

删除 Schema

在页面左侧的草图列表内找到需要删除的 Schema 草图，在缩略图右上角单击**X**并确认即可删除。

导出 Schema

在页面右上角单击 可以导出 Schema 为 PNG 格式图片。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.4.2 Schema 管理

悦数图探索支持界面化方式管理 Schema。

> **Note**

- 用户可以使用 Schema 草图功能图形化设计 Schema。详情参见 Schema 草图。
- 用户可以在控制台直接执行 nGQL 命令管理 Schema。

前提条件

- 账号拥有 GOD、ADMIN 或 DBA 权限。详情参见 悦数图数据库内置角色。
- 已经规划 Schema 的要素。

> **Note**

本示例已经创建图空间。如果账号拥有 GOD 权限，也可以在 控制台 或 **Schema** 上创建一个图空间。

入口

在顶部导航栏里，单击 图标。

创建图空间

1. 单击创建图空间。
2. 设置参数。参数介绍参见 CREATE SPACE。
3. 单击创建。

创建标签/边类型

1. 在图空间列表中，找到图空间，单击图空间名称或者在操作列中单击**Schema**。
2. 单击标签或边类型页签，单击创建。
3. 设置参数。参数介绍参见CREATE TAG和CREATE EDGE。

> **Note**

未设置索引时，用户才可以设置 TTL。详情参见 TTL 配置。

4. 单击创建。

在标签和边类型列表里，可以执行修改和删除操作。

Copyright © Yueshu Graph Technology - All rights reserved.

创建索引

> **Note**
>
> - 创建索引前需要确保相关的 Tag 或 Edge type 已创建。
> - 由于索引会影响写性能，建议首次导入大批量数据时，先导入数据，再创建索引并重建索引。详情参见索引介绍。

1. 在图空间列表中，找到图空间，单击图空间名称或者在操作列中单击**Schema**。
2. 单击索引页签，单击创建。
3. 设置参数。参数介绍参见CREATE INDEX。

> **Note**
>
> 索引属性的顺序会影响 `LOOKUP` 语句的查询结果。详情参见 LOOKUP。

4. 单击创建。

在索引列表里，可以重建索引或删除索引。

查看统计信息

1. 在图空间列表中，找到图空间，单击图空间名称或者在操作列中单击**Schema**。
2. 单击统计页签，单击更新。

查看 **Schema**

> **Note**
>
> 需要在 里开启查看 **Schema** 才能显示该功能。

1. 在图空间列表中，找到图空间，单击图空间名称或者在操作列中单击**Schema**。
2. 单击查看 **Schema**页签，单击获取 **Schema**。

其他操作

在图空间列表中，找到图空间，在操作列可以执行如下操作：

- 查看 Schema DDL：显示该图空间的 Schema 创建语句，包括图空间、Tag、Edge type 和索引。
- 克隆图空间：克隆该图空间的 Schema 到新的图空间。
- 删除图空间：删除该图空间，包括 Schema 和所有点边数据。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.4.3 数据导入

悦数图探索支持界面化地将 CSV 格式数据导入至悦数图数据库中。

前提条件

- 悦数图数据库里已经创建 Schema。
- CSV 文件符合 Schema 要求。
- 账号拥有 GOD、ADMIN、DBA 或 USER 权限。详情参见 悦数图数据库内置角色。

入口

在顶部导航栏里，单击 图标。

操作步骤

导入数据主要分为 2 个部分，新建数据源和创建导入任务，接下来将详细介绍。

新建数据源

在页面右上角单击新建数据源，设置数据来源及其相关设置。当前支持 3 种类型的数据源。

数据源类型	说明
云存储	添加云存储作为 CSV 文件来源，只支持兼容 Amazon S3接口的云服务。
SFTP	添加 SFTP 作为 CSV 文件来源。
本地文件	上传本地 CSV 文件。文件大小不能超过 200 MB，超过限制的文件请放入其他方式的数据源中。

Note

- 上传本地 CSV 文件时，一次可以选择多个 CSV 文件。
- 数据源添加后，可以在页面上方单击数据源管理，切换页签即可查看不同类型的数据源详情，也可以编辑或删除数据源。

创建导入任务

1. 在页面左上角单击创建导入任务，完成如下设置：

Copyright © Yueshu Graph Technology - All rights reserved.

Caution

用户也可以单击导入模版，下载示例配置文件 `example.yaml` ，配置后再上传配置文件。配置方式与 NebulaGraph Importer大致相同。

- 图空间：需要导入数据的图空间名称。
- 任务名称：默认自动生成，可以修改。
- 更多配置（可选）：可以自定义设置并发数、批处理量、重试次数、读取并发数和导入并发数。
- 关联标签：

a. 单击添加 **Tag**，然后在下方新增的标签内选择 Tag。

b. 单击添加导入文件，在文件源里选择数据源类型和文件路径，找到需要导入的文件，然后单击添加。

c. 在预览页面设置文件的分隔符和是否携带表头，然后单击确认。

d. 在**VID** 列为 VID 选择对应的列。支持选择多个列合并为 VID，也可以为 VID 添加前缀或后缀。

e. 在属性框内为属性选择对应的列。对于可以为 `NULL` 或设置了 `DEFAULT` 的属性，可以不指定对应的列。

f. 重复 b ~ e 步骤将步骤 1 所选 Tag 的数据文件全部导入。

g. 重复 a ~ f 步骤将所有需要导入的 Tag 数据全部导入。

- 关联边：与关联标签的操作相同。

2. 完成设置后，单击导入,输入悦数图数据库账号的密码并确认。

导入任务创建后，可以在导入数据页签内查看导入任务的进度，支持编辑任务、查看日志、下载日志、重新导入、下载配置文件、删除任务等操作。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.4.4 控制台

悦数图探索的控制台功能允许用户手动输入 nGQL 语句，并可视化地呈现查询结果。本文介绍控制台的界面。

入口

在顶部导航栏里，单击 图标。

页面介绍

Copyright © Yueshu Graph Technology - All rights reserved.

下表列出了控制台界面上的各种功能。

编号	功能	说明
1	选择图空间	选择执行 nGQL 的图空间。不支持在控制台执行 `USE <space_name>` 语句切换图空间。
2	收藏夹	点击 按钮，展开收藏夹，点击其中一个语句，输入框中即自动输入该语句。
3	历史清单	点击 按钮，在语句运行记录列表里，点击其中一个语句，输入框中即自动输入该语句。列表里提供最近 15 次语句运行记录。
4	清空输入框	点击 按钮，清空输入框中已经输入的内容。
5	运行	在输入框中输入 nGQL 语句后，点击 按钮即开始运行语句。
6	保存为模板	将输入框中输入的 nGQL 语句保存为模板。详情参见查询语句模板。
7	输入框	输入 nGQL 语句的区域。可以同时输入多个语句按顺序执行，语句之间以 `;` 分隔。支持用 `//` 添加注释。
8	自定义参数展示	点击 按钮可展开查看自定义参数，用于参数化查询。详情信息可见管理参数。
9	语句运行状态	运行 nGQL 语句后，这里显示语句运行状态。如果语句运行成功，语句以绿色显示。如果语句运行失败，语句以红色显示。
10	添加到收藏夹	点击 按钮，将语句存入收藏夹中，已收藏的语句该按钮以黄色展示。
11	导出 CSV 文件或 PNG 格式图片	运行 nGQL 语句返回结果后，返回结果为表格形式时，点击 按钮即能将结果以 CSV 文件的形式导出。切换到可视化窗口，点击 按钮即能将结果以 CSV 文件或 PNG 图片的形式导出。
12	展开/隐藏执行结果	点击 按钮，隐藏此条 nGQL 语句返回的结果。
13	关闭执行结果	点击 按钮，关闭此条 nGQL 语句返回的结果。
14	表格窗口	显示语句运行结果。如果语句会返回结果，窗口会以表格形式呈现返回的结果。
15	可视化窗口	显示语句运行结果。如果语句会返回完整的点边结果，窗口会以可视化形式呈现返回的结果。点击右方 按钮，展开数据概览面板。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.4.5 查询语句模板

悦数图探索支持将常用 nGQL 语句保存为模板，方便自己或他人使用。nGQL 语句中的文本还支持参数化，可根据需要填写参数值。

前提条件

悦数图数据库里已经创建 Schema。

入口

在顶部导航栏里，点击 图标。

新建模板

1. 单击 + 新模板，设置如下参数。

参数	示例	说明
模板名称	`test`	模板的名称。
图空间	`basketballplayer`	模板适用的图空间。
描述	`返回指定球员的邻居名称`	描述模板的作用。
查询模板	`MATCH (v:player{name:"${name}"})--(v2:player) RETURN v2.player.name AS Name;`	nGQL 语句模板。可以选中需要参数化的文本，在右侧单击 + 文本参数化，设置参数名称和描述。示例中 `${name}` 为参数化的文本，实际使用时可以填写 `Tim Duncan` 等名称。 可以在单行使用 `//` 添加注释。
输入	-	显示参数化的文本内容。可以编辑或删除参数化的文本。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

控制台页面左上角单击 **+** 保存为模板，会自动将已输入的查询语句作为模板语句。

2. 单击保存为模板。

其他操作

- 目标模板右侧单击 可以修改模板内容。
- 目标模板右侧单击 可以自动跳转至控制台并输入模板。
- 目标模板右侧单击 可以删除模板。
- 右上角筛选框可以筛选指定图空间的模板。
- 右上角搜索框可以搜索模板名称。

使用模板

- （推荐）在图探索页面使用模板。详情参见开始探索。
- 在模板列表页面单击 自动跳转至控制台并输入模板语句，需要自行修改参数化的文本。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.4.6 数据库用户管理

悦数图探索支持管理数据库用户，包括创建用户、删除用户、修改密码等。

前提条件

登录悦数图探索的用户需要拥有相关操作的权限。例如 `God` 权限的用户可以执行所有操作，`Admin` 权限的用户可以将权限内的图空间授权给其他用户。关于角色权限详情，参见角色权限。

入口

在顶部导航栏里，点击 图标。

创建用户

Note

仅 `root` 用户可以创建用户。

1. 在标签用户列表内单击创建用户，设置如下参数。

参数	说明
账号	用户名。
密码	用户名对应密码。
IP 白名单	用户只有通过列表内的 IP 才能连接数据库。多个 IP 之间用英文逗号（,）分隔。

Note

单击左上角添加，可以批量创建用户。

2. 单击确认。

授权用户

1. 切换至标签授权，在左上角选择需要授权给用户的图空间名称。该页面会展示拥有该图空间权限的所有用户（`root` 用户除外）。

Copyright © Yueshu Graph Technology - All rights reserved.

2. 单击授权用户，设置如下参数。

参数	说明
用户名	设置需要授权的用户名。如果登录的是 `root` 用户，可以在下拉菜单中选择用户；如果是 `Admin` 权限的用户，需要手动填写用户名。
角色	在下拉菜单中选择要授权的角色。关于角色权限详情，参见角色权限。

3. 单击确认。

用户列表其他操作

> **Note**
>
> 仅 `root` 用户可以查看用户列表。

- 查看：查看用户在各个空间内的权限。
- 编辑：修改用户对应的密码和 IP 白名单。修改时不需要提供旧密码。非 `root` 用户无法查看用户列表，可以在页面右上角的 图标内修改密码。
- 删除用户：仅 `root` 用户可以删除其他用户。
- 搜索账号：通过关键字搜索账号。

授权列表其他操作

- 编辑：修改用户的角色。
- 取消授权：撤销用户的角色。
- 搜索账号：通过关键字搜索账号。

> **Note**
>
> 修改或撤销角色后，对应的用户需要重新登录才能生效。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.5 图探索

17.5.1 选择图空间

用户使用悦数图探索的功能前，需要先选择一个图空间，然后才能对图空间中的数据进行查询和分析。本文介绍如何选择图空间。

前提条件

已连接悦数图探索。具体操作，参见连接悦数图探索。

操作步骤

登录悦数图探索后，系统首先会显示图空间选择页面，用户只需选择目标图空间即可。

之后如果要再次选择图空间，可以通过以下方式：

1. 在悦数图探索页面左侧的导航栏中，单击图空间 图标。
2. 选择目标图空间。

> **Note**
>
> 用户可以多次选择相同或者不同的图空间，每次选择都会创建新的画布。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.5.2 开始探索

为了探索图数据，用户需要查询出一些初始数据，在这些初始数据的基础上，才能进一步的分析、过滤。本文介绍如何查询初始数据。

前提条件

已选择目标图空间。详情参见选择图空间。

操作步骤

在悦数图探索页面左侧单击开始图标 然后查询数据。查询到的数据会显示在画布中。查询方式如下：

- VID 查询
- Tag 查询
- 子图查询
- 模板查询

VID 查询

用户可输入点的 ID（VID）查询目标点。

支持手动输入、随机导入、文件导入 VID。

> **Note**
>
> 查询区域中每行仅支持填入一个 VID，用回车键分隔。

TAG 查询

用户可以选择 Tag 和对应索引查询目标点，还可以设置结果数量限制和筛选条件。

> **Note**
>
> 请确保对应的图空间中已存在相应的 Tag 和索引，否则无法选择。详情参见创建 Tag 和创建索引。

下图为查询 10 个年龄大于 30 岁，且不等于 40 岁的球员的示例。

Copyright © Yueshu Graph Technology - All rights reserved.

子图查询

用户可以输入一个或多个 VID，指定查询的步数、边类型及流入流出的方向查询子图数据。VID 为必选项，可选项步数默认值为 1，边类型默认值为全部。

> **Note**
>
> 当输入多个 VID 时，VID 之间以 `Enter` 键隔开。

下图为 VID 值为 `Kings` 和 `Suns`，步数为 `2` ，边类型为 `server` 和 `like` 的入边的示例。

Copyright © Yueshu Graph Technology - All rights reserved.

模板查询

用户可以选择已创建的查询语句模板，并填写模板参数值。

Copyright © Yueshu Graph Technology - All rights reserved.

- 返回结果为点时，画布上会展示这些点。
- 返回结果不为点时，会用表格形式展示结果。例如返回球员姓名、年龄等。

关于查询语句模板详情，请参见查询语句模板。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.5.3 节点筛选

用户可以根据过滤条件筛选画布中的点和边。支持仅通过 Tag 或通过一组或多组筛选条件对画布中显示的点进行过滤。

前提条件

使用节点筛选功能前，确保画布中有点数据。具体操作，参见开始探索。

使用说明

- 使用 **Tag** （标签）过滤画布中的点及关联的边时：
- 筛选条件面板显示图空间中的所有 Tag。
- Tag 图例被过滤（置灰）时，画布中的对应 Tag 的点及关联的边会被隐藏。
- 对于多 Tag 的点，如果其任一 Tag 被过滤，则该点会被隐藏。
- 支持通过搜索框搜索 Tag。

- 使用筛选条件过滤画布中的点及关联的边时：
- 每一组筛选条件只针对带此 Tag 的数据，筛选条件包括：Tag、属性、运算符、值。如满足条件，画布中的点会被自动添加选中状态，不满足条件的点及关联的边可被设置为隐藏或置灰。其他 Tag 数据状态不受影响。
- 通过标签过滤（置灰）的优先级最高。如果筛选条件中配置了已通过标签过滤（置灰）的 Tag，画布中不会展示对应的数据。
- 每次只能选择一个 Tag。如果要过滤多个 Tag，需要多次进行添加筛选条件操作。
- 筛选条件不支持对同一个 Tag 进行多次过滤，只会显示第一次过滤的结果。

示例

示例**1** 过滤画布中 **TAG** 为 **PLAYER** 的所有点

1. 在左侧导航栏中，单击节点筛选图标 。
2. 在筛选条件面板中，单击**player**。
3. 画布中显示只有 Tag 为 team 的点。

上图中被过滤出的橙色点即 Tag 为 team 的点。

示例**2** 筛选画布中年龄大于 **33** 岁的球员

1. 在左侧导航栏中，单击节点筛选图标 。
2. 单击添加筛选条件，然后设置筛选项（本示例设置的值分别为 `player` 、 `age` 、 `>` 、 `33` ）。
3. 单击置灰将不满足过滤条件的点显示为灰色。

Copyright © Yueshu Graph Technology - All rights reserved.

4. 打开启动筛选开关。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.5.4 探索拓展

图数据库的探索拓展功能分为以下四类：

- 拓展
- 共同邻居
- 路径查询
- 查看属性

前提条件

使用探索拓展的所有功能前，确保画布中有点数据。具体操作，参见开始探索。

拓展

1. 在左侧导航栏，单击 图标，打开拓展窗口。在窗口内设置拓展条件，包括边类型、方向、点样式、步数或更精确的条件，说明如下。

参数	描述
边类型	默认显示并选中图空间中的所有边。
方向	定义所有选中点的边的方向来查询目标点，包括 `流出` 、 `流入` 、 `双向` 三个选项。
点的样式	`按标签类型分类` ： 目标点的显示颜色和相应的 Tag 颜色一致。 `自定义样式` ： 自定义目标点的显示颜色。
步数	`单步` ： 自定义选中点至目标点的跳数。 `范围` ： 自定义选中点至目标点的范围。
筛选条件	根据筛选条件查询目标点。

2. 选择需要进行拓展的点，可以按住右键框选或按住 `Shift` 键单击左键选择画布中的多个点，然后在拓展窗口单击拓展。对于单个点，可以直接在点上双击左键拓展。

> **Note**
>
> 系统会在面板内的配置被修改后保存当前配置，当双击点或者右键对点快捷拓展时候会以当前配置进行拓展。

共同邻居

在左侧导航栏，单击 图标，打开共同邻居窗口。用户可以按住右键框选或按住 `Shift` 键单击左键选择画布中的两个点或多个点，查询它们的共同邻居。选中的点无共同邻居时，默认返回没有相应数据。

Copyright © Yueshu Graph Technology - All rights reserved.

路径查询

1. 在左侧导航栏，单击 图标，打开路径查询窗口。在窗口内设置边类型、方向、查询类型等，说明如下。

参数	描述
边类型	默认显示并选中图数据库中的所有边。
方向	定义起始点的边的方向来查询和目标点的路径，包括 `流出` 、 `流入` 、 `双向` 三个选项。
查询类型	`全路径` ： 显示从起点至终点所有路径的点和边。 `最短路径` ： 显示从起点至终点的最短路径的点和边。 `非循环路径` ： 显示从起点至终点的非循环路径的点和边。
选中步骤	起始点至终点的步数限制。
筛选条件	根据筛选条件筛选目标点。

2. 按住 `Shift` 键单击左键选中画布中的两个点，默认第一个点为起点，第二个点为终点。然后在路径查询窗口单击查询路径。

查看属性

在左侧导航栏，单击 图标，打开查看属性窗口。用户可以选择在画布中展示或隐藏点或边的属性。

> **Note**

- 缩放比例大于 90% 时候在画布上才会显示点属性，小于 90% 的时候会自动隐藏。
- 缩放比例大于 100% 时候在画布上才会显示边属性，小于 100% 的时候会自动隐藏。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.5.5 图计算

为了更好地挖掘、分析图数据，用户可以基于画布中的点边进行图计算，并直接查看图计算结果。

Note

本功能仅对画布中现存的点进行图计算，如果需要进行复杂图计算，推荐使用 Workflow 进行复杂的可视化图计算。

前提条件

确保画布中有图计算所需的点边数据。具体操作，参见开始探索。

操作步骤

1. 在左侧导航栏，单击 图标，打开图计算窗口。
2. 选择算法并设置相关参数。关于算法和参数的说明，请参见算法简介。
3. 单击运行，画布下方会弹出图计算结果。
4. 在结果页面可以：

- 单击自动补齐一度路径关系将画布中所有点之间的一度路径关系补齐。
- 单击导出**CSV**下载 CSV 格式的图计算结果文件。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.5.6 属性计算

当画布中的点存在大量关系时，为了增强图的可读性和可分析性，可以将起点、终点、边类型相同的边进行聚合。聚合后的边可以对其中的某个属性进行计算并展示。

前提条件

画布中有聚合边。

注意事项

- 当前仅支持求和计算。
- 只支持聚合 INT 类型的属性。
- 支持选择多个 Edge type 分别进行聚合。
- 支持选择多个属性分别进行聚合。
- 一条边只能显示一个聚合结果。可以将鼠标悬停在聚合边上查看所有结果。

操作步骤

方式 1

1. 在左侧导航栏，单击 图标，打开属性计算窗口。
2. 单击 + 号，设置边类型、属性和计算方式。可以选择多个属性分别进行聚合。
3. 单击确认。

单击 + 号可以增加更多 Edge type 的属性计算。

Copyright © Yueshu Graph Technology - All rights reserved.

方式 2

1. 在画布中的聚合边上单击鼠标右键，选择属性计算。
2. 设置属性和计算方式。
3. 单击确认。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.6 可视化查询

可视化查询帮助用户通过构造查询场景，查询目标数据并查看与之匹配的语句。用户只需拖拽和连线即可构造可视化查询语句，系统将查询结果展示在画布上。

Compatibility

不兼容 3.0.0 以下版本的悦数图数据库。

Note

悦数图探索的可视化查询功能目前处于测试阶段。

17.6.1 前提条件

- 已选择图空间。具体操作，参见选择图空间。
- 部分查询需先创建索引。详情参见MATCH语句注意事项和创建索引。

17.6.2 页面元素

在悦数图探索页面顶部，单击 **Visual Query** 进入可视化查询页面。在可视化查询页面左侧可以看到图空间对应的所有 Tag (如 player 和 team) 和名为任意标签 Tag。

Note

任意标签的 Tag 也可以用于查询无 Tag 的点。

Copyright © Yueshu Graph Technology - All rights reserved.

在可视化页面中，其他图标的描述如下表。

图标/元素	描述
	框选的点和边即为需要查询的结果，双击边框可限制查询数量（优先级高于全局设置中的最大返回数的值）。目前不支持只查询边。
	放大可视化查询页面。
	缩小可视化查询页面。
	保存当前的可视化查询图，保存的图被缓存在浏览器中。
	查看保存的可视化查询图，最多查看 10 份最近保存的可视化查询图。单击任一保存的图可将其显示在可视化查询页面中。
nGQL	单击 **nGQL** 为构图显示查询语句。
查询	单击查询将查询结果可视化地显示在画布中。

17.6.3 操作步骤

1. 在可视化查询页面左侧拖动若干个目标 Tag 到右侧创建对应 Tag 的点。
2. 单击点，在点边缘的锚点上按住鼠标左键拖动至另一个点，创建对应的边。
3. 单击点配置过滤条件。说明如下。

- 标签名：设置零、一个或多个 Tag。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

一个点可以有 0 到多个 Tag：

- 设置 0 个 Tag 时，表示查询无 Tag 的点。
- 设置一个 Tag 时，表示查询该 Tag 的点。
- 设置多个 Tag 时，表示查询同时包含所有设置的 Tag 的点。

- 筛选条件：设置一组或多组针对点的过滤条件，包括点的属性、运算符、属性值。

Note

当配置多 Tag 的点时，不支持设置筛选条件查询数据。

4. 单击边配置过滤条件。说明如下。

- 边类型：设置一个或多个 Edge Type。

Note

一条边有且仅有一个 Edge Type：

- 设置一个 Edge Type 时，表示查询该 Edge Type 的边。
- 设置多个 Edge Type 时，表示查询包含任一 Edge Type 的边。

- 方向：设置边的方向，包括流出、流入、双向。
- 单步：设置定长路径。
- 范围：设置变长路径。
- 筛选条件：设置一组或多组针对边的过滤条件，包括边的属性、运算符、属性值。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

当配置多个 Edge Type 查询数据时，不支持设置筛选条件查询数据。

5. 查询场景(即模式)构建完成后，单击 并框选需要返回（RETURN）的结果。
6. 单击可视化查询页面右上角的查询，画布中会显示查询结果。

17.6.4 示例

示例 **1**

查询和球员 Yao Ming 相互关注，且年龄大于 35 岁的球员以及这些球员都效忠哪些球队，并限制最后查询结果中的效忠关系数为 6。

1. 构造查询模式，拖拽 Tag 至可视化查询面板中（2 个 player 和 1 个 team）。
2. 配置过滤条件。

a. 设置第一个球员的过滤条件为 player.name == Yao Ming。

b. 设置两个球员之间边的边类型为 follow、双向、步数为 1。

c. 设置第二个球员的过滤条件为 player.age > 35。

d. 设置第二个球员和球队之间边的边类型为 serve，方向为流出，步数为 1。

e. 单击 选中第二个球员、球队及它们之间的 serve 边。

f. 单击模式查询框，然后设置结果数量限制为 6。

3. 单击查询。系统将 6 条查询关系结果显示至画布中。

示例 **2**

查询两个相互关注的球员共同效忠什么球队，并且查询此球队所有年龄大于 30 岁的球员。

Copyright © Yueshu Graph Technology - All rights reserved.

1. 构造查询模式，拖拽 Tag 至面板中（3 个 player 和 1 个 team）。
2. 配置过滤条件。

a. 设置第一个和第二个球员之间边的边类型为 follow、双向、步数为 1。

b. 设置第一个球员和球队之间的边的边类型为 serve、方向为流出、步数为 1。

c. 设置第二个球员和球队之间的边的边类型为 serve、方向为流出、步数为 1。

d. 设置第三个球员的过滤条件为 player.age > 30。

e. 设置第三个球员和球队之间的边的边类型为 serve、方向为流出、步数为 1。

f. 单击 选中第三个球员、球队及它们之间的 serve 边。

3. 单击查询。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.7 画布功能

17.7.1 画布概览

用户可在画布中可视化地探索图数据。本文主要介绍画布的组成及相关功能。

画布概览图如下：

顶部页签

用户可以通过单击画布顶部加号 + 图标同时操作多个画布。

- 不同画布上的数据可以来自同一个图空间也可以来自不同的图空间。
- 除了最左边页签的第一个画布外，用户可以自定义画布的名称。

可视化模式

用户可以选择 **2D** 模式和 **3D** 模式可视化地在画布上探索图数据。更多信息，参见可视化模式。

数据存储

用户可以通过创建快照，或以图片、CSV 文件的方式导出画布以存储当前画布数据。

Copyright © Yueshu Graph Technology - All rights reserved.

在页面右上方：

- 单击 创建快照。更多信息，参见画布快照。
- 单击 ，然后单击导出 **CSV** 以 CSV 文件的方式存储画布数据。
- 单击 ，然后单击导出 **PNG** 以图片形式存储画布数据。

搜索框

在页面右上方，单击 ，然后在搜索框中输入 VID 或者 Tag 的属性值定位目标点。

多布局

悦数图探索支持 6 种布局方式展示画布上的数据之间的关系。

力导向图	层次图	环形图	网格	神经网络	辐射

缩略图

全屏展示画布中的图，收起缩略图，缩小或放大画布中的图等。同时在缩略图的左下角显示了画布中的图占总图的百分比。

Copyright © Yueshu Graph Technology - All rights reserved.

数据概览

在页面右侧，单击 < 展开数据概览面板。

Copyright © Yueshu Graph Technology - All rights reserved.

- 可以在数据概览面板中看到画布中的 Tag 数、Edge type 数，及对应的点和边的数量。
- 在数据概览面板，单击 Tag 或 Edge type 颜色图标，可自定义颜色、尺寸，Tag 还支持自定义图标、图片。

Note

数据概览面板中只支持批量修改颜色。在画布中右键单击单个点可以手动修改单个点的样式。

Copyright © Yueshu Graph Technology - All rights reserved.

- 支持上传图片以个性化标记画布中的点的样式，上传的图片被存在浏览器中。如果需要永久保存上传的图片，可以将当前画布中的数据创建为快照，具体操作，参见数据快照。

选中画布中的点和边，然后在画布左下角单击选中的点 **{number}**，选中的边 **{number}**查看选中的点或边的数据信息并以 CSV 文件方式导出数据。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.7.2 可视化模式

悦数图探索支持 **2D** 和 **3D** 两种可视化模式探索数据。2D 模式支持数据查看和图操作；3D 模式通过不同视角对图数据进行探索，适用于数据量级比较高或者需要演示的场合。

Note

3D 模式下，不可进行图数据的操作。

2D 模式

2D 模式下可对画布上的数据进行探索操作：

参数	说明
自动	出入度权重：根据画布中所有点的出边和入边的条数，自动调整点的大小。 重置出入度：重置画布中的点至原始大小。 边聚合：自动聚合画布上所有符合聚合规则的边。 取消边聚合：重置画布上已经聚合的边。
信息检测	孤立点：检测画布中所有的无边点。 悬边：检测画布中与度数为1的点关联的边（包含关联点）。 环检测：检测由边和点连接在一起形成循环的路径。 N 步节点检测：从选中点开始，逐步显示出方向的点。
聚合	聚合节点之间的边：聚合画布中选中的点之间的边。 节点取消边聚合：重置画布中选中的点之间的聚合边。
编辑	隐藏：隐藏画布中选中的点边。 隐藏其他：隐藏画布中未选择的所有点边。 撤销：撤销上一步新增或隐藏的操作。 重做：恢复上一步撤销的操作。

Copyright © Yueshu Graph Technology - All rights reserved.

有关更多的 2D 模式下可进行的操作，参见画布功能。

边聚合说明

当画布中的点存在大量关系时，为了增强图的可读性和可分析性，可以将起点、终点、边类型相同的边进行聚合，使得点与点之间关系更加清晰。

- 边聚合会自动显示聚合的边数量。
- 边聚合支持对其中的属性进行计算。详情参见属性计算。
- 鼠标悬停在聚合边上，会显示边类型、聚合的边数量、边属性和值。如果进行了属性计算，还会显示计算结果。
- 除了在上方操作栏取消边聚合，还可以双击聚合边或者在聚合边上右键选择取消边聚合。

3D 模式

在页面左上方，单击视图切换开关可切换至 3D 视图模式。3D 视图模式下，可切换回到 2D 图探索模式，并且不会对 2D 的探索产生任何副作用。

参数	说明
鸟瞰	以鸟瞰的方式展示当前图空间的所有数据。默认最多展示图空间中 20000 节点，2000 边。单击 ⌄ 可以调整，但是设置过大可能会导致浏览器崩溃。
画质	高：点以球的形式展示，光影效果更好。 低：点以圆的形式展示，可以支持更大数据量。
重新受力	当不同的点重叠时，分散点与点之间的距离。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.7.3 管理快照

悦数图探索的快照功能可保存图探索得到的可视化结果，以便再次打开浏览器时还原保存的图数据。

创建快照

1. 在画布右上角，单击相机 图标。
2. 填写快照名称和备注（选填）。
3. 单击提交。

> **Note**
>
> 已创建的快照被存放在快照历史中。更多信息，参见下文。

快照历史

> **Note**
>
> 目前最多存储 50 个历史快照。

在悦数图探索页面左侧导航栏，单击 图标进入快照列表页面。用户可切换图空间查看相应图空间的历史快照，还可以导入历史快照至画布，下载历史快照至本地，以及删除快照。

在目标历史快照右侧的操作列下：

- 单击 导入历史快照至新画布中。
- 单击 以 JSON 的格式下载历史快照至本地。
- 单击 删除历史快照。

在快照列表左上方，单击导入快照可导入之前下载的 JSON 格式快照文件至快照列表中，以便用户之间离线共享快照数据。系统会根据 JSON 文件中记录的图空间信息将导入的快照自动放置于相应的图空间中。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.8 工作流

17.8.1 工作流简介

悦数图探索支持用自定义工作流进行可视化的复杂图计算。

背景信息

悦数图探索提供多种组件（Component），包括图查询组件和图计算组件，用户可以基于编排调度工具 Dag Controller 自由组合这些组件，例如将图查询组件的输出作为图计算组件的输入，最终形成一个有向无环的工作流（Workflow）。

需要进行图计算时再将工作流实例化，组件实例化后称为任务（Task），工作流实例化后称为作业（Job）。一个作业可以由多个任务构成。悦数图探索将作业发送给悦数图计算进行图计算，完成后可以在作业列表查看对应的图计算结果。

功能说明

- 支持新增、查看、修改、删除、对比、克隆、重命名工作流。
- 支持查询组件和多种算法组件，可以搜索、新增、配置、重命名组件。
- 支持查看作业列表、进度、结果、日志，以及重跑作业。
- 支持搜索工作流或作业。

注意事项

- 需要在 里开启工作流才能显示 Workflow 页面。
- 使用工作流需要额外部署 Dag Controller 和悦数图计算。详情参见悦数图计算和部署悦数图探索。
- 图查询组件的输入只能是查询语言。
- 图查询组件的结果默认支持保存在 NFS（即本地），也可以保存在 HDFS，方便被多个算法调用。
- 图计算组件的输入可以是悦数图数据库或 HDFS 的指定数据，也可以依赖于图查询组件的结果。

  如果依赖于前一个图查询组件的结果，必须和该图查询组件全连接，即该图计算组件的白色输入锚点全都和上一个图查询组件的白色输出锚点连接。
- 部分算法的参数配置也可以依赖于上游节点。
- 图计算组件结果允许保存在 HDFS、悦数图数据库中，但是并非所有算法的结果都适合存入到悦数图数据库里，在设置结果保存页面时，部分算法只能选择 HDFS。

Copyright © Yueshu Graph Technology - All rights reserved.

算法参数配置说明

请参见算法简介。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.8.2 资源配置

使用工作流之前，需要进行资源配置，包括悦数图数据库配置、HDFS 配置和悦数图计算配置。

前提条件

- 已部署 3.5.0 或以上版本的悦数图计算。详情参见悦数图计算。
- 已配置并启动 Dag Controller。详情参见部署悦数图探索。

操作步骤

1. 在悦数图探索页面顶部的导航栏中，单击 **Workflow**。
2. 在页面右上角单击工作流配置。
3. 配置如下资源:

Copyright © Yueshu Graph Technology - All rights reserved.

- 悦数 配置

  执行图查询或写入图计算结果的 Graph 服务地址。默认为登录 Explorer 使用的 Graph 服务，无法修改。可以设置三种服务的超时时间。

- DAG 配置

  执行图计算的 Dag Controller 的相关配置。

- 用户名：运行服务的用户名，无需修改。
- 数据路径：Analytics 数据目录，NFS 服务的共享目录。工作流默认使用 NFS 方式存储图计算结果，但是需要用户手动安装 NFS 并挂载目录。
- SSH 免密私钥路径：Dag Controller 所在机器的私钥文件路径。用于机器间 SSH 免密登录。

- 悦数 Analytics 节点配置

  添加执行图计算的 NebulaGraph Analytics 地址。

- Nebula Analytics 节点 IP 地址：填写新的 Analytics 节点 IP 地址。
- 用户名：免密访问该 Analytics 节点所使用的用户名。所有 Analytics 节点的用户名必须保持一致。
- Analytics 节点 SSH 端口号：默认为 `22` 。
- Analytics 节点用户 SSH 免密私钥路径：用于机器间 SSH 免密登录。默认为 `~/.ssh/id_rsa` 。
- Analytics 节点本地数据目录：默认为 `~/analytics-data` 。
- Analytics 节点算法脚本路径：默认为 `~/nebula-analytics/scripts/run_algo.sh` 。

- NFS 配置

> **Note**
>
> 用户需要自行将 NFS Server 部署在 Dag Controller 所在机器上并配置共享目录，然后将 NFS Client 部署在所有 Analytics 节点机器上并挂载共享目录。

默认开启，任务运行结果将保存至本地。每个 Analytics 节点都需要配置 NFS 的本地数据目录。

Copyright © Yueshu Graph Technology - All rights reserved.

- HDFS 配置（可选）

  默认情况下使用的是 NFS 保存图计算结果，如果需要使用 HDFS，请先在 Analytics 所在机器上安装 HDFS 客户端，该配置才能生效。
- HDFS 名称：填写 HDFS 配置名称，方便区分不同 HDFS 配置。
- HDFS 路径：即 `fs.defaultFS` 配置。支持配置保存路径，例如 `hdfs://192.168.8.100:9000/test` 。
- HDFS 用户名：使用 HDFS 的用户名称。

4. 配置完成后，单击保存。
5. 在右上角单击配置检查，然后单击开始检查，检查配置是否正常。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.8.3 工作流示例

本文介绍如何创建一个简单的工作流。

前提条件

- 已有数据源。数据源可以是悦数图数据库中的数据或者 HDFS 上的 CSV 文件。
- 已完成资源配置。

Copyright © Yueshu Graph Technology - All rights reserved.

新增工作流

以 MATCH 语句 `MATCH (v1:player)--(v2) RETURN id(v1), id(v2);` 的结果做为 PageRank 算法的输入，介绍如何创建一个简单的工作流。

Copyright © Yueshu Graph Technology - All rights reserved.

1. 在悦数图探索页面顶部的导航栏中，单击 **Workflow**。
2. 在工作流标签页单击新建工作流，进入流程画布页面。
3. 在左侧组件库列表里选择查询-**>Query**，按住左键拖拽至画布中，单击该图查询组件，在右侧弹出的工作流配置面板中进行如下设置。

配置项	说明
Query	单击 可以修改组件名称，方别识别。
查询语言	选择执行nGQL语句的图空间，以及填写nGQL语句。填写语句后单击解析参数，会在输出里展示返回的列名。
输入	设置自定义参数，可以用于参数化查询。单击添加参数可以增加更多自定义参数。
输出	解析查询语言得到的返回结果列名。可以修改名称，相当于用 `AS` 设置列的别名。
结果	设置结果的保存位置。为方便其他算法调用结果，图查询组件的结果只支持保存在 NFS/HDFS 上。

Note

连接锚点显示为黄色，表示可选，可以自行设置或者由其他任意组件提供输入。

4. 在左侧组件库列表里选择节点重要度-**>PageRank**，按住左键拖拽至画布中，将图查询组件的 `output0` 锚点连接至图计算组件的 `input0` 锚点， `output1` 锚点连接至 `input1` 锚点。

Copyright © Yueshu Graph Technology - All rights reserved.

如果串联使用多个图查询组件，需要自行添加参数化文本。例如语句 `GO FROM ${id} OVER follow YIELD dst(vertex)` ，填写语句后单击 `解析参数` ，图查询组件中会显示黄色锚点表示 `${id}` 。前一个图查询组件的输出锚点可以连接至该黄色锚点作为后一个组件的输入。

5. 单击该图计算组件，在右侧弹出的工作流配置面板中进行如下设置。

Copyright © Yueshu Graph Technology - All rights reserved.

配置项	说明
PageRank	单击 可以修改组件名称，方别识别。
输入	支持多种数据源作为输入。 **Graph Database** 可以选择图空间和对应的边类型。 **Dependence** 会根据锚点的连接情况自动识别。 **HDFS**可以选择 HDFS 和数据源文件的相对路径。 **Local**可以选择 NFS 中数据源文件的相对路径。
参数配置	设置图算法特有的参数。不同算法的参数不同，部分参数可以从任意上游组件获取，此时锚点显示为黄色。
输出	显示图计算结果的列名，无法修改。
执行配置	机器数量表示将要执行算法的机器数量。 进程数量表示将要执行算法的总进程数，会根据机器数量平均分配到各个机器。 线程数量表示每个进程会启动多少个线程。
结果	设置结果的保存位置。可以保存在 HDFS 或悦数图数据库中。 保存为 HDFS 时只需要选择 HDFS ，保存路径无需修改，会根据作业和任务 ID 自动生成。 保存为悦数图数据库时，需要预先在相应图空间中创建 Tag 用于保存结果。Tag 的属性说明请参见算法简介。 部分算法只支持保存在 HDFS 上。

6. 在画布左上角自动生成的工作流名称旁单击 修改工作流名称，然后在画布右上角单击运行，会自动跳转至作业页面显示作业进度，耐心等待作业完成即可查看结果。详情参见作业管理。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

单击运行时，会自动进行保存，如果不执行图计算，仅仅修改，修改完成后请单击 进行保存，或者单击 另存为新的工作流。

常见问题

如何删除组件?

单击鼠标左键选择要删除的组件，单击 `Backspace` 即可删除该组件。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.8.4 工作流管理

本文介绍如何管理工作流，包括查看、修改、重命名、克隆、删除、对比。

操作步骤

1. 在悦数图探索页面顶部的导航栏中，单击 **Workflow**。
2. 在工作流标签页可以查看所有已保存的工作流，页面显示工作流名称、创建时间、最近一次修改时间、包含的图算法。

- 在列表上方单击对比，可以选择两个工作流或者相同工作流的不同历史版本进行代码的对比。
- 在列表上方的搜索框，可以通过关键字搜索工作流。
- 在列表的操作列，可以执行如下操作：
- 运行：直接将该工作流实例化为作业执行。
- 打开：打开工作流查看，可以修改工作流，修改完后请单击 进行保存，或者单击 另存为新的工作流。
- 查看作业：跳转至作业列表查看该工作流实例化的所有作业。
- ⋮ ：可以查看工作流的代码、重命名工作流、克隆工作流和删除工作流。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.8.5 作业管理

本文介绍如何查看作业列表、进度、结果、日志，以及重跑作业。

操作步骤

1. 在悦数图探索页面顶部的导航栏中，单击 **Workflow**。
2. 在作业标签页可以查看所有执行的作业，页面显示作业 ID、作业名称、状态、创建时间、结束时间、工作流版本。

- 在列表上方单击对比，可以选择两个工作流或者相同工作流的不同作业进行代码的对比。
- 在列表上方的筛选框，可以筛选工作流和版本。
- 在列表上方的搜索框，可以通过关键字搜索作业。

- 在列表的操作列，可以执行如下操作：
- 在悦数图探索中打开：对于执行成功的作业，可以选择图空间和组件，查看组件的输出结果。支持将结果导出为 CSV 文件。
- 重跑：对于执行失败的作业，可以重跑该作业。
- 打开：可以重跑作业、查看作业结果、日志，以及跳转到工作流中进行编辑（工作流为最新版本）。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.8.6 工作流 API

工作流 **API** 概览

悦数图探索提供 API 接口使用工作流的部分功能。

当前支持的 API 接口如下：

- 新增作业
- 获取所有作业列表
- 获取指定工作流的作业列表
- 查询指定作业详情
- 取消作业运行
- 获取指定任务的运行结果数据

请求方式

支持使用 curl 调用 API 接口实现对应的功能。

格式如下：

```
curl <options> http://<explorer_address>:<explorer_port>/<api_path>?{<body>}
```

- `<options>`： curl 支持大量选项，工作流使用较多的是 `-X` 、 `-H` 、 `-d` 。关于选项的详细说明，参见 curl 官方文档。
- `<explorer_address>`： 悦数图探索访问地址。
- `<explorer_port>`： 悦数图探索访问端口。
- `<api_path>`： API 的调用路径。例如 `api-open/v1/jobs` 。
- `<body>`： 调用 API 时传入的 Body 参数。

获取授权 **TOKEN**

使用 API 时，需要做 Token 信息校验。请使用如下命令获取 Token 信息。

```
curl -i -X POST -H "Content-Type: application/json" -H "Authorization: Bearer <account_base64_encode>" -d '{"address":"<database_address>","port":<database_port>}' http://
<explorer_address>:<explorer_port>/api-open/v1/connect
```

- `<account_base64_encode>`： Base64 编码后的悦数图数据库账号和密码字符串。以用户名 `root` 、密码 `123` 为例，序列化后的字符串为 `["root","123"]` ，Base64 编码后为 `WyJyb290IiwiMTIzIl0=` 。
- `<database_address>`： 悦数图数据库访问地址。
- `<database_port>`： 悦数图数据库访问端口。
- `<explorer_address>`： 悦数图探索访问地址。
- `<explorer_port>`： 悦数图探索访问端口。

示例：

```
curl -i -X POST -H "Content-Type: application/json" -H "Authorization: Bearer WyJyb290IiwiMTIzIl0=" -d '{"address":"192.168.8.111","port":9669}' http://192.168.8.145:7002/api-open/v1/
connect
```

返回结果：

```
HTTP/1.1 200 OK
Content-Type: application/json
Set-Cookie: explorer_token=eyJhbxxx; Path=/; # Max-Age=259200; HttpOnly
Traceparent: 00-1c3f55cdbf81e13a2331ed88155ce0bf-2b97474943563f20-# 00
Date: Thu, 14 Jul 2022 06:47:01 GMT
Content-Length: 54

{
  "code": 0,
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  "data": {
    "success": true
  },
  "message": "Success"
}
```

需要关注的参数如下：

- `explorer_token`： Token 信息。
- `Max-Age`： Token 有效时间。单位：秒。默认为 259200 秒，即 3 天。可以在安装目录内的 `config/app-config.yaml` 文件内修改默认有效时间。

请求结果

- API 如果调用成功，会返回如下信息：

```
{
  code: 0,
  message: 'Success',
  data: <ResponseData> //根据接口返回相应结果。
}
```

- API如果调用失败，会返回对应的通用错误码，例如：

```
{
  code: 40004000,
  message: '<ErrBadRequest>',  //返回具体报错信息。
}
```

通用错误码的说明参见下文。

通用错误码

错误码	信息	说明
40004000	`ErrBadRequest`	请求异常
40004001	`ErrParam`	请求参数异常
40104000	`ErrUnauthorized`	请求未授权
40104001	`ErrSession`	登录会话异常
40304000	`ErrForbidden`	请求被拒绝
40404000	`ErrNotFound`	请求资源不存在
50004000	`ErrInternalServer`	内部服务异常
50004001	`ErrInternalDatabase`	数据库异常
50004002	`ErrInternalController`	控制器异常
50004003	`ErrInternalLicense`	证书校验异常
90004000	`ErrUnknown`	未知错误

Copyright © Yueshu Graph Technology - All rights reserved.

作业/任务状态码

状态码	说明
0	准备中
1	执行中
2	执行成功
3	执行失败
4	已中断
5	暂停中

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

新增作业

本文介绍如何使用工作流的 API 新增作业。

API 路径

`api-open/v1/workflows/<workflow_id>/jobs`

`<workflow_id>`：工作流 ID。参见下文的请求参数。

请求参数

路径参数

参数	类型	是否必填	默认值	示例	说明
`workflow_id`	number	必填	-	`4216617528`	工作流 ID。需要指定工作流将其实例化为作业。可以在指定的工作流页面左上角查看 ID。

Headers 参数

参数	类型	是否必填	默认值	示例	说明
`Content-Type`	string	必填	-	`application/json`	内容类型。
`explorer_token`	string	必填	-	`eyJhbxxx`	授权 Token，用于验证账号信息。如何获取授权 Token 请参见工作流 API 概览。

Body 参数

Note

自定义的传入参数需要用户自行保证参数的合理性和正确性，否则作业会执行失败。

参数	类型	是否必填	默认值	示例	说明
`input`	object	可选	-	-	自定义的传入参数。
- `task_id`	object	可选	-	`query_1`	任务 ID。可以在组件设置页面右上角查看。一个任务可以设置多个由键值对表示的参数。
- `param_name: param_value`	string: {string 或 number}	可选	-	`param0: player100`	`param_name` 为参数的键，即参数名。`param_value` 为参数的值。

请求示例

以下图为例，在 nGQL 语句使用自定义参数 `name`。在创建作业时传入参数值 `Tim Duncan`。

Copyright © Yueshu Graph Technology - All rights reserved.

```
curl -i -X POST -H "Content-Type: application/json" -H "Cookie: "explorer_token=eyJhbxxx"" -d '{"input":{"query_1":{"name":"Tim Duncan"}}}' http://192.168.8.145:7002/api-open/v1/workflows/4216617528/jobs
```

返回参数

参数	类型	示例	说明
`code`	number	`0`	请求结果码。请求成功返回 `0`，请求不成功返回对应的错误码。详情参见工作流 API 概览。
`message`	string	`Success`	执行结果信息。
`data`	object	-	返回的数据列表。
- `id`	string	`107`	新增作业的 ID。

返回示例

```
{
  "cookie": [],
  "Content-Type": "application/json",
  "Traceparent": "00-1ba128615cdc2226c921973a689e9f1b-7630b12963494672-00",
  "Date": "Fri, 15 Jul 2022 07:19:25 GMT",
  "Content-Length": "48"
}

{
  "code": 0,
  "data": {
    "id": 107
  },
  "message": "Success"
}
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

获取所有作业列表

本文介绍如何使用工作流的 API 获取所有作业列表。

API 路径

`api-open/v1/jobs`

请求参数

路径参数

无。

Headers 参数

参数	类型	是否必填	默认值	示例	说明
`Content-Type`	string	必填	-	`application/json`	内容类型。
`explorer_token`	string	必填	-	`eyJhbxxx`	授权 Token，用于验证账号信息。如何获取授权 Token 请参见工作流 API 概览。

Body 参数

参数	类型	是否必填	默认值	示例	说明
`filter`	object	可选	-	-	过滤器的设置。
- `name`	string	可选	-	`workflow_q745a_20220715092236`	作业名称。
- `status`	number	可选	-	`2`	作业状态码。详情参见工作流 API 概览。
- `fromCreateTime`	number	可选	-	`1657848036000`	起始时间戳。基于作业的创建时间进行过滤。
- `toCreateTime`	number	可选	-	`1657848157000`	结束时间戳。基于作业的创建时间进行过滤。
- `orderByCreateTime`	string	可选	`desc`	-	排序方式。支持 `desc` 和 `asc` 。
`pageSize`	number	可选	`10`	-	每页记录数。
`page`	number	可选	`1`	-	页码。

请求示例

Note

`jobs?` 后的内容为 Body 参数，`filter` 的内容是进过 URL 编码的结果。原始内容为：`{ "status": 2, "orderByCreateTime": "asc"}` 。

```
curl -i -X GET -H "Content-Type: application/json" -H "Cookie: "explorer_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/jobs?
filter=%7B%20%22status%22%3A%202%2C%20%20%22orderByCreateTime%22%3A%20%22asc%22%7D&pageSize=10&page=1
```


Copyright © Yueshu Graph Technology - All rights reserved.

返回参数

参数	类型	示例	说明
`code`	number	`0`	请求结果码。请求成功返回 `0`，请求不成功返回对应的错误码。详情参见工作流 API 概览。
`message`	string	`Success`	执行结果信息。
`data`	object	-	返回的数据列表。
- `total`	number	`2`	记录总数。
- `Page`	number	`1`	页码。
- `PageSize`	number	`10`	每页记录数。
- `items`	object	-	记录详情列表。
- `id`	number	`105`	作业 ID。
- `name`	string	`workflow_q745a_20220715090915`	作业名称。
- `workflowId`	string	`4216617528`	工作流 ID。
- `workflowName`	string	`workflow_q745a`	工作流名称。
- `status`	number	`2`	作业状态码。详情参见工作流 API 概览。
- `runBeginTime`	number	`1657847358000`	作业开始执行时间。
- `runEndTime`	number	`1657847364000`	作业执行结束时间。
- `createTime`	number	`1657847355906`	作业创建时间。

返回示例

```
{
  "cookie": [],
  "Content-Type": "application/json",
  "Traceparent": "00-d3a1943f5baf46771e9afc629e0b5d40-920db2f06142f5ff-00",
  "Date": "Fri, 15 Jul 2022 06:17:21 GMT",
  "Content-Length": "512"
}

{
  "code": 0,
  "data": {
    "items": [
      {
        "id": 105,
        "name": "workflow_q745a_20220715090915",
        "workflowId": "4216617528",
        "workflowName": "workflow_q745a",
        "status": 2,
        "runBeginTime": 1657847358000,
        "runEndTime": 1657847364000,
        "createTime": 1657847355906
      },
      {
        "id": 106,
        "name": "workflow_q745a_20220715092236",
        "workflowId": "4216617528",
        "workflowName": "workflow_q745a",
        "status": 2,
        "runBeginTime": 1657848157000,
        "runEndTime": 1657848163000,
        "createTime": 1657848156290
      }
    ],
    "total": 2,
    "Page": 1,
    "PageSize": 10
  },
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  "message": "Success"
}
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

获取指定工作流的作业列表

本文介绍如何使用工作流的 API 获取指定工作流的作业列表。

API 路径

`api-open/v1/workflows/<workflow_id>/jobs`

`<workflow_id>`： 工作流 ID。参见下文的请求参数。

请求参数

路径参数

参数	类型	是否必填	默认值	示例	说明
`workflow_id`	number	必填	-	`4216617528`	工作流 ID。需要指定工作流将其实例化为作业。可以在指定的工作流页面左上角查看 ID。

Headers 参数

参数	类型	是否必填	默认值	示例	说明
`Content-Type`	string	必填	-	`application/json`	内容类型。
`explorer_token`	string	必填	-	`eyJhbxxx`	授权 Token，用于验证账号信息。如何获取授权 Token 请参见工作流 API 概览。

Body 参数

参数	类型	是否必填	默认值	示例	说明
`filter`	object	可选	-	-	过滤器的设置。
- `name`	string	可选	-	`workflow_q745a_20220715092236`	作业名称。
- `status`	number	可选	-	`2`	作业状态码。详情参见工作流 API 概览。
- `fromCreateTime`	number	可选	-	`1657848036000`	起始时间戳。基于作业的创建时间进行过滤。
- `toCreateTime`	number	可选	-	`1657848157000`	结束时间戳。基于作业的创建时间进行过滤。
- `orderByCreateTime`	string	可选	`desc`	-	排序方式。支持 `desc` 和 `asc` 。
`pageSize`	number	可选	`10`	-	每页记录数。
`page`	number	可选	`1`	-	页码。

请求示例

Note

`jobs?` 后的内容为 Body 参数， `filter` 的内容是进过 URL 编码的结果。原始内容为： `{"status": 2, "fromCreateTime": 1657874100000}` 。

```
curl -i -X GET -H "Content-Type: application/json" -H "Cookie: "explorer_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/workflows/4216617528/jobs?
filter=%7B%22status%22%3A%202%2C%20%20%22fromCreateTime%22%3A%201657874100000%7D&pageSize=10&page=1
```


Copyright © Yueshu Graph Technology - All rights reserved.

返回参数

参数	类型	示例	说明
`code`	number	`0`	请求结果码。请求成功返回 `0`，请求不成功返回对应的错误码。详情参见工作流 API 概览。
`message`	string	`Success`	执行结果信息。
`data`	object	-	返回的数据列表。
- `total`	number	`2`	记录总数。
- `Page`	number	`1`	页码。
- `PageSize`	number	`10`	每页记录数。
- `items`	object	-	记录详情列表。
- `id`	number	`105`	作业 ID。
- `name`	string	`workflow_q745a_20220715090915`	作业名称。
- `workflowId`	string	`4216617528`	工作流 ID。
- `workflowName`	string	`workflow_q745a`	工作流名称。
- `status`	number	`2`	作业状态码。详情参见工作流 API 概览。
- `runBeginTime`	number	`1657847358000`	作业开始执行时间。
- `runEndTime`	number	`1657847364000`	作业执行结束时间。
- `createTime`	number	`1657847355906`	作业创建时间。

返回示例

```
{
  "cookie": [],
  "Content-Type": "application/json",
  "Traceparent": "00-008c3056686dd3f3be38b8eda42a917e-b5616e30434cb803-00",
  "Date": "Fri, 15 Jul 2022 08:44:06 GMT",
  "Content-Length": "297"
}
{
  "code": 0,
  "data": {
    "items": [
      {
        "id": 115,
        "name": "workflow_q745a_20220715163650",
        "workflowId": "4216617528",
        "workflowName": "workflow_q745a",
        "status": 2,
        "runBeginTime": 1657874212000,
        "runEndTime": 1657874218000,
        "createTime": 1657874210088
      }
    ],
    "total": 1,
    "Page": 1,
    "PageSize": 10
  },
  "message": "Success"
}
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

查询指定作业详情

本文介绍如何使用工作流的 API 查询指定作业详情。

API 路径

`api-open/v1/jobs/<job_id>`

`<job_id>`：作业 ID。参见下文的请求参数。

请求参数

路径参数

参数	类型	是否必填	默认值	示例	说明
`job_id`	number	必填	-	`1964`	作业 ID。可以通过 API 获取所有作业列表查询，或者在作业列表页面查看。

Headers 参数

参数	类型	是否必填	默认值	示例	说明
`Content-Type`	string	必填	-	`application/json`	内容类型。
`explorer_token`	string	必填	-	`eyJhbxxx`	授权 Token，用于验证账号信息。如何获取授权 Token 请参见工作流 API 概览。

Body 参数

无。

请求示例

```
curl -i -X GET -H "Content-Type: application/jsom" -H "Cookie: "explorer_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/jobs/1964
```


Copyright © Yueshu Graph Technology - All rights reserved.

返回参数

参数	类型	示例	说明
`code`	number	`0`	请求结果码。请求成功返回 `0`，请求不成功返回对应的错误码。详情参见工作流 API 概览。
`message`	string	`Success`	执行结果信息。
`data`	object	-	返回的数据列表。
- `id`	number	`1964`	作业 ID。
- `name`	string	`workflow_xkkjf_20220712103332`	作业名称。
- `workflowId`	string	`3992429968`	工作流 ID。
- `workflowName`	string	`workflow_xkkjf`	工作流名称。
- `status`	number	`2`	作业状态码。详情参见工作流 API 概览。
- `tasks`	object	-	任务详情。
- `id`	string	`f93dea90fc3a11ecac7e6da0662c195b`	任务 ID。
- `name`	string	`BFS`	任务名称。
- `runBeginTime`	datetime	`2022-07-12T10:33:35+08:00`	任务开始执行时间。
- `runEndTime`	datetime	`2022-07-12T10:33:38+08:00`	任务执行结束时间。
- `status`	number	`2`	任务状态码。详情参见工作流 API 概览。

返回示例


```
{
  "cookie": [],
  "Content-Type": "application/json",
  "Traceparent": "00-3db17c9fd9e0a4c3824973471523d214-4384705e523dce83-00",
  "Date": "Fri, 15 Jul 2022 09:08:20 GMT",
  "Content-Length": "400"
}
{
  "code": 0,
  "data": {
    "id": 1964,
    "name": "workflow_xkkjf_20220712103332",
    "workflowId": "3992429968",
    "workflowName": "workflow_xkkjf",
    "status": 2,
    "tasks": [
      {
        "id": "f93dea90fc3a11ecac7e6da0662c195b",
        "name": "BFS",
        "runBeginTime": "2022-07-12T10:33:35+08:00",
        "runEndTime": "2022-07-12T10:33:38+08:00",
        "status": 2
      }
    ],
    "runBeginTime": 1657593215000,
    "runEndTime": 1657593218000,
    "createTime": 1657593212505
  },
  "message": "Success"
}
```


最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

终止执行指定作业

本文介绍如何使用工作流的 API 终止执行指定作业。

API 路径

`api-open/v1/jobs/<job_id>/cancel`

`<job_id>`：作业 ID。参见下文的请求参数。

请求参数

路径参数

参数	类型	是否必填	默认值	示例	说明
`job_id`	number	必填	-	`1964`	作业 ID。可以通过 API 获取所有作业列表查询，或者在作业列表页面查看。

Headers 参数

参数	类型	是否必填	默认值	示例	说明
`Content-Type`	string	必填	-	`application/x-www-form-urlencoded`	内容类型。
`explorer_token`	string	必填	-	`eyJhbxxx`	授权 Token，用于验证账号信息。如何获取授权 Token 请参见工作流 API 概览。

Body 参数

无。

请求示例

```
curl -i -X PUT -H "Content-Type: application/x-www-form-urlencoded" -H "Cookie: "explorer_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/jobs/30600/cancel
```

返回参数

参数	类型	示例	说明
`code`	number	`0`	请求结果码。请求成功返回 `0`，请求不成功返回对应的错误码。详情参见工作流 API 概览。
`message`	string	`Success`	执行结果信息。
`data`	object	-	返回的数据列表。
- `success`	bool	`true`	是否成功终止。

返回示例

```
{
  "cookie": [],
  "Content-Type": "application/json",
  "Traceparent": "00-8b4b47413a211d9b5e0839aadc712052-4a98bae37fe5948a-00",
  "Date": "Mon, 18 Jul 2022 01:45:08 GMT",
  "Content-Length": "54"
}
{
  "code": 0,
  "data": {
    "success": true
  },
  "message": "Success"
}
```


Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

获取指定任务的运行结果

本文介绍如何使用工作流的 API 获取指定任务的运行结果。

API 路径

`api-open/v1/jobs/<job_id>/tasks/<task_id>/sample_result`

- `<job_id>`：作业 ID。参见下文的请求参数。
- `<task_id>`：任务 ID。参见下文的请求参数。

请求参数

路径参数

参数	类型	是否必填	默认值	示例	说明
`job_id`	number	必填	-	`29987`	作业 ID。可以通过 API 获取所有作业列表查询，或者在作业列表页面查看。
`task_id`	number	必填	-	`8c171f70fb6f11ecac7e6da0662c195b`	任务 ID。可以通过 API 查询指定作业详情查询，或者在指定作业页面单击组件，在右上角查看。

Headers 参数

参数	类型	是否必填	默认值	示例	说明
`Content-Type`	string	必填	-	`application/x-www-form-urlencoded`	内容类型。
`explorer_token`	string	必填	-	`eyJhbxxx`	授权 Token，用于验证账号信息。如何获取授权 Token 请参见工作流 API 概览。

Body 参数

参数	类型	是否必填	默认值	示例	说明
`limit`	number	必填	`10`	-	返回结果行数限制。

请求示例

```
curl -i -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Cookie: "explorer_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/jobs/29987/tasks/
8c171f70fb6f11ecac7e6da0662c195b/sample_result?limit=1000
```

返回参数

参数	类型	示例	说明
`code`	number	`0`	请求结果码。请求成功返回 `0`，请求不成功返回对应的错误码。详情参见工作流 API 概览。
`message`	string	`Success`	执行结果信息。
`data`	object	-	返回的数据列表。
- `items`	list	-	详细结果列表。
- `result`	string	`"player110","0.150000"`	根据算法不同，结果可能是 2 列或 3 列。

Copyright © Yueshu Graph Technology - All rights reserved.

返回示例

```
{
  "cookie": [],
  "Content-Type": "application/json",
  "Traceparent": "00-14047b04b6810be06be22e010f500506-4c310a844b824a7f-00",
  "Date": "Fri, 15 Jul 2022 09:36:56 GMT",
  "Content-Length": "2014"
}
{
  "code": 0,
  "data": {
    "items": [
      [
        "player110",
        "0.150000"
      ],
      [
        "team219",
        "0.452126"
      ],
      ......
      [
        "player121",
        "0.262148"
      ]
    ]
  },
  "message": "Success"
}
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.9 内联框架

悦数图探索支持内联框架（iframe），可以将画布嵌入至第三方页面中使用。本文介绍如何嵌入画布。

17.9.1 前提条件

已安装悦数图探索。

17.9.2 注意事项

- 嵌入的悦数图探索页面默认直接访问目标图空间，因此会屏蔽部分页面及功能。例如上方导航栏、左侧导航栏中的模板查询和切换图空间等。如果需要访问多个图空间，可以在多个页面中分别嵌入。
- 暂不支持切换语言，默认为中文页面。

17.9.3 步骤

1. 在悦数图探索安装目录内修改配置文件 `config/app-config.yaml` 。需要修改的内容如下。

```
# 取消 CertFile 和 KeyFile 参数的注释。
CertFile: "./config/NebulaGraphExplorer.crt"
KeyFile: "./config/NebulaGraphExplorer.key"

# 修改 IframeMode.Enable为 true。
IframeMode:
  Enable: true
# 可以设置窗口的 URI 白名单，默认无限制。
  # Origins:
  #  - "http://192.168.8.8"
```

2. 在 `config` 文件夹内使用 `openssl` 命令生成自签名证书。示例如下。

```
openssl req -newkey rsa:4096 -x509 -sha512 -days 365 -nodes -subj "/CN=NebulaGraphExplorer.com" -out NebulaGraphExplorer.crt -keyout NebulaGraphExplorer.key
```

- `-newkey` ： 生成证书请求或者自签名证书的时候自动生成密钥。
- `-x509` ： 生成自签名证书。
- `-sha512` ： 指定消息摘要算法。
- `-days` ： `-x509` 生成的证书的有效天数。
- `-nodes` ： 不加密输出密钥。
- `-subj` ： 设置请求的主题。
- `-out` ： 指定生成的证书请求或者自签名证书名称。
- `-keyout` ： 指定自动生成的密钥名称。

3. 用户自行开发，在第三方页面中通过 iframe 方式嵌入悦数图探索。
4. 在父页面通过 postMessage 方法传递登录消息，格式如下：

```
{ type: 'NebulaGraphExploreLogin',
  data: {
    authorization: 'WyJyb29IiwibmVidWxhIl0=',
    host: '192.168.8.240:9669',
    space: 'basketballplayer'
    } }
```

- type：方法类型必须为 `NebulaGraphExploreLogin` 。
- data：
- `authorization` ： 悦数图数据库账号和密码组成数组并序列化，然后进行 Base64 编码。数组格式为 `['账号', '密码']` ，示例为 `['root', 'nebula']` ，编码后为 `WyJyb29IiwibmVidWxhIl0=` 。
- `host` ： 悦数图数据库的 Graph 服务地址。
- `space` ： 目标图空间名称。

Copyright © Yueshu Graph Technology - All rights reserved.

5. 启动悦数图探索服务。

> **Note**
>
> 如果是 RPM/DEB 安装的悦数图探索，请执行命令 `sudo ./yueshu-explorer-server &` 。

```
./scripts/start.sh
```

6. 访问第三方页面，检查是否可以查看到嵌入的悦数图探索页面。示例页面中第一个页面展示 `basketballplayer` 图空间，第二个和第三个页面展示其他图空间。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.10 系统设置

本文介绍悦数图探索的系统设置，包括全局设置和自定义设置。

17.10.1 全局设置

全局设置中包含语言设置、Beta 功能设置和画布查询最大返回数量设置。

- 语言设置：切换界面语言，支持中文和英文。
- Beta 功能：Beta 功能开关。Beta 功能目前包括工作流和查看 Schema。
- 画布查询最大返回数量：点边的查询结果在画布中显示的最大数量。

> **Note**
>
> 2D 模式的画布中展示的点边建议不超过 5000 个，否则渲染时可能会卡顿。

17.10.2 自定义设置

自定义设置中包含主页面 Logo、登录页 Logo 和产品名称设置。

- 主页面 Logo：支持上传图片作为主页面左上角的 Logo。
- 登录页 Logo：支持上传图片作为登录页中的 Logo。
- 产品名称：支持修改登录页中显示的产品名称。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.11 基本操作和快捷键

本文列出了悦数图探索的基本操作及快捷键。

17.11.1 基本操作

操作	说明
移动画布	在画布空白处鼠标左键拖拽移动。
放大或缩小画布	使用鼠标滚轮放大或缩小。
选中单个点或单条边	鼠标左键单击节点或边。
选中多个点或多条边	同时按住 Shift 和鼠标左键单击点或边。
批量选中数据	按住鼠标右键并框选数据；或者同时按住 Shift 和鼠标左键，并框选数据。
移动单个点	选中点按住鼠标左键拖拽。

17.11.2 快捷键

操作	说明
Enter	拓展
Shift + '-'	缩小
Shift + '+'	放大
Shift + 'l'	显示
Ctrl/Cmd + 'z'	撤销
Ctrl/Cmd + Shift + 'z'	重做
Ctrl/Cmd + 'a'	选中所有点
选中 + 'Backspace'	隐藏
选中 + Shift + 'Backspace'	隐藏其他

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

17.12 常见问题 FAQ

本文列出了使用悦数图探索时可能遇到的常见问题，用户可以使用文档中心或者浏览器的搜索功能查找相应问题。

17.12.1 如果 Graph 服务返回的查询结果数据量过大，会导致 Dag Controller 服务崩溃吗?

Dag Controller 服务仅仅提供调度功能，不会崩溃，但是数据量过大可能会导致悦数图计算服务读写 HDFS 或者悦数图数据库时内存不足而崩溃。

17.12.2 如果一个作业中的某个任务失败，能否从失败的任务开始重新执行?

暂不支持，只能整体重新执行。

17.12.3 如果任务结果保存很慢，或任务间数据传输很慢，如何加速?

Dag Controller 包含图查询组件和图计算组件。图查询是发送请求给 Graph 进程进行查询，因此只能增大 Graph 服务的内存进行加速；图计算是由悦数图计算提供的分布式节点进行计算，可以增大计算集群规模进行加速。

17.12.4 HDFS 服务器无法连接时，任务状态一直为 `running` 怎么办?

为 HDFS 连接设置超时时间、次数，配置如下：

```
<configuration>
<property>
    <name>ipc.client.connect.timeout</name>
    <value>3000</value>
</property>

<property>
    <name>ipc.client.connect.max.retries.on.timeouts</name>
    <value>3</value>
</property>
</configuration>
```

17.12.5 任务运行失败，报错 `Err:dial unix: missing address` 怎么办?

修改 `dag-ctrl/etc/dag-ctrl-api.yaml` 配置文件，配置 SSH 的 `UserName` 。

17.12.6 任务运行失败，报错 `bash: /home/xxx/yueshu-analytics/scripts/run_algo.sh: No such file or directory` 怎么办?

修改 `dag-ctrl/etc/tasks.yaml` 配置文件，配置算法执行路径 `exec_file` 。

17.12.7 任务运行失败，报错 `/lib64/libm.so.6: version 'GLIBC_2.29' not found (required by /home/vesoft/jdk-18.0.1/jre/lib/amd64/server/libjvm.so)` 怎么办?

由于 JDK18 版本太新，而操作系统版本太旧， `YUM` 无法下载 `GLIBC_2.29` ，可以安装 JDK1.8，请同步修改 `yueshu-analytics/scripts/set_env.sh` 中的 JDK 地址。

17.12.8 任务运行失败，报错 `handshake failed: ssh: unable to authenticate, attempted methods [none publickey], no supported methods remain` 怎么办?

重新配置 `.ssh` 文件夹及 `.ssh/authorized_keys` 文件的权限， `.ssh` 文件夹权限为 `744` ， `.ssh/authorized_keys` 文件权限为 `600` 。

Copyright © Yueshu Graph Technology - All rights reserved.

17.12.9 任务运行失败，报错 `There are 0 Yueshu Analytics available. clusterSize should be less than or equal to it` 怎么办?

按如下流程排查:

1. 检查节点间 SSH 免密登录配置是否配置成功。可以在 Dag Controller 机器上执行 `ssh <user_name>@<node_ip>` 命令看能否成功登录。

> **Note**
>
> Dag Controller 和悦数图计算在同一台机器时，也需要配置免密登录。

2. 检查 Dag Controller 的配置文件。
- 检查 `etc/dag-ctrl-api.yaml` 中的 SSH 用户和启动 Dag Controller 服务的用户、配置 SSH 免密登录的用户是不是一致。
- 检查 `etc/tasks.yaml` 中的算法路径是否正确。
- 检查 `scripts/set_env.sh` 中的 Hadoop 和 Java 的路径是否正确。
3. 修改上述配置后需要重启 Dag Controller 使配置生效。

17.12.10 任务运行失败，报错 `no available namenodes: dial tcp xx.xx.xx.xx:8020: connect: connection timed out` 怎么办?

请检查 HDFS 的 namenode 端口 8020 是否开放。

17.12.11 任务运行失败，报错 `org.apache.hadoop.net.ConnectTimeoutException: 60000 millis timeout` 怎么办?

请检查 HDFS 的 datanode 端口 50010 是否开放。

如果没有开放端口，还可能报类似如下错误:

- `Check failed: false close hdfs-file failed`
- `org.apache.hadoop.ipc.RemoteException(java.io.IOException): File /analytics/xx/tasks/analytics_xxx/xxx.csv could only be replicated to 0 nodes instead of minReplication`

17.12.12 任务运行失败，报错 `broadcast.hpp:193] Check failed: (size_t)recv_bytes >= sizeof(chunk_tail_t) recv message too small: 0` 怎么办?

任务要处理的数据量过小，但是配置的计算节点数与进程数太多。需要在提交作业时设置较小的 `clusterSize` 和 `processes` 。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

18. 图计算

18.1 算法简介

图计算可以检测图结构，例如图中社区的检测、图的划分等，也可以揭示各个点之间关联关系的内在特征，例如点的中心性、相似性等。本文介绍相关算法和参数。

> **Note**
>
> 本文仅介绍悦数图计算的参数，NebulaGraph Algorithm 的参数请先参见对应的算法文件。

> **Note**
>
> 执行图计算时不仅需要设置算法的参数，对数据源也有要求。数据源需要包含起点和终点。PageRank、DegreeWithTime、SSSP、APSP、LPA、HANP、Louvain 算法还需要包含权重（weight）。

- 如果数据源来自 HDFS，需要指定 CSV 文件，包含 `src` 和 `dst` 列，部分算法还需要包含 `weight` 列。
- 如果数据源来自悦数图数据库，需要指定边类型，该类型的边提供 `src` 和 `dst` 列，部分算法还需要指定边类型的某个属性作为 `weight` 列。

18.1.1 节点重要度算法

PageRank

PageRank（页面排序）算法根据点之间的关系（边）计算点的相关性和重要性，通常使用在搜索引擎页面排名中。如果一个网页被很多其他网页链接，说明这个网页比较重要（PageRank 值较高）；如果一个 PageRank 值很高的网页链接到其他网页，那么被链接到的网页的 PageRank 值会提高。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`ITERATIONS`	`10`	最大迭代次数。
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`EPS`	`0.0001`	收敛精度，两轮迭代的结果差值小于这个值，结束迭代。
`DAMPING`	`0.85`	阻尼系数，访问页面后的跳转概率。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`VALUE`	double	点的 PageRank 值。

KCore

KCore 算法用于计算出没有小于 K 度的点组成的子图，通常使用在社区发现、金融风控等场景。其计算结果是判断点重要性最常用的参考值之一，体现了点的传播能力。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`TYPE`	`vertex`	计算类型。取值：`vertex`、`subgraph`。`vertex` 表示为每个点计算核心度，`subgraph` 表示计算邻居。
`KMIN`	`1`	范围计算时设置 K 的最小值。仅在 `TYPE = subgraph` 时生效。
`KMAX`	`1000000`	范围计算时设置 K 的最大值。仅在 `TYPE = subgraph` 时生效。

- `TYPE=vertex` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`VALUE`	int	输出点的核心度。

- `TYPE=subgraph` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`VALUE`	与 `VID` 类型相同	输出点的邻居。

DegreeCentrality（NStepDegree）

DegreeCentrality（度中心性） 算法用于查找图中的流行点。度中心性测量来自点的传入或传出（或两者）关系的数量，具体取决于关系投影的方向。一个点的度越大就意味着这个点的度中心性越高，该点在网络中就越重要。

> **Note**
>
> 悦数图计算仅粗略估算度中心性。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`STEP`	`3`	计算度数。`-1` 表示无穷大。
`BITS`	`6`	用于基数估计的 hyperloglog 位宽。
`TYPE`	`both`	计算的边的方向。取值：`in` 、`out` 、`both` 。

- `TYPE=both` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`BOTH_DEGREE`	int	输出点的双向度中心性。
`OUT_DEGREE`	int	输出点的出方向度中心性。
`IN_DEGREE`	int	输出点的入方向度中心性。

- `TYPE=out` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`OUT_DEGREE`	int	输出点的出方向度中心性。

- `TYPE=in` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`IN_DEGREE`	int	输出点的入方向度中心性。

DegreeWithTime

DegreeWithTime 算法是基于边的时间范围统计邻居，查找出图中的流行点。

> **Note**
>
> 仅悦数图计算支持该算法。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 传入参数

参数	默认值	说明
`TYPE`	`both`	计算的边的方向。取值：`in`、`out`、`both`。
`BEGIN_TIME`	-	起始时间。格式为`yyyy-MM-dd HH:mm:ss.SSS`。
`END_TIME`	-	结束时间。格式为`yyyy-MM-dd HH:mm:ss.SSS`。

- `TYPE=both`时的输出参数

参数	类型	说明
`VID`	创建图空间时`vid_type`决定	点 ID。
`BOTH_DEGREE`	int	输出点的双向流行度。
`OUT_DEGREE`	int	输出点的出方向流行度。
`IN_DEGREE`	int	输出点的入方向流行度。

- `TYPE=out`时的输出参数

参数	类型	说明
`VID`	创建图空间时`vid_type`决定	点 ID。
`OUT_DEGREE`	int	输出点的出方向流行度。

- `TYPE=in`时的输出参数

参数	类型	说明
`VID`	创建图空间时`vid_type`决定	点 ID。
`IN_DEGREE`	int	输出点的入方向流行度。

BetweennessCentrality

BetweennessCentrality（介数中心性）算法是一种检测点对图中信息流的影响量的方法，用于查找从图的一部分到另一部分时作为桥梁的点。每个点都会根据通过该点的最短路径的数量获得一个分数，即介数中心性分数。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`ITERATIONS`	`10`	最大迭代次数。
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`CHOSEN`	`-1`	选取的点ID，`-1` 表示随机选。
`CONSTANT`	`2`	系数。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`VALUE`	double	点的介数中心性分数。

ClosenessCentrality

ClosenessCentrality（紧密中心性）算法用于计算一个点到所有其他可达点的最短距离的平均值的倒数。值越大，点在图中的位置越靠近中心，也可以用来衡量信息从该点传输到其他点的时间长短。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`NUM_SAMPLES`	`10`	采样的点数量。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`VALUE`	double	点的紧密中心性分数。

18.1.2 路径算法

APSP

APSP（全图最短路径）算法用于寻找图中两点之间的所有最短路径。

> **Note**
>
> 仅悦数图计算支持该算法。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 输出参数

参数	类型	说明
`VID1`	创建图空间时 `vid_type` 决定	起点的 ID。
`VID2`	创建图空间时 `vid_type` 决定	终点的 ID。
`DISTANCE`	double	输出 `VID1` 到 `VID2` 的距离。

SSSP

SSSP（单源最短路径）算法用于计算给定的一个点（起始点）出发到其余各点的最短路径长度。通常使用在网络路由、路径设计等场景。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`ROOT`	-	起始点的 VID。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点 ID。
`DISTANCE`	double	输出 `ROOT` 到 `VID` 的距离。

BFS

BFS（广度优先遍历）算法是一种基础的图遍历算法，它给定一个起始点，以递增的跳数访问其他点，即先遍历点的所有相邻点，再往相邻点的相邻点延伸。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`ROOT`	-	起始点的 VID。

- 输出参数

参数	类型	说明
`ROOT`	创建图空间时 `vid_type` 决定	起始点的 ID。
`VISITED`	int	输出 `ROOT` 访问过的点数量。

Copyright © Yueshu Graph Technology - All rights reserved.

ShortestPath

ShortestPath（最短路径）算法用于寻找图中起点和终点任意两两之间的最短路径，适用于路径设计、网络规划等场景。

- 悦数图计算
- 传入参数

参数	默认值	说明
`src`	`"100"`	起点。多个值用逗号分隔。
`dst`	`"200"`	终点。多个值用逗号分隔。

- 输出参数

参数	类型	说明
`VALUE`	list	返回最短路径中的点。格式为 `src, vid1,vid2...dst` 。同时有多个最短路径时，只返回一条。

18.1.3 社区发现算法

LPA

LPA（标签传播）算法是一种基于图的半监督学习方法，其基本思路是用已标记点的标签信息去预测未标记点的标签信息。利用样本间的关系建图，点包括已标注和未标注数据，其边表示两个点的相似度，点的标签按相似度传递给其他点。标签数据就像是一个源头，可以对无标签数据进行标注，点的相似度越大，标签越容易传播。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`ITERATIONS`	`10`	最大迭代次数。
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false` ，系统会自动添加反向边。
`IS_CALC_MODULARITY`	`false`	是否计算模块度。
`IS_OUTPUT_MODULARITY`	`false`	是否计算并输出模块度。设置为 `true` 时，默认输出到文件的第三列，也可以通过选项 `--nebula_output_props` 和 `--nebula_output_types` 输出到悦数图数据库中。使用悦数图探索时暂不支持输出到悦数图数据库中。
`IS_STAT_COMMUNITY`	`false`	是否统计社区的数量。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`LABEL`	与 `VID` 类型相同	输出标签相同的点的 ID。

HANP

HANP（Hop Attenuation & Node Preference）算法是LPA算法的优化算法，考虑了标签的其他信息，例如度的信息、距离信息等，同时在传播时引入了衰减系数，防止过渡传播。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`ITERATIONS`	`10`	最大迭代次数。
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`PREFERENCE`	`1.0`	对邻居节点度的偏向性。`m>0` 表示偏向节点度高的邻居，`m<0` 表示偏向节点度低的邻居，`m=0` 表示不考虑邻居节点度。
`HOP_ATT`	`0.1`	衰减因子。取值范围 `0` ～ `1`。值越大衰减的越快，可以传递的次数越少。
`IS_OUTPUT_MODULARITY`	`false`	是否计算并输出模块度。设置为 `true` 时，默认输出到文件的第三列，也可以通过选项 `--nebula_output_props` 和 `--nebula_output_types` 输出到悦数图数据库中。使用悦数图探索时暂不支持输出到悦数图数据库中。
`IS_STAT_COMMUNITY`	`false`	是否统计社区的数量。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`LABEL`	与 `VID` 类型相同	输出标签相同的点的 ID。

ConnectedComponent

ConnectedComponent（联通分量）算法用于计算出图中的一个子图，当中所有节点都相互连接。考虑路径方向的为强联通分量（strongly connected component），不考虑路径方向的为弱联通分量（weakly connected component）。

> **Note**
>
> 悦数图计算仅支持弱联通分量。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`IS_CALC_MODULARITY`	`false`	是否计算模块度。
`IS_OUTPUT_MODULARITY`	`false`	是否计算并输出模块度。设置为 `true` 时，默认输出到文件的第三列，也可以通过选项 `--nebula_output_props` 和 `--nebula_output_types` 输出到悦数图数据库中。使用悦数图探索时暂不支持输出到悦数图数据库中。
`IS_STAT_COMMUNITY`	`false`	是否统计社区的数量。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`LABEL`	与 `VID` 类型相同	输出标签相同的点的 ID。

Louvain

Louvain 算法是基于模块度的社区发现算法，该算法在效率和效果上都表现较好，并且能够发现层次性的社区结构，其优化目标是最大化整个社区网络的模块度。模块度用于区分社区内和社区间链路密度的差异，是衡量每个点划分社区的好坏。通常情况下，一个优秀的分群方法将会使得社区内部的模块度高于社区与社区之间。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`IS_DIRECTED`	`true`	是否考虑边的方向。如果设置为 `false`，系统会自动添加反向边。
`OUTER_ITERATION`	`20`	第一阶段最大迭代次数。
`INNER_ITERATION`	`10`	第二阶段最大迭代次数。
`IS_CALC_MODULARITY`	`false`	是否计算模块度。
`IS_OUTPUT_MODULARITY`	`false`	是否计算并输出模块度。设置为 `true` 时，默认输出到文件的第三列，也可以通过选项 `--nebula_output_props` 和 `--nebula_output_types` 输出到悦数图数据库中。使用悦数图探索时暂不支持输出到悦数图数据库中。
`IS_STAT_COMMUNITY`	`false`	是否统计社区的数量。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`LABEL`	与 `VID` 类型相同	输出标签相同的点的 ID。

Copyright © Yueshu Graph Technology - All rights reserved.

InfoMap

InfoMap 算法使用双层编码方式将有向图进行社区分类。不同社区内部节点的编码复用，可以大幅缩短描述的信息长度。在实现方式上，该算法包含了 PageRank 算法，用于将随机游走转变为随机冲浪。

> **Note**
>
> 仅悦数图计算支持该算法。

- 悦数图计算
- 传入参数

参数	默认值	说明
`pagerank_iter`	`10`	内部 PageRank 算法的最大迭代次数。
`pagerank_threshold`	`0.0001`	内部 PageRank 算法的收敛精度。
`teleport_prob`	`0.15`	穿越概率。
`inner_iter`	`3`	内层迭代次数。
`outer_iter`	`2`	外层迭代次数。
`comm_info_num`	`100`	输出的社群数量。

- 输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`LABEL`	与 `VID` 类型相同	输出标签相同的点的 ID。

18.1.4 图特征算法

TriangleCount

TriangleCount（三角计数）算法用于统计图中三角形个数。三角形越多，代表图中节点关联程度越高，组织关系越严密。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`OPT`	`3`	计算类型。取值：`1`（统计整个图）、`2`（通过每个点统计）、`3`（列出所有三角形）。
`REMOVED_DUPLICATION_EDGE`	`true`	是否排除重复边。
`REMOVED_SELF_EDGE`	`true`	是否排除自环边。

- `OPT=1` 时的输出参数

参数	类型	说明
`COUNT`	int	输出全图的三角形数量。

- `OPT=2` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	输出每个点的 ID。
`COUNT`	int	输出每个点的三角形数量。

- `OPT=3` 时的输出参数

参数	类型	说明
`VID1`	与 `VID` 类型相同	输出构成三角形的点 A 的 ID。
`VID2`	与 `VID` 类型相同	输出构成三角形的点 B 的 ID。
`VID3`	与 `VID` 类型相同	输出构成三角形的点 C 的 ID。

Node2Vec

Node2Vec算法在 DeepWalk 的基础上提出了更加合理的图特征学习方法，提出了用于网络中可伸缩特征学习的半监督算法，使用 SGD 优化一个自定义的基于图的目标函数，该方法可以最大化的在 D 维特征空间保留节点的网络领域信息；在随机游走的基础上设计了一种二阶随机游走的过程，相当于对 DeepWalk 算法的一种扩展，它保留了邻居节点的图特征。适用于节点功能相似性比较、节点结构相似性比较、社团聚类等场景。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`is_weighted`	`false`	随机游走是否有偏差。
`p`	`1.0`	随机游走的后向偏差。
`q`	`0.5`	随机游走的前向偏差。
`epoch`	`1`	迭代次数。
`step`	`10`	每次迭代的步数。
`rate`	`0.02`	随机游走的比例。

- 输出参数

  输出多个列，同一列中的点是有关联的。

Tree_stat

Tree_stat 算法用于统计图空间中指定根节点的子图的宽度或深度。

> **Note**
>
> 仅悦数图计算支持该算法。

- 悦数图计算
- 传入参数

参数	默认值	说明
`root`	`100`	根节点的 VID。
`stat`	`width,depth`	统计宽度或深度。多个值用逗号分隔。

- 输出参数

参数	类型	说明
`VALUE`	list	返回一行统计数据，格式与参数 `stat` 一致。

HyperANF

HyperANF 算法用于评估图中任意两点的平均距离。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

仅悦数图计算支持该算法。

- 悦数图计算
- 传入参数

参数	默认值	说明
`bits`	`6`	HyperLogLog 计数器的 bit 位长度，取值范围：6~16。

- 输出参数

参数	类型	说明
`VALUE`	double	平均距离。

18.1.5 聚类算法

ClusteringCoefficient

ClusteringCoefficient（聚集系数）算法用于计算图中节点的聚集程度。在各类反映真实世界的网络结构，特别是社交网络结构中，各个点之间倾向于形成密度相对较高的网络群，也就是说，相对于在两个点之间随机连接而得到的网络，真实世界网络的聚集系数更高。

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`TYPE`	`local`	聚集类型。取值：`local`（为每个点计算聚集系数）、`global`（为全图计算聚集系数）。
`REMOVED_DUPLICATION_EDGE`	`true`	是否排除重复边。
`REMOVED_SELF_EDGE`	`true`	是否排除自环边。

- `TYPE=local` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`VALUE`	double	输出每个点的聚集系数。

- `TYPE=global` 时的输出参数

参数	类型	说明
`VID`	创建图空间时 `vid_type` 决定	点的 ID。
`VALUE`	double	输出全图的聚集系数。只有一行数据。

18.1.6 相似度算法

Jaccard

Jaccard（杰卡德相似度）算法用于计算两个点（或集合）的相似程度，预测他们之间的关系。适用于社交网上的好友推荐、关系预测等场景。

Copyright © Yueshu Graph Technology - All rights reserved.

参数说明如下。

- 悦数图计算
- 传入参数

参数	默认值	说明
`IDS1`	-	若干个 VID 构成的集合A。多个 VID 之间用英文逗号（,）隔开。不可为空。
`IDS2`	-	若干个 VID 构成的集合B。多个 VID 之间用英文逗号（,）隔开。可以为空，为空时表示所有点。
`REMOVED_SELF_EDGE`	`true`	是否排除自环边。

- 输出参数

参数	类型	说明
`VID1`	创建图空间时 `vid_type` 决定	第一个点的 ID。
`VID2`	创建图空间时 `vid_type` 决定	第二个点的 ID。
`VALUE`	double	`VID1` 和 `VID2` 的相似度。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

18.2 NebulaGraph Algorithm

NebulaGraph Algorithm （简称 Algorithm）是悦数图数据库的图算法组件，一款基于 GraphX 的 Spark 应用程序，通过提交 Spark 任务的形式使用完整的算法工具对悦数图数据库数据库中的数据执行图计算，也可以通过编程形式调用 lib 库下的算法针对 DataFrame 执行图计算。

18.2.1 版本兼容性

NebulaGraph Algorithm 版本和悦数图数据库内核的版本对应关系如下。

悦数图数据库版本	**NebulaGraph Algorithm** 版本
nightly	3.0-SNAPSHOT
3.0.0 ~ 3.5.x	3.x.0
2.6.x	2.6.x
2.5.0、2.5.1	2.5.0
2.0.0、2.0.1	2.1.0

18.2.2 前提条件

在使用 Algorithm 之前，用户需要确认以下信息：

- 悦数图数据库服务已经部署并启动。详细信息，参考 悦数图数据库安装部署。
- Spark 版本为 2.4.x。
- Scala 版本为 2.11。
- （可选）如果用户需要在 Github 中克隆最新的 Algorithm，并自行编译打包，可以选择安装 Maven。

18.2.3 使用限制

图计算会输出点的数据集，算法结果会以DataFrame形式作为点的属性存储。用户可以根据业务需求，自行对算法结果做进一步操作，例如统计、筛选。

> **Compatibility**
>
> Algorithm v3.1.0 版本之前，直接提交算法包时，点 ID 的数据必须为整数，即点 ID 可以是 INT 类型，或者是 String 类型但数据本身为整数。

Copyright © Yueshu Graph Technology - All rights reserved.

18.2.4 支持算法

NebulaGraph Algorithm 支持的图计算算法如下。

算法名	说明	应用场景	属性名称	属性数据类型
PageRank	页面排序	网页排序、重点节点挖掘	pagerank	double/string
Louvain	鲁汶	社团挖掘、层次化聚类	louvain	int/string
KCore	K 核	社区发现、金融风控	kcore	int/string
LabelPropagation	标签传播	资讯传播、广告推荐、社区发现	lpa	int/string
Hanp	标签传播进阶版	社区发现、推荐	hanp	int/string
ConnectedComponent	弱联通分量	社区发现、孤岛发现	cc	int/string
StronglyConnectedComponent	强联通分量	社区发现	scc	int/string
ShortestPath	最短路径	路径规划、网络规划	shortestpath	string
TriangleCount	三角形计数	网络结构分析	trianglecount	int/string
GraphTriangleCount	全图三角形计数	网络结构及紧密程度分析	count	int
BetweennessCentrality	中介中心性	关键节点挖掘，节点影响力计算	betweenness	double/string
ClosenessCentrality	紧密中心性	关键节点挖掘、节点影响力计算	closeness	double/string
DegreeStatic	度统计	图结构分析	degree,inDegree,outDegree	int/string
ClusteringCoefficient	聚集系数	推荐、电信诈骗分析	clustercoefficient	double/string
Jaccard	杰卡德相似度计算	相似度计算、推荐	jaccard	string
BFS	广度优先遍历	层序遍历、最短路径规划	bfs	string
DFS	深度优先遍历	层序遍历、最短路径规划	dfs	string
Node2Vec	-	图分类	node2vec	string

Note

如果需要将算法结果写入到悦数图数据库中，请确保对应图空间中的 Tag 有和上表对应的属性名称和数据类型。

Copyright © Yueshu Graph Technology - All rights reserved.

18.2.5 实现方法

NebulaGraph Algorithm 实现图计算的流程如下：

1. 利用 NebulaGraph Spark Connector 从悦数图数据库中读取图数据为 DataFrame。
2. 将 DataFrame 转换为 GraphX 的图。
3. 调用 GraphX 提供的图算法（例如 PageRank）或者自行实现的算法（例如 Louvain 社区发现）。

详细的实现方法可以参见相关 Scala 文件。

18.2.6 获取 NebulaGraph Algorithm

编译打包

1. 克隆仓库 `nebula-algorithm`。

```
$ git clone -b v3.0.0 https://github.com/vesoft-inc/nebula-algorithm.git
```

2. 进入目录 `nebula-algorithm`。

```
$ cd nebula-algorithm
```

3. 编译打包。

```
$ mvn clean package -Dgpg.skip -Dmaven.javadoc.skip=true -Dmaven.test.skip=true
```

编译完成后，在目录 `nebula-algorithm/target` 下生成类似文件 `nebula-algorithm-3.x.x.jar`。

Maven 远程仓库下载

下载地址

18.2.7 使用方法

调用算法接口（推荐）

`lib` 库中提供了 10 种常用图计算算法，用户可以通过编程调用的形式调用算法。

1. 在文件 `pom.xml` 中添加依赖。

```
<dependency>
    <groupId>com.vesoft</groupId>
    <artifactId>nebula-algorithm</artifactId>
    <version>3.0.0</version>
</dependency>
```

2. 传入参数调用算法（以 PageRank 为例）。更多调用示例请参见示例。

> **Note**
>
> 执行算法的 DataFrame 默认第一列是起始点，第二列是目的点，第三列是边权重（非悦数图数据库中的 Rank）。

```
val prConfig = new PRConfig(5, 1.0)
val prResult = PageRankAlgo.apply(spark, data, prConfig, false)
```

如果用户的节点 ID 是 String 类型，可以参考 PageRank 的示例。示例中进行了 ID 转换，将 String 类型编码为 Long 类型， 并在算法结果中将 Long 类型 ID 解码为原始的 String 类型。

Copyright © Yueshu Graph Technology - All rights reserved.

直接提交算法包

1. 设置配置文件。

```
{
    # Spark 相关配置
    spark: {
    app: {
        name: LPA
        # Spark 分片数量
        partitionNum:100
    }
    master:local
    }

    data: {
    # 数据源，可选值为 nebula、csv、json。
    source: nebula
    # 数据落库，即图计算的结果写入的目标，可选值为 nebula、csv、json。
    sink: nebula
    # 算法是否需要权重。
    hasWeight: false
    }

    # 悦数图数据库相关配置
    nebula: {
    # 数据源。悦数图数据库作为图计算的数据源时，nebula.read 的配置才生效。
    read: {
        # 所有 Meta 服务的 IP 地址和端口，多个地址用英文逗号（,）分隔。格式："ip1:port1,ip2:port2"。
        # 使用 docker-compose 部署，端口需要填写 docker-compose 映射到外部的端口
        # 可以用`docker-compose ps`查看
        metaAddress: "192.168.*.10:9559"
        # 悦数图数据库图空间名称
        space: basketballplayer
        # 悦数图数据库 Edge type，多个 labels 时，多个边的数据将合并。
        labels: ["serve"]
        # 悦数图数据库 每个 Edge type 的属性名称，此属性将作为算法的权重列，请确保和 Edge type 对应。
        weightCols: ["start_year"]
    }

    # 数据落库。图计算结果落库到悦数图数据库时，nebula.write 的配置才生效。
    write:{
        # Graph 服务的 IP 地址和端口，多个地址用英文逗号（,）分隔。格式："ip1:port1,ip2:port2"。
        # 使用 docker-compose 部署，端口需要填写 docker-compose 映射到外部的端口
        # 可以用`docker-compose ps`查看
        graphAddress: "192.168.*.11:9669"
        # 所有 Meta 服务的 IP 地址和端口，多个地址用英文逗号（,）分隔。格式："ip1:port1,ip2:port2"。
        # 使用 docker-compose 部署，端口需要填写 docker-compose 映射到外部的端口
        # 可以用`docker-compose ps`查看
        metaAddress: "192.168.*.12:9559"
        user:root
        pswd:nebula
        # 在提交图计算任务之前需要自行创建图空间及 Tag
        # 悦数图数据库图空间名称
        space:nb
        # 悦数图数据库 Tag 名称，图计算结果会写入该 Tag。Tag 中的属性名称固定如下：
        # PageRank：pagerank
        # Louvain：louvain
        # ConnectedComponent：cc
        # StronglyConnectedComponent：scc
        # LabelPropagation：lpa
        # ShortestPath：shortestpath
        # DegreeStatic：degree、inDegree、outDegree
        # KCore：kcore
        # TriangleCount：tranglecpunt
        # BetweennessCentrality：betweennedss
        tag:pagerank
    }
    }

    local: {
    # 数据源。图计算的数据源为 csv 文件或 json 文件时，local.read 的配置才生效。
    read:{
        filePath: "hdfs://127.0.0.1:9000/edge/work_for.csv"
        # 如果 CSV 文件没有表头，使用 [_c0, _c1, _c2, ..., _cn] 表示其表头，有表头或者是 json 文件时，直接使用表头名称即可。
        # 起始点 ID 列的表头。
        srcId:"_c0"
        # 目的点 ID 列的表头。
        dstId:"_c1"
        # 权重列的表头
        weight: "_c2"
        # csv 文件是否有表头
        header: false
        # csv 文件的分隔符
        delimiter:","
    }

    # 数据落库。图计算结果落库到 csv 文件或 text 文件时，local.write 的配置才生效。
    write:{
        resultPath:/tmp/
    }
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
    algorithm: {
    # 需要执行的算法，可选值为：
    # pagerank、louvain、connectedcomponent、labelpropagation、shortestpaths、
    # degreestatic、kcore、stronglyconnectedcomponent、trianglecount、
    # betweenness、graphtriangleCount。
    executeAlgo: pagerank

    # PageRank 参数
    pagerank: {
        maxIter: 10
        resetProb: 0.15
        encodeId:false # 如果点 ID 是字符串类型，请配置为 true。
    }

    # Louvain 参数
    louvain: {
        maxIter: 20
        internalIter: 10
        tol: 0.5
        encodeId:false # 如果点 ID 是字符串类型，请配置为 true。
    }

    # ...
    }
    }
}
```

Note

当配置为 `sink: nebula` 的时候，意味着算法运算结果将被写回悦数图数据库集群，这对写回到的 TAG 中的属性名有隐含的约定。详情参考本文支持算法部分。

2. 提交图计算任务。

```
${SPARK_HOME}/bin/spark-submit --master <mode> --class com.vesoft.nebula.algorithm.Main <nebula-algorithm-3.0.0.jar_path> -p <application.conf_path>
```

示例：

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.algorithm.Main /root/nebula-algorithm/target/nebula-algorithm-3.0-SNAPSHOT.jar -p /root/nebula-algorithm/src/main/
resources/application.conf
```

18.2.8 视频

- 图计算工具——NebulaGraph Algorithm 介绍（2 分 36 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

18.3 悦数图计算

悦数图计算是一款高性能图计算框架工具，支持对悦数图数据库数据库中的数据执行图分析。

18.3.1 前提条件

- 联系我们获取悦数图计算安装包。
- 已加载 License Key。
- 已部署 2.2.x 或以上版本的 HDFS。
- 已安装 1.8 版本的 JDK。

18.3.2 适用场景

支持将数据源为悦数图数据库集群、HDFS 上的 CSV 文件或本地 CSV 文件中的数据导入 悦数图计算，并将图计算结果输出至悦数图数据库集群、HDFS 上的 CSV 文件或本地 CSV 文件。

18.3.3 使用限制

输入和输出均为悦数图数据库集群时，图计算结果只能输出到数据源所在的图空间。

18.3.4 版本兼容性

悦数图计算版本和悦数图数据库内核的版本对应关系如下。

悦数图数据库版本	悦数图计算版本
3.5.x	3.5.0
3.4.0 ~ 3.4.1	3.5.0、3.4.0
3.3.0	3.3.0
3.1.0 ~ 3.2.x	3.2.0
3.0.x	1.0.x
2.6.x	0.9.0

Copyright © Yueshu Graph Technology - All rights reserved.

18.3.5 支持算法

悦数图计算支持的图计算算法如下。

算法名	说明	分类
APSP	全图最短路径	路径
SSSP	单源最短路径	路径
BFS	广度优先遍历	路径
ShortestPath	最短路径	路径
PageRank	页面排序	节点重要度
KCore	K核	节点重要度
DegreeCentrality	度中心性	节点重要度
DegreeWithTime	基于边的时间范围统计邻居	节点重要度
BetweennessCentrality	中介中心性	节点重要度
ClosenessCentrality	紧密中心性	节点重要度
TriangleCount	三角计数	图特征
Node2Vec	图神经网络	图特征
Tree_stat	树结构统计	图特征
HyperANF	图平均距离估算	图特征
LPA	标签传播	社区发现
HANP	标签传播进阶版	社区发现
WCC	弱联通分量	社区发现
LOUVAIN	鲁汶	社区发现
InfoMap	社区分类	社区发现
Clustering Coefficient	聚集系数	聚类
Jaccard	杰卡德相似度	相似度

18.3.6 安装 悦数图计算

1. 安装 悦数图计算。在多个机器安装由多个悦数图计算服务构成的集群时，需要安装路径相同，并设置节点间 SSH 免密登录。

```
$ sudo rpm -ivh <analytics_package_name> --prefix <install_path>
$ sudo chown <user>:<user> -R <install path>
```

例如：

```
$ sudo rpm -ivh nebula-analytics-3.5.0-centos.x86_64.rpm --prefix=/home/vesoft/nebula-analytics
$ sudo chown vesoft:vesoft -R /home/vesoft/nebula-analytics
```

2. 配置 `set_env.sh` 文件，路径为 `nebula-analytics/scripts/set_env.sh` 。配置正确的 Hadoop 路径和 JDK 路径。如果有多台机器，请确保路径一致。

> **Note**
>
> MPICH 进程管理器和 MPICH 库使用的 TCP 端口范围默认为 10000 到 10100。如需调整，修改 `set_env.sh` 文件中环境变量 `MPIR_CVAR_CH3_PORT_RANGE` 的值。

Copyright © Yueshu Graph Technology - All rights reserved.

```
export HADOOP_HOME=<hadoop_path>
export JAVA_HOME=<java_path>
```

3. 配置 `analytics.conf` 文件，路径为 `nebula-analytics/scripts/analytics.conf` 。设置 `license_manager_url` 的值为许可证管理工具所在的主机 IP 和端口号 `9119` ，例如 `192.168.8.100:9119` 。

Copyright © Yueshu Graph Technology - All rights reserved.

18.3.7 使用方法

安装完成后，用户可以设置不同算法的参数，然后执行脚本，即可获得算法的结果，并导出为指定格式。

Copyright © Yueshu Graph Technology - All rights reserved.

1. 选择悦数图计算集群的任一节点，进入目录 `scripts` 。

```
$ cd scripts
```


Copyright © Yueshu Graph Technology - All rights reserved.

2. 确认数据源和输出路径。配置方法如下：

- 数据源为悦数图数据库集群

a. 修改配置文件 `nebula.conf`，设置悦数图数据库集群相关信息。

```
# 连接悦数图数据库时的重试次数。
--retry=3
# 要读取或写入的图空间名称。
--space=baskeyballplayer

# 读取悦数图数据库设置
# 要读取的边的名称。
--edges=LIKES
# 要读取的作为边的权重属性的名称。可以是属性名，也可以是 _rank。
#--edge_data_fields
# 每次扫描读取的行数。
--read_batch_size=10000

# 写回悦数图数据库设置
# 悦数图数据库 的 graphd 服务地址。
--graph_server_addrs=192.168.8.100:9669
# 悦数图数据库 的登录用户名。
--user=root
# 悦数图数据库 的登录密码。
--password=nebula
# 写回悦数图数据库时采用的模式: insert 和 update。
--mode=insert
# 写回到悦数图数据库的 Tag 名称。
--tag=pagerank
# 写回到悦数图数据库的 Tag 对应的属性名称。
--prop=pr
# 写回到悦数图数据库的 Tag 对应的属性的类型。
--type=double
# 写回时，每次写入的行数。
--write_batch_size=1000
# 写回失败的数据所存储的文件。
--err_file=/home/xxx/analytics/err.txt

# 其他设置
# 服务访问超时时间
--graphd_timeout=60000
--metad_timeout=60000
--storaged_timeout=60000
```

b. 修改需要使用的算法脚本，例如 `run_pagerank.sh`，设置相关参数。

```
# 集群所有机器所运行的进程数之和，推荐每台机器为 1 或者 NUMA 架构的 node 数。
WNUM=3
# 每个进程的线程数，推荐最大设置为机器的硬件线程数。
WCORES=4
# 数据源路径
# 可以通过文件 nebula.conf 设置从悦数图数据库读取：
INPUT=${INPUT:="nebula:$PROJECT/scripts/nebula.conf"}
# 也可以通过本地或 HDFS 上的 CSV 文件读取：
# #INPUT=${INPUT:="$PROJECT/data/graph/v100_e2150_ua_c3.csv"}

# 图计算结果输出路径
# 可以输出至悦数图数据库集群，如果数据源也为悦数图数据库，结果会输出至 nebula.conf 指定的图空间。
OUTPUT=${OUTPUT:="nebula:$PROJECT/scripts/nebula.conf"}
# 也可以输出至本地或 HDFS 上的 CSV 文件：
# OUTPUT=${OUTPUT:='hdfs://192.168.8.100:9000/_test/output'}

# true 为有向图，false 为无向图。
IS_DIRECTED=${IS_DIRECTED:=true}
# 是否进行 ID 编码
NEED_ENCODE=${NEED_ENCODE:=true}
# 数据源的点 ID 类型，例如：string、int32、int64。
VTYPE=${VTYPE:=int32}
# 编码类型。distributed 为分布式点ID编码，single 为单机点 ID 编码。
ENCODER=${ENCODER:="distributed"}
# PageRank 算法的参数。不同算法的参数不同。
EPS=${EPS:=0.0001}
DAMPING=${DAMPING:=0.85}
# 迭代次数
ITERATIONS=${ITERATIONS:=100}
```


Copyright © Yueshu Graph Technology - All rights reserved.

- 数据源为本地或 HDFS 上的 CSV 文件

  修改需要使用的算法脚本，例如 `run_pagerank.sh`，设置相关参数。

```
# 集群所有机器所运行的进程数之和，推荐每台机器为 1 或者 NUMA 架构的 node 数。
WNUM=3
# 每个进程的线程数，推荐最大设置为机器的硬件线程数。
WCORES=4
# 数据源路径
# 可以通过文件 nebula.conf 设置从悦数图数据库读取：
# INPUT=${INPUT:="nebula:$PROJECT/scripts/nebula.conf"}
# 也可以通过本地或 HDFS 上的 CSV 文件读取：
INPUT=${INPUT:="$PROJECT/data/graph/v100_e2150_ua_c3.csv"}

# 图计算结果输出路径
# 可以输出至悦数图数据库集群，如果数据源也为悦数图数据库，结果会输出至 nebula.conf 指定的图空间。
# OUTPUT=${OUTPUT:="nebula:$PROJECT/scripts/nebula.conf"}
# 也可以输出至本地或 HDFS 上的 CSV 文件：
OUTPUT=${OUTPUT:='hdfs://192.168.8.100:9000/_test/output'}

# true 为有向图，false 为无向图。
IS_DIRECTED=${IS_DIRECTED:=true}
# 是否进行 ID 编码
NEED_ENCODE=${NEED_ENCODE:=true}
# 数据源的点 ID 类型，例如：string、int32、int64。
VTYPE=${VTYPE:=int32}
# 编码类型。distributed 为分布式点ID编码，single 为单机点 ID 编码。
ENCODER=${ENCODER:="distributed"}
# PageRank 算法的参数。不同算法的参数不同。
EPS=${EPS:=0.0001}
DAMPING=${DAMPING:=0.85}
# 迭代次数
ITERATIONS=${ITERATIONS:=100}
```

3. 修改配置文件 `cluster`，设置执行算法的悦数图计算集群节点和任务分配权重。

```
# 悦数图计算集群节点 IP 地址:任务分配权重
192.168.8.200:1
192.168.8.201:1
192.168.8.202:1
```

4. 执行算法脚本。例如：

```
./run_pagerank.sh
```

5. 在输出路径查看计算结果。

- 输出至悦数图数据库集群，请根据 `nebula.conf` 的设置查看计算结果。
- 输出至 HDFS 上的 CSV 文件或本地 CSV 文件，请根据图计算脚本内的 `OUTPUT` 设置查看计算结果，计算结果为 `.gz` 格式的压缩文件。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

18.4 悦数图探索 Workflow

悦数图探索提供 Workflow 功能进行可视化图计算。

详情参见工作流简介。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19. 自动化部署运维

19.1 什么是 NebulaGraph Operator

悦数图数据库具备自动部署和运维功能，这一功能是基于开源 NebulaGraph Operator 进行优化和提升，从而实现了对悦数图数据库集群的高效管理和维护。

19.1.1 基本概念

NebulaGraph Operator 是用于在 Kubernetes 系统上自动化部署和运维 NebulaGraph 集群的工具。依托于 Kubernetes 扩展机制，悦数图数据库将其运维领域的知识全面注入至 Kubernetes 系统中，让悦数图数据库成为真正的云原生图数据库。

19.1.2 工作原理

对于 Kubernetes 系统内不存在的资源类型，用户可以通过添加自定义 API 对象的方式注册，常见的方法是使用 CustomResourceDefinition（CRD）。

NebulaGraph Operator 将悦数图数据库集群的部署管理抽象为 CRD。通过结合多个内置的 API 对象，包括 StatefulSet、Service 和 ConfigMap，悦数图数据库集群的日常管理和维护被编码为一个控制循环。在 Kubernetes 系统内，每一种内置资源对象，都运行着一个特定的控制循环，将它的实际状态通过事先规定好的编排动作，逐步调整为最终的期望状态。当一个 CR 实例被提交时，NebulaGraph Operator 会根据控制流程驱动数据库集群进入最终状态。

Copyright © Yueshu Graph Technology - All rights reserved.

19.1.3 功能介绍

NebulaGraph Operator 已具备的功能如下：

- 集群创建和卸载：NebulaGraph Operator 简化了用户部署和卸载集群的过程。用户只需提供对应的 CR 文件，NebulaGraph Operator 即可快速创建或者删除一个对应的悦数图数据库集群。更多信息参考使用 Kubectl 部署悦数图数据库集群或者使用 Helm 部署悦数图数据库集群。
- 集群扩容和缩容：通过在控制循环中调用悦数图数据库原生提供的扩缩容接口，悦数图数据库封装 NebulaGraph Operator 实现了扩缩容的逻辑，用户可以通过 YAML 配置进行简单的扩缩容，且保证数据的稳定性。更多信息参考使用 Kubeclt 扩缩容集群 或使用 Helm 扩缩容集群。
- 备份和恢复：支持备份悦数图数据库集群的数据至兼容 S3 协议的存储服务中并支持恢复数据至集群中。更多信息参考使用 NebulaGraph Operator 备份和恢复数据。
- 集群升级：支持升级 3.5.0 版的悦数图数据库集群至 3.5.x 版。
- 故障自愈：NebulaGraph Operator 调用悦数图数据库集群提供的接口，动态地感知服务状态。一旦发现异常，NebulaGraph Operator 自动进行容错处理。更多信息参考故障自愈。
- 均衡调度：基于调度器扩展接口，NebulaGraph Operator 提供的调度器可以将应用 Pods 均匀地分布在悦数图数据库集群中。

19.1.4 使用限制

版本限制

NebulaGraph Operator 不支持 v1.x 版本的 NebulaGraph，其与悦数图数据库版本的对应关系如下：

悦数图数据库版本	**NebulaGraph Operator** 版本
3.5.x	1.5.0、1.6.x
3.0.0 ~ 3.4.1	1.3.0、1.4.0 ~ 1.4.2
3.0.0 ~ 3.3.x	1.0.0、1.1.0、1.2.0
2.5.x ~ 2.6.x	0.9.0
2.5.x	0.8.0

版本兼容性

- 1.x 版本的 NebulaGraph Operator 不兼容 3.x 以下版本的 NebulaGraph.
- 由于 0.9.0 版本的 NebulaGraph Operator 的日志盘和数据盘分开存储，因此用 0.9.0 版的 NebulaGraph Operator 管理通过 0.8.0 版本创建的 2.5.x 版本的悦数图数据库集群会导致兼容性问题。用户可以备份 2.5.x 版本的悦数图数据库集群，然后使用 0.9.0 版本的 Operator 创建 2.6.x 版本集群。

功能限制

NebulaGraph Operator 的扩缩容功能仅对企业版的悦数图数据库集群开放，不支持使用 NebulaGraph Operator 扩缩容社区版的悦数图数据库集群。

19.1.5 更新说明

Release

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.2 使用流程

使用 NebulaGraph Operator 访问悦数图数据库集群服务的流程如下:

1. 安装 NebulaGraph Operator。
2. 创建悦数图数据库集群。

   具体步骤参考使用 Kubectl 部署悦数图数据库集群或者使用 Helm 部署悦数图数据库集群。
3. 连接悦数图数据库服务。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.3 部署 NebulaGraph Operator

用户可使用 Helm 工具部署 NebulaGraph Operator。

19.3.1 背景信息

NebulaGraph Operator 为用户管理悦数图数据库集群，使用户无需在生产环境中手动安装、扩展、升级和卸载 NebulaGraph，减轻用户管理不同应用版本的负担。

19.3.2 前提条件

安装 NebulaGraph Operator 前，用户需要安装以下软件并确保安装版本的正确性（NebulaGraph Operator 不负责处理安装这些软件过程中出现的问题）。

软件	版本要求
Kubernetes	>= 1.16
Helm	>= 3.2.0
CoreDNS	>= 1.6.0

Note

- 如果使用基于角色的访问控制的策略，用户需开启 RBAC（可选）。
- CoreDNS 是一个灵活的、可扩展的 DNS 服务器，被安装在集群内作为集群内 Pods 的 DNS 服务器。悦数图数据库集群中的每个组件通过 DNS 解析类似 `x.default.svc.cluster.local` 这样的域名相互通信。

19.3.3 操作步骤

安装 **NebulaGraph Operator**

1. 添加 NebulaGraph Operator Helm 仓库。

```
helm repo add nebula-operator https://vesoft-inc.github.io/nebula-operator/charts
```

2. 拉取最新的 Operator Helm 仓库。

```
helm repo update
```

参考 Helm 仓库获取更多 `helm repo` 相关信息。

3. 创建命名空间用于安装 NebulaGraph Operator。

```
kubectl create namespace <namespace_name>
```

例如，创建 `nebula-operator-system` 命名空间。

```
kubectl create namespace nebula-operator-system
```

- nebula-operator chart 中的所有资源都会安装在该命名空间下。
- 用户也可创建其他命名空间。

4. 安装 NebulaGraph Operator。

```
helm install nebula-operator nebula-operator/nebula-operator --namespace=<namespace_name> --version=${chart_version}
```

例如，安装1.6.2版的 Operator 命令如下。

Copyright © Yueshu Graph Technology - All rights reserved.

```
helm install nebula-operator nebula-operator/nebula-operator --namespace=nebula-operator-system --version=1.6.2
```

- `1.6.2` 为 nebula-operator chart 的版本，不指定 `--version` 时默认使用最新版的 chart。执行 `helm search repo -l nebula-operator` 查看 chart 版本。

用户可在执行安装 NebulaGraph Operator chart 命令时自定义 Operator 的配置。更多信息，查看下文自定义配置 **Chart**。

自定义配置 **Chart**

执行 `helm install [NAME] [CHART] [flags]` 命令安装 Chart 时，可指定 Chart 配置。更多信息，参考安装前自定义 Chart。

在 nebula-operator chart 配置文件中查看相关的配置选项。

或者通过命令 `helm show values nebula-operator/nebula-operator` 查看可配置的选项，如下所示。

```
[abby@master ~]$ helm show values nebula-operator/nebula-operator
image:
  nebulaOperator:
    image: vesoft/nebula-operator:v1.6.2
    imagePullPolicy: Always
  kubeRBACProxy:
    image: bitnami/kube-rbac-proxy:0.14.2
    imagePullPolicy: Always
  kubeScheduler:
    image: registry.k8s.io/kube-scheduler:v1.24.11
    imagePullPolicy: Always

imagePullSecrets: []
kubernetesClusterDomain: ""

controllerManager:
  create: true
  replicas: 2
  env: []
  resources:
    limits:
      cpu: 200m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 100Mi

admissionWebhook:
  create: false

scheduler:
  create: true
  schedulerName: nebula-scheduler
  replicas: 2
  env: []
  resources:
    limits:
      cpu: 200m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 100Mi
...
```


Copyright © Yueshu Graph Technology - All rights reserved.

部分参数描述如下：

参数	默认值	描述
`image.nebulaOperator.image`	`vesoft/nebula-operator:v1.6.2`	NebulaGraph Operator 的镜像，版本为1.6.2。
`image.nebulaOperator.imagePullPolicy`	`IfNotPresent`	镜像拉取策略。
`imagePullSecrets`	-	镜像拉取密钥。
`kubernetesClusterDomain`	`cluster.local`	集群域名。
`controllerManager.create`	`true`	是否启用 controller-manager。
`controllerManager.replicas`	`2`	controller-manager 副本数。
`admissionWebhook.create`	`false`	是否启用 Admission Webhook。默认关闭，如需开启，需设置为 `true` 并且需要安装 cert-manager。
`shceduler.create`	`true`	是否启用 Scheduler。
`shceduler.schedulerName`	`nebula-scheduler`	调度器名称。
`shceduler.replicas`	`2`	nebula-scheduler 副本数。

以下示例为在安装 NebulaGraph Operator 时，指定 NebulaGraph Operator 的 AdmissionWebhook 机制为开启状态（默认关闭 AdmissionWebhook）：

```
helm install nebula-operator nebula-operator/nebula-operator --namespace=<nebula-operator-system> --set admissionWebhook.create=true
```

更新 **NebulaGraph Operator**

1. 拉取最新的 Helm 仓库。

```
helm repo update
```

2. 通过 `--set` 传递配置参数，更新 NebulaGraph Operator。

- `--set`：通过命令行的方式新增或覆盖指定项。有关可以更新的配置项，查看上文自定义配置 **Chart**。

例如，更新 NebulaGraph Operator 的 AdmissionWebhook 机制为开启状态。

```
helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=nebula-operator-system --version=1.6.2 --set admissionWebhook.create=true
```

更多信息，参考 Helm 升级。

Copyright © Yueshu Graph Technology - All rights reserved.

升级 **NebulaGraph Operator**

版本兼容性

- 不支持升级 0.9.0 及以下版本的 NebulaGraph Operator 至 1.x 版本。
- 1.x 版本的 NebulaGraph Operator 不兼容 3.x 以下版本的 NebulaGraph。

1. 拉取最新的 Helm 仓库。

```
helm repo update
```

2. 升级 NebulaGraph Operator 至 1.6.2 版本。

```
helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=<namespace_name> --version=1.6.2
```

示例：

```
helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=nebula-operator-system --version=1.6.2
```

输出：

```
Release "nebula-operator" has been upgraded. Happy Helming!
NAME: nebula-operator
LAST DEPLOYED: Tue Nov 16 02:21:08 2021
NAMESPACE: nebula-operator-system
STATUS: deployed
REVISION: 3
TEST SUITE: None
NOTES:
NebulaGraph Operator installed!
```

3. 拉取最新的 CRD 配置文件。

Note

升级 Operator 后，需要同时升级相应的 CRD 配置，否则悦数图数据库集群创建会失败。有关 CRD 的配置，参见 apps.nebula-graph.io_nebulaclusters.yaml。

a. 下载 NebulaGraph Operator chart 至本地。

```
helm pull nebula-operator/nebula-operator --version=1.6.2
```

- `--version`：升级版本号。如不指定，则默认为最新版本。

b. 执行 `tar -zxvf` 解压安装包。

例如：解压 1.6.2 chart 包至 `/tmp` 路径下。

```
tar -zxvf nebula-operator-1.6.2.tgz -C /tmp
```

- `-C /tmp`：如不指定，则默认解压至当前路径。

4. 在 `nebula-operator` 目录下升级 CRD 配置文件。

```
kubectl apply -f crds/nebulacluster.yaml
```

输出：

```
customresourcedefinition.apiextensions.k8s.io/nebulaclusters.apps.nebula-graph.io configured
```

卸载 **NebulaGraph Operator**

1. 卸载 NebulaGraph Operator chart。

Copyright © Yueshu Graph Technology - All rights reserved.

```
helm uninstall nebula-operator --namespace=<nebula-operator-system>
```

2. 删除 CRD。

```
kubectl delete crd nebulaclusters.apps.nebula-graph.io
```

19.3.4 后续操作

- 部署悦数图数据库集群前，需部署 LM 并加载 License Key。更多信息，请参考部署 LM。
- 使用 NebulaGraph Operator 自动化部署悦数图数据库集群。更多信息，请参考使用 Kubectl 部署悦数图数据库集群或者使用 Helm 部署悦数图数据库集群。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.4 部署悦数图数据库

19.4.1 部署 LM

用户使用 Operator 部署悦数图数据库 3.5.0 及之后版本前，需要先部署 License Manager（LM）并且在 LM 中配置 License。LM 是一个独立的服务，用于管理悦数图数据库的 License。LM 会在悦数图数据库启动时，检查 License 是否有效，如果 License 无效，悦数图数据库将无法启动。

部署说明

Operator 暂不支持部署 LM，用户需要自行部署 LM。

由于 LM 需要存储数据，是有状态的服务，用户可通过 StatefulSet 资源类型部署 LM，也可以在 Kubernetes 集群外部部署 LM。

在 K8s 集群外部部署 LM

有关如何在 K8s 集群外部的机器上部署 LM，请参见安装和启动 LM。

> **Caution**
>
> 如果 LM 部署在 K8s 集群外部，需要确保 LM 服务的端口（默认为 `9119`）可以被 K8s 集群中的所有节点访问。

在 K8s 中使用 StatefulSet 部署 LM

前提条件

- 准备 LM 的镜像。
- 已创建 StorageClass，用于存储 LM 的数据。更多信息，参见 Storage Class。

部署步骤

1. 创建命名空间

```
# 创建 nebula-license-manager 命名空间。
kubectl create namespace nebula-license-manager
```

2. 创建 Secret，用于拉取私有仓库中 LM 镜像。

```
kubectl -n nebula-license-manager create secret docker-registry <image-pull-secret> \
--docker-server=DOCKER_REGISTRY_SERVER \
--docker-username=DOCKER_USER \
--docker-password=DOCKER_PASSWORD
```

- `<image-pull-secret>`：指定 Secret 的名称。
- DOCKER_REGISTRY_SERVE：指定拉取镜像的私有仓库服务器地址，例如 `reg.example-inc.com`
- DOCKER_USE：镜像仓库用户名。
- DOCKER_PASSWORD：镜像仓库密码。

3. 创建 LM 的 StatefulSet 资源配置文件。示例如下：

```
apiVersion: v1
kind: Service
metadata:
  name: nebula-license-manager
  namespace: nebula-license-manager
  labels:
    app: nebula-license-manager
spec:
  ports:
  - port: 9119
  selector:
    app: nebula-license-manager
```


Copyright © Yueshu Graph Technology - All rights reserved.

```yaml
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nebula-license-manager
  namespace: nebula-license-manager
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nebula-license-manager
  serviceName: nebula-license-manager
  template:
    metadata:
      labels:
        app: nebula-license-manager
    spec:
      containers:
      - name: nebula-license-manager
        image: # 填写对应的 LM 镜像地址。
        ports:
        - containerPort: 9119
        volumeMounts:
        - name: data
          mountPath: /usr/local/nebula-license-manager/data
        livenessProbe:
          httpGet:
            path: /health
            port: 9119
        readinessProbe:
          httpGet:
            path: /health
            port: 9119
      # 用于指定一个或多个用于拉取私有镜像的 Secret 名称。
      imagePullSecrets:
        - name: image-pull-secret # 表示 Secret 名称。
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      accessModes:
      - ReadWriteOnce
      storageClassName: local-path # 存储类别，根据自己的情况指定。
      resources:
        requests:
          storage: 2Gi
```

4. 创建 LM。

```
kubectl apply -f nebula-license-manager.yaml
```

5. 验证 LM 是否部署成功。

```
kubectl -n nebula-license-manager get pods
```

访问 LM

通过上述创建的 Service 可访问创建的 LM。执行如下命令，获取 LM 的 Service 的 IP 和端口。

```
kubectl -n nebula-license-manager get svc nebula-license-manager
```

返回：

```
NAME                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
nebula-license-manager   ClusterIP   10.108.xxx.219   <none>        9119/TCP   37m
```

通过 `ClusterIP` 类型的 Service IP 和 `9119` 端口即可在3.5.1集群内访问 LM，例如 `10.108.xxx.219:9119` 。 或者，通过域名 `<service-name>.<namespace>.svc.cluster.local` 和端口 `9119` 访问 LM，例如 `nebula-license-manager.nebula-license-manager.svc.cluster.local:9119` 。

监控 LM

用户可以使用监控工具（悦数运维监控、Prometheus）监控 LM 的状态及指标。详情参见监控 LM。

Copyright © Yueshu Graph Technology - All rights reserved.

使用 **LM** 管理 **License**

- 对于使用在 K8s 集群外部署的 LM 管理 License 相关命令，参见使用 LM。
- 对于使用在 K8s 集群内部署的 LM 管理 License 相关命令如下：

```
# 查看 License 信息。
kubectl -n nebula-license-manager exec -it nebula-license-manager-0 -- \
/usr/local/nebula-license-manager/nebula-license-manager-cli info

# 加载 License Key。
# 启动悦数图数据库前，必须加载 License Key。
kubectl -n nebula-license-manager exec -it nebula-license-manager-0 -- \
/usr/local/nebula-license-manager/nebula-license-manager-cli load --key XXXXX-XXXXX-...

# 查看 License 配额使用情况。
kubectl -n nebula-license-manager exec -it nebula-license-manager-0 -- \
  /usr/local/nebula-license-manager/nebula-license-manager-cli usage
```

后续操作

部署 LM 并加载 License 后，需要在悦数图数据库集群中通过 `licenseManagerURL` 参数配置 LM 的访问地址和端口。

详情参见使用 Kubectl 部署或使用 Helm 部署。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.4.2 使用 Kubectl 部署悦数图数据库集群

版本兼容性

1.x 版本的 NebulaGraph Operator 不兼容 3.x 以下版本的悦数图数据库。

前提条件

- 安装 NebulaGraph Operator
- 已创建 StorageClass
- 已安装 LM 并加载 License Key

Copyright © Yueshu Graph Technology - All rights reserved.

创建集群

本文以创建名为`nebula`的集群为例，说明如何部署悦数图数据库集群。

Copyright © Yueshu Graph Technology - All rights reserved.

1. 创建命名空间，例如 `nebula` 。如果不指定命名空间，默认使用 `default` 命名空间。

```
kubectl create namespace nebula
```

2. 创建 Secret，用于拉取私有仓库中悦数图数据库镜像。

```
kubectl -n <nebula> create secret docker-registry <image-pull-secret> \
--docker-server=DOCKER_REGISTRY_SERVER \
--docker-username=DOCKER_USER \
--docker-password=DOCKER_PASSWORD
```

- `<nebula>`： 存放该 Secret 的命名空间。
- `<image-pull-secret>`： 指定 Secret 的名称。
- DOCKER_REGISTRY_SERVE：指定拉取镜像的私有仓库服务器地址，例如 `reg.example-inc.com` 。
- DOCKER_USE：镜像仓库用户名。
- DOCKER_PASSWORD：镜像仓库密码。

3. 创建集群配置文件。

   联系销售人员获取完整配置示例。必须自定义修改以下参数，其他参数可按需修改。

   - `spec.metad.licenseManagerURL` - `spec.<graphd|metad|storaged>.image` - `spec.imagePullSecrets`

Copyright © Yueshu Graph Technology - All rights reserved.

示例配置的参数描述如下：

参数	默认值	描述
`metadata.name`	-	创建的悦数图数据库集群名称。
`spec.graphd.replicas`	`1`	Graphd 服务的副本数。
`spec.graphd.image`	`vesoft/nebula-graphd`	Graphd 服务的容器镜像。
`spec.graphd.version`	`v3.5.1`	Graphd 服务的版本号。
`spec.graphd.service`		访问 Graphd 服务的 Service 配置。
`spec.graphd.logVolumeClaim.storageClassName`	-	Graphd 服务的日志盘存储卷的存储类名称。使用示例配置时需要将其替换为事先创建的存储类名称，参见 Storage Classes 查看创建存储类详情。
`spec.metad.replicas`	`1`	Metad 服务的副本数。
`spec.metad.image`	`vesoft/nebula-metad`	Metad 服务的容器镜像。
`spec.metad.version`	`v3.5.1`	Metad 服务的版本号。
`spec.metad.dataVolumeClaim.storageClassName`	-	Metad 服务的数据盘存储配置。使用示例配置时需要将其替换为事先创建的存储类名称，参见 Storage Classes 查看创建存储类详情。
`spec.metad.logVolumeClaim.storageClassName`	-	Metad 服务的日志盘存储配置。使用示例配置时需要将其替换为事先创建的存储类名称，参见 Storage Classes 查看创建存储类详情。
`spec.storaged.replicas`	`3`	Storaged 服务的副本数。
`spec.storaged.image`	`vesoft/nebula-storaged`	Storaged 服务的容器镜像。
`spec.storaged.version`	`v3.5.1`	Storaged 服务的版本号。
`spec.storaged.dataVolumeClaims.resources.requests.storage`	-	Storaged 服务的数据盘存储大小，可指定多块数据盘存储数据。当指定多块数据盘时，路径为：`/usr/local/nebula/data1`、`/usr/local/nebula/data2` 等。
`spec.storaged.dataVolumeClaims.resources.storageClassName`	-	Storaged 服务的数据盘存储配置。使用示例配置时需要将其替换为事先创建的存储类名称，参见 Storage Classes 查看创建存储类详情。
`spec.storaged.logVolumeClaim.storageClassName`	-	Storaged 服务的日志盘存储配置。使用示例配置时需要将其替换为事先创建的存储类名称，参见 Storage Classes 查看创建存储类详情。
`spec.agent`	`{}`	Agent 服务的配置。用于备份和恢复及日志清理功能，如果不自定义该配置，将使用默认配置。
`spec.reference.name`	-	依赖的控制器名称。
`spec.schedulerName`	-	调度器名称。
`spec.imagePullPolicy`	悦数图数据库镜像的拉取策略。关于拉取策略详情，请参考 Image pull policy。	镜像拉取策略。
`spec.logRotate`	-	日志轮转配置。详情参见管理集群日志。
`spec.enablePVReclaim`	`false`	定义是否在删除集群后自动删除 PVC 以释放数据。详情参见回收 PV。

Copyright © Yueshu Graph Technology - All rights reserved.

悦数图数据库特有参数描述如下：

参数	默认值	描述
`spec.metad.licenseManagerURL`	-	配置指向 LM 的 URL，由 LM 的访问地址和端口（默认端口 `9119`）组成。例如，`192.168.8.100:9119`。必须配置此参数以获取 **License** 信息，否则无法使用悦数图数据库集群。
`spec.storaged.enableAutoBalance`	`false`	是否启用自动均衡。详情参见均衡扩容后的 Storage 数据。
`spec.enableBR`	`false`	定义是否启用 BR 工具。详情参见备份与恢复。
`spec.imagePullSecrets`	-	定义拉取私有仓库中镜像所需的 Secret。

Enterpriseonly

拉取悦数图数据库集群镜像前，请确保已联系销售人员（inqury@vesoft.com）获取悦数图数据库集群的镜像信息。

4. 创建悦数图数据库集群。

```
kubectl create -f apps_v1alpha1_nebulacluster.yaml
```

返回：

```
nebulacluster.apps.nebula-graph.io/nebula created
```

5. 查看悦数图数据库集群状态。

```
kubectl get nebulaclusters nebula
```

返回：

```
NAME     GRAPHD-DESIRED   GRAPHD-READY   METAD-DESIRED   METAD-READY   STORAGED-DESIRED   STORAGED-READY   AGE
nebula   1                1              1               1             3                  3                86s
```

扩缩容集群

仅支持通过 v1.1.0 及以上版本的 NebulaGraph Operator 扩缩容悦数图数据库集群。

用户可以通过编辑 `apps_v1alpha1_nebulacluster.yaml` 文件中的 `replicas` 的值进行悦数图数据库集群的扩缩容。

Copyright © Yueshu Graph Technology - All rights reserved.

扩容集群

本文举例扩容悦数图数据库集群中 Storage 服务至 5 个。步骤如下：

1. 将 `apps_v1alpha1_nebulacluster.yaml` 文件中 `storaged.replicas` 的参数值从 `3` 改为 `5` 。

```
  storaged:
    resources:
      requests:
        cpu: "500m"
        memory: "500Mi"
      limits:
        cpu: "1"
        memory: "1Gi"
    replicas: 5
    image: vesoft/nebula-storaged
    version: v3.5.1
    dataVolumeClaims:
    - resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    - resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
  reference:
    name: statefulsets.apps
    version: v1
  schedulerName: default-scheduler
```

2. 执行以下命令使上述更新同步至悦数图数据库集群 CR 中。

```
kubectl apply -f apps_v1alpha1_nebulacluster.yaml
```

3. 查看 Storage 服务的副本数。

```
kubectl get pods -l app.kubernetes.io/cluster=nebula
```

返回：

```
NAME                 READY   STATUS    RESTARTS   AGE
nebula-graphd-0      1/1     Running   0          2m
nebula-metad-0       1/1     Running   0          2m
nebula-storaged-0    1/1     Running   0          2m
nebula-storaged-1    1/1     Running   0          2m
nebula-storaged-2    1/1     Running   0          2m
nebula-storaged-3    1/1     Running   0          5m
nebula-storaged-4    1/1     Running   0          5m
```

由上可看出 Storage 服务的副本数被扩容至 5 个。

缩容集群

缩容集群的原理和扩容一样，用户只需将 `apps_v1alpha1_nebulacluster.yaml` 文件中的 `replicas` 的值缩小。具体操作，请参考上文的扩容集群部分。

Caution

目前仅支持对悦数图数据库集群中的 Graph 服务和 Storage 服务进行扩缩容，不支持扩缩容 Meta 服务。

删除集群

使用 Kubectl 删除悦数图数据库集群的命令如下：

```
kubectl delete -f apps_v1alpha1_nebulacluster.yaml
```

后续操作

连接悦数图数据库数据库

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.4.3 使用 Helm 部署悦数图数据库集群

版本兼容性

1.x 版本的 NebulaGraph Operator 不兼容 3.x 以下版本的悦数图数据库。

前提条件

- 安装 NebulaGraph Operator
- 已创建 StorageClass
- 已安装 LM 并加载 License Key

创建悦数图数据库集群

1. 添加 NebulaGraph Operator Helm 仓库

```
helm repo add nebula-operator https://vesoft-inc.github.io/nebula-operator/charts
```

2. 更新 Helm 仓库，拉取最新仓库资源。

```
helm repo update
```

3. 为安装集群所需的配置参数设置环境变量。

```
export NEBULA_CLUSTER_NAME=nebula         #悦数图数据库集群的名字。
export NEBULA_CLUSTER_NAMESPACE=nebula    #悦数图数据库集群所处的命名空间的名字。
export STORAGE_CLASS_NAME=fast-disks              #悦数图数据库集群的 StorageClass。
```

4. 为悦数图数据库集群创建命名空间（如已创建，略过此步）。

```
kubectl create namespace "${NEBULA_CLUSTER_NAMESPACE}"
```

5. 创建 Secret，用于拉取私有仓库中悦数图数据库镜像。

```
kubectl -n "${NEBULA_CLUSTER_NAMESPACE}" create secret docker-registry <image-pull-secret> \
--docker-server=DOCKER_REGISTRY_SERVER \
--docker-username=DOCKER_USER \
--docker-password=DOCKER_PASSWORD
```

- `<image-pull-secret>`：指定 Secret 的名称。
- DOCKER_REGISTRY_SERVE：指定拉取镜像的私有仓库服务器地址，例如 `reg.example-inc.com`。
- DOCKER_USE：镜像仓库用户名。
- DOCKER_PASSWORD：镜像仓库密码。

6. 创建悦数图数据库集群。

```
helm install "${NEBULA_CLUSTER_NAME}" nebula-operator/nebula-cluster \

    # 配置指向 LM 访问地址和端口，默认端口为`9119`。必须配置此参数以获取 License 信息。
    --set nebula.metad.licenseManagerURL=`192.168.8.XXX:9119` \
    # 配置集群中各服务的镜像地址。
    --set nebula.graphd.image=<reg.example-inc.com/test/graphd-ent> \
    --set nebula.metad.image=<reg.example-inc.com/test/metad-ent> \
    --set nebula.storaged.image=<reg.example-inc.com/test/storaged-ent> \
    # 配置拉取私有仓库中镜像的 Secret。
    --set nebula.imagePullSecrets=<image-pull-secret> \

    --set nameOverride=${NEBULA_CLUSTER_NAME} \
    --set nebula.storageClassName="${STORAGE_CLASS_NAME}" \
    # 指定悦数图数据库集群的版本。
    --set nebula.version=v3.5.1 \
    # 指定集群 chart 的版本，不指定则默认安装最新版本 chart。
    # 执行 helm search repo nebula-operator/nebula-cluster 命令可查看所有 chart 版本。
    --version=1.6.2
    --namespace="${NEBULA_CLUSTER_NAMESPACE}" \
```


Copyright © Yueshu Graph Technology - All rights reserved.

执行 `helm show values nebula-operator/nebula-cluster` 命令，或者单击 nebula-cluster/values.yaml 可查看悦数图数据库集群的所有配置参数。

单击 Chart parameters 查看可配置的集群参数的描述及默认值。

通过 `--set` 参数设置悦数图数据库集群的配置参数，例如，`--set nebula.storaged.replicas=3` 可设置悦数图数据库集群中 Storage 服务的副本数为 3。

7. 查看悦数图数据库集群创建状态。

```
kubectl -n "${NEBULA_CLUSTER_NAMESPACE}" get pod -l "app.kubernetes.io/cluster=${NEBULA_CLUSTER_NAME}"
```

扩缩容集群

仅支持通过 v1.1.0 及以上版本的 NebulaGraph Operator 扩缩容悦数图数据库集群。

用户可通过定义悦数图数据库中不同服务对应的 `replicas` 的值扩缩容 Nebuala Graph 集群。

例如，扩容悦数图数据库集群中 Storage 的副本数为 5（原始值为 2），命令如下：

```
helm upgrade "${NEBULA_CLUSTER_NAME}" nebula-operator/nebula-cluster \
    --namespace="${NEBULA_CLUSTER_NAMESPACE}" \
    --set nameOverride=${NEBULA_CLUSTER_NAME} \
    --set nebula.storageClassName="${STORAGE_CLASS_NAME}" \
    --set nebula.storaged.replicas=5
```

同理，将悦数图数据库集群中服务对应的 `replicas` 的值设置成小于原始值，即可实现集群服务的缩容。

Caution

目前仅支持对悦数图数据库集群中的 Graph 服务和 Storage 服务进行扩缩容，不支持扩缩容 Meta 服务。

用户可点击 nebula-cluster/values.yaml 查看 nebula-cluster chart 的更多配置。有关文件中配置项的解释，参考下文悦数图数据库 集群 **Chart** 配置参数说明。

删除集群

使用 Helm 删除集群的命令如下：

```
helm uninstall "${NEBULA_CLUSTER_NAME}" --namespace="${NEBULA_CLUSTER_NAMESPACE}"
```

或者使用真实值删除集群，例如：

```
helm uninstall nebula --namespace=nebula
```

后续操作

连接悦数图数据库

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.5 通过 Nebular Operator 连接悦数图数据库

使用 NebulaGraph Operator 创建悦数图数据库 集群后，用户可在悦数图数据库集群内部访问悦数图数据库，也可在集群外访问悦数图数据库。如果用户在集群内部，可以通过访问集群内的虚拟 IP 地址（ClusterIP）访问数据库。而如果用户在集群外部，则需要使用集群节点的公共 IP 地址或者通过部署 Nginx Ingress 控制器来访问悦数图数据库。IP 地址类型可以在创建集群的配置文件中，通过 `spec.graphd.service` 指定。关于 Service 的更多信息，参考Service。

19.5.1 前提条件

使用 NebulaGraph Operator 创建悦数图数据库集群。具体步骤参考使用 Kubectl 部署悦数图数据库集群 或者使用 Helm 部署悦数图数据库集群。

19.5.2 通过 `NodePort` 在悦数图数据库集群外部连接悦数图数据库

用户可创建 `NodePort` 类型的 Service，通过集群任一节点 IP 和暴露的节点端口，从集群外部访问集群内部的服务。用户也可以使用云厂商（例如 Azure、AWS 等）提供的负载均衡服务，设置 Service 的类型为 `LoadBalancer`，通过云厂商提供的负载均衡器的公网 IP 和端口，从集群外部访问集群内部的服务。

`NodePort` 类型的 Service 通过标签选择器 `spec.selector` 将前端的请求转发到带有标签 `app.kubernetes.io/cluster: <cluster-name>` 、 `app.kubernetes.io/component: graphd` 的 Graphd pod 中。

当根据集群示例模板，其中 `spec.graphd.service.type=NodePort` ，创建 NebulaGraph 集群后，NebulaGraph Operator 会自动在同一命名空间下，创建名为 `<cluster-name>-graphd-svc` 、类型为 `NodePort` 的 Service。通过任一节点 IP 和暴露的节点端口，可直接连接 NebulaGraph 数据库（参见下文第 4 步）。用户也可以根据自己的需求，创建自定义的 Service。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤如下：

1. 创建名为 `graphd-nodeport-service.yaml` 的文件。示例内容如下：

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
    app.kubernetes.io/managed-by: nebula-operator
    app.kubernetes.io/name: nebula-graph
  name: nebula-graphd-svc-nodeport
  namespace: default
spec:
  externalTrafficPolicy: Local
  ports:
  - name: thrift
    port: 9669
    protocol: TCP
    targetPort: 9669
  - name: http
    port: 19669
    protocol: TCP
    targetPort: 19669
  selector:
    app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
    app.kubernetes.io/managed-by: nebula-operator
    app.kubernetes.io/name: nebula-graph
  type: NodePort  # 设置 Service 类型为 NodePort。
```

- 悦数图数据库默认使用 `9669` 端口为客户端提供服务。`19669` 为 Graph 服务的 HTTP 端口号。
- `targetPort` 的值为映射至 Pod 的端口，可自定义。

2. 执行以下命令使 Service 服务在集群中生效。

```
kubectl create -f graphd-nodeport-service.yaml
```

3. 查看 Service 中悦数图数据库映射至集群节点的端口。

```
kubectl get services -l app.kubernetes.io/cluster=<nebula>  #<nebula>为变量值，请用实际集群名称替换。
```

返回：

```
NAME                          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                                      AGE
nebula-graphd-svc-nodeport    NodePort    10.107.153.129 <none>        9669:32236/TCP,19669:31674/TCP,19670:31057/TCP   24h
...
```

`NodePort` 类型的 Service 中，映射至集群节点的端口为 `32236`。

4. 使用节点 IP 和上述映射的节点端口连接 NebulaGraph。

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- <nebula_console_name> -addr <node_ip> -port <node_port> -u <username> -p <password>
```

示例如下：

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- nebula-console -addr 192.168.8.24 -port 32236 -u root -p vesoft
If you don't see a command prompt, try pressing enter.

(root@nebula) [(none)]>
```

- `--image`： 为连接悦数图数据库的工具 NebulaGraph Console 的镜像。
- `<nebula-console>`： 自定义的 Pod 名称。本示例为 `nebula-console`。
- `-addr`： 悦数图数据库集群中任一节点 IP 地址。本示例为 `192.168.8.24`。
- `-port`： 悦数图数据库映射至节点的端口。本示例为 `32236`。
- `-u`： 悦数图数据库账号的用户名。未启用身份认证时，可以使用任意已存在的用户名（默认为 root）。
- `-p`： 用户名对应的密码。未启用身份认证时，密码可以填写任意字符。

Copyright © Yueshu Graph Technology - All rights reserved.

19.5.3 在悦数图数据库集群内连接悦数图数据库

用户也可以创建 `ClusterIP` 类型的 Service，为集群内的其他 Pod 提供访问悦数图数据库的入口。通过该 Service 的 IP 和数据库 Graph 服务的端口号（`9669`），可连接悦数图数据库。更多信息，请参考 ClusterIP。

1. 创建名为 `graphd-clusterip-service.yaml` 的文件。示例内容如下：

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
    app.kubernetes.io/managed-by: nebula-operator
    app.kubernetes.io/name: nebula-graph
  name: nebula-graphd-svc
  namespace: default
spec:
  externalTrafficPolicy: Local
  ports:
  - name: thrift
    port: 9669
    protocol: TCP
    targetPort: 9669
  - name: http
    port: 19669
    protocol: TCP
    targetPort: 19669
  selector:
    app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
    app.kubernetes.io/managed-by: nebula-operator
    app.kubernetes.io/name: nebula-graph
  type: ClusterIP  # 设置 Service 类型为 ClusterIP。
```

- 悦数图数据库默认使用 `9669` 端口为客户端提供服务。`19669` 为 Graph 服务的 HTTP 端口号。
- `targetPort` 的值为映射至 Pod 的端口，可自定义。

2. 执行以下命令使 Service 服务在集群中生效。

```
kubectl create -f graphd-clusterip-service.yaml
```

3. 查看 Service，命令如下：

```
$ kubectl get service -l app.kubernetes.io/cluster=<nebula>  #<nebula>为变量值，请用实际集群名称替换。
NAME                      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                                  AGE
nebula-graphd-svc         ClusterIP   10.98.213.34   <none>        9669/TCP,19669/TCP,19670/TCP             23h
...
```

4. 使用上述 `<cluster-name>-graphd-svc` Service 的 IP 连接悦数图数据库：

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- <nebula_console_name> -addr <cluster_ip>  -port <service_port> -u <username> -p <password>
```

示例：

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- nebula-console -addr 10.98.213.34  -port 9669 -u root -p vesoft
```

- `--image`： 为连接悦数图数据库的工具 NebulaGraph Console 的镜像。
- `<nebula-console>`： 自定义的 Pod 名称。
- `-addr`： 连接 Graphd 服务的 IP 地址，即 `ClusterIP` 类型的 Service IP 地址。
- `-port`： 连接 Graphd 服务的端口。默认端口为 `9669`。
- `-u`： 悦数图数据库账号的用户名。未启用身份认证时，可以使用任意已存在的用户名（默认为 `root`）。
- `-p`： 用户名对应的密码。未启用身份认证时，密码可以填写任意字符。

如果返回以下内容，说明成功连接数据库：

```
If you don't see a command prompt, try pressing enter.

(root@nebula) [(none)]>
```


Copyright © Yueshu Graph Technology - All rights reserved.

用户还可以使用完全限定域名（**FQDN**）连接数据库，域名格式为 `<cluster-name>-graphd.<cluster-namespace>.svc.<CLUSTER_DOMAIN>` ， `CLUSTER_DOMAIN` 的默认值为 `cluster.local` 。

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- <nebula_console_name> -addr <cluster_name>-graphd-svc.default.svc.cluster.local -port <service_port> -u <username> -
p <password>
```

- `<service_port>` 为 Graph 服务默认的端口 `9669` 。

19.5.4 通过 `Ingress` 在悦数图数据库集群外部连接悦数图数据库

当集群中有多个 Pod 时，为每个 Pod 分别提供服务会变得非常困难和繁琐，而使用 Ingress 可以轻松解决这个问题。Ingress 可以将流量路由到集群内部的多个 Pod。

Nginx Ingress 是 Kubernetes 中的一个 Ingress 控制器（Controller），是对 Kubernetes Ingress 资源的一个实现，通过 Watch 机制感知 Kubernetes 集群中的 Ingress 资源。它将这些 Ingress 规则转换为 Nginx 配置并启动一个 Nginx 实例来处理流量。

用户可以通过 HostNetwork 和 DaemonSet 组合的模式使用 Nginx Ingress 从集群外部连接悦数图数据库集群。

由于使用 HostNetwork，Nginx Ingress 的 Pods 可能被调度在同一个节点上（当多个 Pod 尝试在同一个节点上监听相同的端口时，将会出现端口冲突）。为了避免这种情况，Nginx Ingress 以 DaemonSet 模式（确保集群中每个节点都运行一个 Pod 副本）部署在这些节点上，需先选择一些节点并为节点打上标签，专门用于部署 Nginx Ingress。

由于 Ingress 不支持 TCP 或 UDP 服务，为此 nginx-ingress-controller 使用 `--tcp-services-configmap` 和 `--udp-services-configmap` 参数指向一个 ConfigMap，该 ConfigMap 中的键指需要使用的外部端口，值指要公开的服务的格式，值的格式为 `<命名空间/服务名称>:<服务端口>` 。

例如指向名为 `tcp-services` 的 ConfigMap 的配置如下：

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: tcp-services
  namespace: nginx-ingress
data:
  9769: "default/nebula-graphd-svc:9669"
```


Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤如下：

1. 创建名为 `nginx-ingress-daemonset-hostnetwork.yaml` 的文件。

   单击 nginx-ingress-daemonset-hostnetwork.yaml 查看完整的 YAML 示例内容。

   > **Note**
   >
   > 上述 YAML 中的资源对象均使用 `nginx-ingress` 命名空间。用户可执行 `kubectl create namesapce nginx-ingress` 创建命名空间，或者自定义其他命名空间。

2. 为任一节点（本示例使用的节点名为 `worker2`，IP 为 `192.168.8.160`）打上标签，以运行上述 YAML 文件中名为 `nginx-ingress-controller` 的 DaemonSet。

```
kubectl label node worker2 nginx-ingress=true
```

3. 执行以下命令使 Nginx Ingress 在集群中生效。

```
kubectl create -f nginx-ingress-daemonset-hostnetwork.yaml
```

   返回：

```
configmap/nginx-ingress-controller created
configmap/tcp-services created
serviceaccount/nginx-ingress created
serviceaccount/nginx-ingress-backend created
clusterrole.rbac.authorization.k8s.io/nginx-ingress created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress created
role.rbac.authorization.k8s.io/nginx-ingress created
rolebinding.rbac.authorization.k8s.io/nginx-ingress created
service/nginx-ingress-controller-metrics created
service/nginx-ingress-default-backend created
service/nginx-ingress-proxy-tcp created
daemonset.apps/nginx-ingress-controller created
```

   成功部署 Nginx Ingress 后，由于 Nginx Ingress 中配置的网络类型为 `hostNetwork`，因此用户可通过部署了 Nginx Ingress 的节点的 IP（`192.168.8.160`）和外部端口（`9769`）访问悦数图数据库服务。

4. 执行以下命令部署连接悦数图数据库服务的 Console 并通过宿主机 IP（本示例为 `192.168.8.160`）和上述配置的外部端口访问悦数图数据库服务。

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- <nebula_console_name> -addr <host_ip> -port <external_port> -u <username> -p <password>
```

   示例：

```
kubectl run -ti --image vesoft/nebula-console:v3.5.0 --restart=Never -- nebula-console -addr 192.168.8.160 -port 9769 -u root -p vesoft
```

- `--image`： 为连接悦数图数据库的工具 NebulaGraph Console 的镜像。
- `<nebula-console>`： 自定义的 Pod 名称。本示例为 `nebula-console`。
- `-addr`： 部署 Nginx Ingress 的节点 IP，本示例为 `192.168.8.160`。
- `-port`： 外网访问使用的的端口。本示例设置为 `9769`。
- `-u`： 悦数图数据库账号的用户名。未启用身份认证时，可以使用任意已存在的用户名（默认为 root）。
- `-p`： 用户名对应的密码。未启用身份认证时，密码可以填写任意字符。

   如果返回以下内容，说明成功连接数据库：

```
If you don't see a command prompt, try pressing enter.

(root@nebula) [(none)]>
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.6 配置悦数图数据库

19.6.1 自定义悦数图数据库集群的配置参数

悦数图数据库集群中 Meta、Storage、Graph 服务都有各自的配置，其在用户创建的悦数图数据库集群实例的 YAML 文件中被定义为 `config`。`config` 中的设置会被映射并加载到对应服务的 ConfigMap 中。各个服务在启动时会挂载 ConfigMap 中的配置到 `/usr/local/nebula/etc/` 目录下。

> **Note**
>
> 暂不支持通过 Helm 自定义悦数图数据库集群的配置参数。

`config` 结构如下：

```
Config map[string]string `json:"config,omitempty"`
```

前提条件

已使用 K8s 创建一个集群。具体步骤，参见使用 Kubectl 创建悦数图数据库集群。

操作步骤

以下示例使用名为 `nebula` 的集群、名为 `nebula_cluster.yaml` 的 YAML 配置文件，说明如何为集群的 Graph 服务配置 `config`：

1. 执行以下命令进入 `nebula` 集群的编辑页面。

```
kubectl edit nebulaclusters.apps.nebula-graph.io nebula
```

2. 在 YAML 文件的 `spec.graphd.config` 配置项中，添加 `enable_authorize` 和 `auth_type`。

```yaml
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
  name: nebula
  namespace: default
spec:
  graphd:
    resources:
      requests:
        cpu: "500m"
        memory: "500Mi"
      limits:
        cpu: "1"
        memory: "1Gi"
    replicas: 1
    image: vesoft/nebula-graphd
    version: v3.5.1
    storageClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    config: //为 Graph 服务自定义参数。
      "enable_authorize": "true"
      "auth_type": "password"
...
```

如果需要为 Meta 服务和 Storage 服务配置 `config`，则在 `spec.metad.config` 和 `spec.storaged.config` 中添加对应的配置项。

3. 执行 `kubectl apply -f nebula_cluster.yaml` 使上述更新生效。

在自定义参数 `enable_authorize` 和 `auth_type` 后，Graph 服务对应的 ConfigMap（`nebula-graphd`）中的配置将被覆盖。

在线修改集群配置

在线修改集群配置主要通过调用 `HTTP` 接口对集群的配置参数进行修改，而无需重启集群 Pod。

Copyright © Yueshu Graph Technology - All rights reserved.

需要注意的是，只有当 `config` 中的配置项均为可在运行时动态修改的参数时，才能触发在线修改集群配置操作。如果 `config` 中的配置项包含不可动态修改的参数，那么集群配置将通过重启 Pod 的方式进行更新。

关于各个服务可动态修改的参数，分别参见 Meta 服务配置参数、Storage 服务配置参数、Graph 服务配置参数的配置参数表格中是否支持运行时动态修改一列。

更多信息

有关 Meta、Storage、Graph 服务的配置参数的详细介绍，参见 Meta 服务配置参数、Storage 服务配置参数、Graph 服务配置参数。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.6.2 回收 PV

NebulaGraph Operator 使用持久化卷 PV（Persistent Volume）和持久化卷声明 PVC（Persistent Volume Claim）来存储持久化数据。如果用户不小心删除了一个悦数图数据库集群，默认 PV 和 PVC 对象及其数据仍可保留，以确保数据安全。

用户也可以在集群实例的配置文件中通过设置参数 `spec.enablePVReclaim` 为 `true` 来定义在删除集群后自动删除 PVC 以释放数据。至于在删除 PVC 后是否删除 PV，用户需要自定义 PV 的回收策略。参见 StorageClass 中设置 reclaimPolicy 和 PV Reclaiming 了解 PV 回收策略。

前提条件

已使用 K8s 创建一个集群。具体步骤，参见使用 Kubectl 创建悦数图数据库集群。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

以下示例使用名为 `nebula` 的集群、名为 `nebula_cluster.yaml` 的 YAML 配置文件，说明如何设置 `enablePVReclaim`：

1. 执行以下命令进入 `nebula` 集群的编辑页面。

```
kubectl edit nebulaclusters.apps.nebula-graph.io nebula
```

2. 在 YAML 文件的 `spec` 配置项中，添加 `enablePVReclaim` 并设置其值为 `true`。

```yaml
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
  name: nebula
spec:
  enablePVReclaim: true  //设置其值为 true。
  graphd:
    image: vesoft/nebula-graphd
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    replicas: 1
    resources:
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 500Mi
    version: v3.5.1
  imagePullPolicy: IfNotPresent
  metad:
    dataVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    image: vesoft/nebula-metad
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    replicas: 1
    resources:
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 500Mi
    version: v3.5.1
  nodeSelector:
    nebula: cloud
  reference:
    name: statefulsets.apps
    version: v1
  schedulerName: default-scheduler
  storaged:
    dataVolumeClaims:
    - resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    - resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    image: vesoft/nebula-storaged
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    replicas: 3
    resources:
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 500Mi
    version: v3.5.1
...
```

3. 执行 `kubectl apply -f nebula_cluster.yaml` 使上述更新生效。

Copyright © Yueshu Graph Technology - All rights reserved.

当集群删除后，系统将会自动删除 PVC 以回收存储资源。是否删除 PV，取决于 PV 的回收策略。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.6.3 均衡扩容后的 Storage 数据

Enterpriseonly

仅企业版悦数图数据库集群支持均衡扩容后的 Storage 数据。

用户在扩容 Storage 服务后，可以自主决定是否均衡 Storage 服务中的数据。本文介绍如何设置均衡 Storage 数据。

悦数图数据库的 Storage 服务的扩容分为两个阶段，第一个阶段等待所有扩容的 Pods 为 `Ready` 的状态；第二个阶段执行 `BALANCE DATA` 和 `BALANCE LEADER` 命令均衡数据。这两个阶段将控制器副本的扩容过程与均衡数据过程解耦，使用户可以选择在低流量时进行均衡数据操作，有效降低数据迁移对在线服务的影响。

用户可以在集群的 CR 实例的配置文件中通过参数 `enableAutoBalance` 来控制是否自动均衡数据。

前提条件

已使用 K8s 创建一个集群。具体步骤，参见使用 Kubectl 创建悦数图数据库集群。

Copyright © Yueshu Graph Technology - All rights reserved.

操作步骤

以下示例使用名为 `nebula` 的集群、名为 `nebula_cluster.yaml` 的 YAML 配置文件，说明如何设置 `enableAutoBalance`：

Copyright © Yueshu Graph Technology - All rights reserved.

1. 执行以下命令进入 `nebula` 集群的编辑页面。

```
kubectl edit nebulaclusters.apps.nebula-graph.io nebula
```

2. 在 YAML 文件的 `spec.storaged` 配置项中，添加 `enableAutoBalance` 并设置其值为 `true` 。

```yaml
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
  name: nebula
spec:
  graphd:
    image: vesoft/nebula-graphd
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    replicas: 1
    resources:
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 500Mi
    version: v3.5.1
  imagePullPolicy: IfNotPresent
  metad:
    dataVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    image: vesoft/nebula-metad
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    replicas: 1
    resources:
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 500Mi
    version: v3.5.1
  nodeSelector:
    nebula: cloud
  reference:
    name: statefulsets.apps
    version: v1
  schedulerName: default-scheduler
  storaged:
    enableAutoBalance: true   //将其值设置为 true 时表示扩容后自动均衡 Storage 数据。
    dataVolumeClaims:
    - resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    - resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    image: vesoft/nebula-storaged
    logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    replicas: 3
    resources:
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 500Mi
    version: v3.5.1
...
```

- 当设置 `enableAutoBalance` 的值为 `true` 时，表示扩容后自动均衡 Storage 数据。
- 当设置 `enableAutoBalance` 的值为 `false` 时，表示扩容后不会均衡 Storage 数据。
- 当不设置 `enableAutoBalance` 参数时，默认扩容 Storage 后系统不会自动均衡 Storage 数据。

3. 执行 `kubectl apply -f nebula_cluster.yaml` 使上述更新生效。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.6.4 管理集群日志

悦数图数据库集群各服务（graphd、metad、storaged）在运行期间会生成运行日志，日志默认存放在各个服务容器的 `/usr/local/nebula/logs` 目录下。

查看运行日志

如果您需要查看悦数图数据库集群的运行日志，可以通过 `kubectl logs` 命令查看。

例如，查看 Storage 服务的运行日志：

```
// 查看 Storage 服务 Pod 的名称（nebula-storaged-0）。
$ kubectl get pods -l app.kubernetes.io/component=storaged
NAME                              READY   STATUS    RESTARTS      AGE
nebula-storaged-0                 1/1     Running   0             45h
...

// 进入 Storage 服务所在容器 storaged。
$ kubectl exec -it nebula-storaged-0 -c storaged -- /bin/bash

// 查看 Storage 服务的运行日志。
$ cd /usr/local/nebula/logs
```

清理日志

集群服务在运行期间生成的运行日志会占用磁盘空间，为避免占用过多磁盘空间，Operator 使用 sidecar 容器定期清理和归档日志。

为了方便日志的采集和管理，每个悦数图数据库服务都会部署一个 sidecar 容器，负责收集该服务容器产生的日志，并将其发送到指定的日志磁盘中。sidecar 容器使用 logrotate 工具自动清理和归档日志。

在集群实例的 YAML 配置文件中，通过 `spec.logRotate` 字段开启日志轮转功能，同时在 `spec.<graphd|metad|storaged>.config` 下设置 `timestamp_in_logfile_name` 值为 `false` 以自动对各服务日志进行清理和归档。默认情况下，日志轮转功能是关闭的。为悦数图数据库各个服务都开启日志轮转功能示例如下：

```yaml
...
spec:
  graphd:
    config:
      # 是否在日志文件名中包含时间戳，需设置为 false 以实现日志轮转。默认值为 true。
      "timestamp_in_logfile_name": "false"
  metad:
    config:
      "timestamp_in_logfile_name": "false"
  storaged:
    config:
      "timestamp_in_logfile_name": "false"
  logRotate: # 日志轮转配置
    # 日志文件在被删除前会被轮转的次数。默认值为 5，0 表示删除前不会被轮转。
    rotate: 5
    # 仅当日志文件增长超过定义的字节大小时才会轮转日志文件。默认值为 200M。
    size: "200M"
```

收集日志

如果不想挂载额外的日志磁盘备份日志文件，或者想通过诸如 fluent-bit 之类的服务收集日志并将其发送到日志中心，可以配置日志至标准错误输出。Operator 使用 glog 工具将日志记录到标准错误输出。

Caution

目前 Operator 仅收集标准错误日志。

在集群实例的 YAML 配置文件中，可以在各个服务下的 `config` 和 `env` 字段中配置日志记录到标准错误输出。

```yaml
...
spec:
  graphd:
    config:
      # 是否将标准错误重定向到单独的输出文件。默认值为 false，表示不重定向。
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
      redirect_stdout: "false"
      # 日志内容的严重程度级别：INFO、WARNING、ERROR 和 FATAL。取值分别为 0、1、2 和 3。
      stderrthreshold: "0"
    env:
    - name: GLOG_logtostderr # 日志写入标准错误而不是文件。
      value: "1" # 1 表示写入标准错误，0 表示写入文件中。
    image: vesoft/nebula-graphd
    replicas: 1
    resources:
      requests:
        cpu: 500m
        memory: 500Mi
    service:
      externalTrafficPolicy: Local
      type: NodePort
    version: v3.5.1
  metad:
    config:
      redirect_stdout: "false"
      stderrthreshold: "0"
    dataVolumeClaim:
      resources:
        requests:
          storage: 1Gi
      storageClassName: ebs-sc
    env:
    - name: GLOG_logtostderr
      value: "1"
    image: vesoft/nebula-metad
  ...
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.6.5 悦数图数据库集群的滚动更新策略

悦数图数据库集群使用分布式架构将数据分成多个逻辑分片，这些分片通常均分在不同的节点上。分布式系统中，同一份数据通常会有多个副本。为了保证多个副本数据的一致性，悦数图数据库集群使用 Raft 协议实现了多分片副本同步。Raft 协议中，每个分片都会选举出一个 Leader 副本，Leader 副本负责处理写请求，Follower 副本负责处理读请求。

通过 Operator 创建的悦数图数据库集群在滚动更新过程中，一个存储节点会暂时停止提供服务以进行更新。关于滚动更新的概述，参见执行滚动更新。如果 Leader 副本所在的节点停止提供服务，会导致该分片的读写不可用。为了避免这种情况，Operator 会在悦数图数据库集群滚动更新过程中，默认将 Leader 副本迁移到其他未受影响节点上。这样，当一个存储节点处于更新状态时，其他节点上的 Leader 副本能够继续处理客户端请求，以保证集群的读写可用性。

一个存储节点上的所有 Leader 副本全部迁移到其他节点的这个过程可能会持续较长时间。为了更好地控制滚动更新的时间，Operator 提供了一个名为 `enableForceUpdate` 参数。当确定没有外部访问流量时，可将该参数设置为 `true`，这样，Leader 副本将不会被迁移到其他节点上，从而加快滚动更新的速度。

滚动更新触发条件

Operator 会在以下情况下触发悦数图数据库集群的滚动更新：

- 悦数图数据库 集群的版本发生变化。
- 悦数图数据库 集群的配置发生变化。

配置滚动更新策略

在创建集群实例的 YAML 文件中，添加 `spec.storaged.enableForceUpdate` 参数，设置为 `true` 或 `false`，以控制滚动更新的速度。

当 `enableForceUpdate` 为 `true` 时，表示不迁移分片 Leader 副本，从而加快滚动更新的速度；反之，表示迁移分片 Leader 副本，以保证集群的读写可用性。默认值为 `false`。

Caution

设置 `enableForceUpdate` 为 `true` 时，确保没有流量进入集群进行读写操作。因为该设置会强制重建集群 Pod，此过程会发生数据丢失或客户端请求失败的情况。

配置示例：

```
...
spec:
...
  storaged:
    enableForceUpdate: true // 设置为 true 时，加快滚动更新的速度。
    ...
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.7 使用 NebulaGraph Operator 备份和恢复数据

本文介绍如何对 Kubernetes 上的悦数图数据库集群进行数据备份和数据恢复。

> **Enterpriseonly**

仅支持备份和恢复 Kubernetes 上的企业版悦数图数据库集群。

19.7.1 原理介绍

Backup&Restore工具是悦数图数据库企业版数据备份和恢复的命令行工具。NebulaGraph Operator 使用Backup&Restore工具来实现 Kubernetes 上的悦数图数据库集群数据的备份和恢复。

备份数据时，NebulaGraph Operator 会创建一个 Job，然后将悦数图数据库集群中的数据备份到指定的存储服务上。

恢复数据时，NebulaGraph Operator 会根据 NebulaRestore 资源对象定义的信息检查指定的备份悦数图数据库集群是否存在、远端存储的访问是否正常，并且会创建一个新的集群，然后将备份数据恢复到新创建的悦数图数据库集群中。更多信息参见恢复流程图。

19.7.2 前提条件

使用 NebulaGraph Operator 备份和恢复数据，需要满足以下条件：

- Nebula Operator 版本 >= 1.4.0。
- Kubernetes 上部署的企业版悦数图数据库集群正在运行。
- 在创建集群的 YAML 中，设置了 `spec.enableBR` 为 `true` 。

```
// 集群 YAML 文件示例部分内容。
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
  name: nebula
spec:
enableBR: true // 设置为 true 时，启用备份和恢复功能。
...
```

- 只支持使用 S3 协议的存储服务（例如 AWS S3，Minio 等）备份和恢复数据。
- 集群中有足够的计算资源以恢复数据。

19.7.3 备份数据

备份说明

- NebulaGraph Operator 支持全量备份和增量备份。
- 数据备份过程中，指定图空间中的 DDL 和 DML 语句将会阻塞，我们建议在业务低峰期进行操作，例如凌晨 2 点至 5 点。
- 执行增量备份的集群和指定的上一次备份的集群需为同一个，并且和指定的上一次备份的（存储桶）路径需相同。
- 确保每次增量备份距离上一次备份的时间小于一个 `wal_ttl` 的时间。
- 不支持备份指定图空间数据。
- 执行备份 Job 前，需要创建一个 Secret，用于存储拉取私有仓库中Backup&Restore镜像所需的凭证。

```
kubectl - <nebula> create secret docker-registry <br-ent-secret> \
--docker-server=REGISTRY_SERVER \
```


Copyright © Yueshu Graph Technology - All rights reserved.

```
  --docker-username=REGISTRY_USERNAME \
  --docker-password=REGISTRY_PASSWORD \
```

- `<nebula>`： 存放该 Secret 的命名空间。确保该命名空间与备份 Job 的命名空间相同。
- `<br-ent-secret>`： Secret 的名称。
- `REGISTRY_SERVER`： 指定拉取镜像的私有仓库服务器地址，例如 `reg.example-inc.com` 。
- `REGISTRY_USERNAME`： 镜像仓库用户名。
- `REGISTRY_PASSWORD`： 镜像仓库密码。

全量备份

当备份数据至兼容 S3 协议的存储服务上时，需要创建一个备份任务（Job），它会将全量悦数图数据库数据备份到指定的存储位置。

全量备份任务（Job）的 YAML 示例如下：

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: nebula-full-backup
spec:
  parallelism: 1
  ttlSecondsAfterFinished: 600
  template:
    spec:
      restartPolicy: OnFailure
      imagePullSecrets:
        - name: br-ent-secret  # 拉取私有仓库中Backup&Restore镜像所需的凭证的 Secret 的名称。
      containers:
        - image: reg.vesoft-inc.com/cloud-dev/br-ent:v3.5.1}
          imagePullPolicy: Always
          name: backup
          command:
            - /bin/sh
            - -ecx
            - 'exec /usr/local/bin/br-ent backup full
            --meta nebula-metad-0.nebula-metad-headless.<namespace>.svc.cluster.local:9559  # 指定 Meta 服务的地址，将 <namespace> 替换为集群所在的命名空间。
            --storage s3://BUCKET                          # 指定备份数据的存储位置。
            --s3.access_key ACCESS_KEY                     # 指定访问存储服务的 AccessKey。
            --s3.secret_key SECRET_KEY                     # 指定访问存储服务的 SecretKey。
            --s3.region REGION                             # 指定存储服务所在的地理区域。
            --s3.endpoint https://s3.REGION.amazonaws.com'   # 指定存储服务的访问地址。
```

增量备份

除了 `spec.template.spec.containers[0].command` 指定的执行命令和 Job 名称不同外，增量备份的 YAML 文件与全量备份的 YAML 文件相同。增量备份的 YAML 示例：

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: nebula-incr-backup
spec:
  parallelism: 1
  ttlSecondsAfterFinished: 600
  template:
    spec:
      restartPolicy: OnFailure
      imagePullSecrets:
        - name: br-ent-secret
      containers:
        - image: reg.vesoft-inc.com/cloud-dev/br-ent:v3.5.1
          imagePullPolicy: Always
          name: backup
          command:
            - /bin/sh
            - -ecx
            - 'exec /usr/local/bin/br-ent backup incr
            --meta nebula-metad-0.nebula-metad-headless.<namespace>.svc.cluster.local:9559 # 指定 Meta 服务的地址，将 <namespace> 替换为集群所在的命名空间。
            --storage s3://BUCKET                          # 指定备份数据的存储位置。
            --s3.access_key ACCESS_KEY                     # 指定访问存储服务的 AccessKey。
            --s3.secret_key SECRET_KEY                     # 指定访问存储服务的 SecretKey。
            --s3.region REGION                             # 指定存储服务所在的地理区域。
            --s3.endpoint https://s3.REGION.amazonaws.com'   # 指定存储服务的访问地址。
```


Copyright © Yueshu Graph Technology - All rights reserved.

参数说明

主要参数说明如下：

参数	默认值	说明
`spec.parallelism`	1	并行执行的任务数。
`spec.ttlSecondsAfterFinished`	60	在任务完成后，保留任务信息的时间。
`spec.template.spec.containers[0].image`	`vesoft/br-ent:3.5.1`	Backup&Restore工具的镜像地址。
`spec.template.spec.containers[0].command`	-	备份数据至兼容 S3 协议存储服务的命令。 有关命令中选项的描述，参见参数说明。

有关 Job 的更多设置，参见 Kubernetes Jobs。

设置完备份任务（Job）的 YAML 文件后，执行以下命令启动备份任务：

```
kubectl apply -f <backup_file_name>.yaml
```

数据备份成功后，会在指定的存储位置生成备份文件，例如 `BACKUP_2023_02_12_10_04_16` 。

19.7.4 恢复数据

恢复说明

- 数据恢复执行成功后，会创建一个新的集群，老的集群不会被删除，用户可自行决定是否删除老的集群。新集群的名称由系统自动生成。
- 数据恢复会有一段时间的服务不可用，建议在业务低峰期进行操作。

恢复过程

当恢复备份在兼容 S3 协议服务上的数据时，需要创建一个 Secret，用于存储访问兼容 S3 协议的服务的凭证。然后创建一个恢复数据的资源对象（NebulaRestore），它会告诉 Operator 基于此资源对象定义的信息创建一个新的悦数图数据库集群，将备份数据恢复到新创建的集群中。

基于备份文件 `BACKUP_2023_02_12_10_04_16` 恢复数据的 YAML 示例：

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: aws-s3-secret
type: Opaque
data:
  access-key: QVNJQVEOWFlxxx
  secret-key: ZFJ6OEdNcDdxenMwVGxxx
---
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaRestore
metadata:
  name: restore1
spec:
  br:
    clusterName: nebula
    backupName: "BACKUP_2023_02_12_10_04_16"
    concurrency: 5
    s3:
      region: "us-west-2"
      bucket: "nebula-br-test"
      endpoint: "https://s3.us-west-2.amazonaws.com"
      secretName: "aws-s3-secret"
```


Copyright © Yueshu Graph Technology - All rights reserved.

参数说明

- Secret

参数	默认值	说明
`metadata.name`	-	Secret 的名称。
`type`	`Opaque`	Secret 的类型。参见Types of Secret。
`data.access-key`	-	访问兼容 S3 协议的存储服务的 AccessKey。
`data.secret-key`	-	访问兼容 S3 协议的存储服务的 SecretKey。

- NebulaRestore

参数	默认值	说明
`metadata.name`	-	资源对象 NebulaRestore 的名称。
`spec.br.clusterName`	-	备份集群的名称。基于这个集群进行备份，然后创建新集群，新集群名称由系统自动生成。
`spec.br.backupName`	-	备份文件的名称。基于这个备份文件恢复数据。
`spec.br.concurrency`	`5`	用来控制数据恢复时下载文件的并发数量。默认值 `5` 。
`spec.br.s3.region`	-	S3 存储桶所在的地理区域。
`spec.br.s3.bucket`	-	存储备份数据的 S3 存储桶路径。
`spec.br.s3.endpoint`	-	S3 存储桶的访问地址。
`spec.br.s3.secretName`	-	用于访问 S3 存储桶的凭证的 Secret 的名称。

设置完恢复数据的 YAML 文件后，执行以下命令启动恢复数据的任务：

```
kubectl apply -f <restore_file_name>.yaml
```

执行以下命令查看 NebulaRestore 对象的状态。

```
kubectl get rt <NebulaRestore_name> -n <namespace>

# 输出示例：
NAME       STATUS     STARTED   COMPLETED   AGE
restore1   Complete   67m       59m         67m
```

在恢复任务完成后，会创建一个新的 NebulaGraph 集群。执行以下命令查看新集群的状态：

```
kubectl get nc -n <namespace>

# 输出示例：
NAME     GRAPHD-DESIRED   GRAPHD-READY   METAD-DESIRED   METAD-READY   STORAGED-DESIRED   STORAGED-READY   AGE
nebula   1                1              1               1             3                  3                2d3h
ngxvsm   1                1              1               1             3                  3                92m  # 新集群
```

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.8 故障自愈

NebulaGraph Operator 调用悦数图数据库集群提供的接口，动态地感知服务是否正常运行。当悦数图数据库集群中某一组件停止运行时，NebulaGraph Operator 会自动地进行容错处理。本文通过删除悦数图数据库集群中 1 个 Storage 服务 Pod，模拟集群故障为例，说明 Nebular Operator 如何进行故障自愈。

19.8.1 前提条件

安装 NebulaGraph Operator

19.8.2 操作步骤

1. 创建悦数图数据库集群。具体步骤参考使用 Kubectl 部署悦数图数据库集群或者使用 Helm 部署悦数图数据库集群。
2. 待所有 Pods 都处于 `Running` 状态时，模拟故障，删除名为 `<cluster_name>-storaged-2` Pod。

```
kubectl delete pod <cluster-name>-storaged-2 --now
```

`<cluster_name>` 为悦数图数据库集群的名称。

3. NebulaGraph Operator 自动创建名为 `<cluster-name>-storaged-2` 的 Pod，以修复故障。

执行 `kubectl get pods` 查看 `<cluster-name>-storaged-2` Pod 的创建状态。

```
...
nebula-cluster-storaged-1        1/1     Running             0          5d23h
nebula-cluster-storaged-2        0/1     ContainerCreating   0          1s
...
```

```
...
nebula-cluster-storaged-1        1/1     Running     0          5d23h
nebula-cluster-storaged-2        1/1     Running     0          4m2s
...
```

当 `<cluster-name>-storaged-2` 的状态由 `ContainerCreating` 变为 `Running` 时，说明自愈成功。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

19.9 常见问题

19.9.1 NebulaGraph Operator 支持 v1.x 版本的悦数图数据库吗?

不支持，因为 v1.x 版本的悦数图数据库不支持 DNS，而 NebulaGraph Operator 需要使用 DNS。

19.9.2 使用本地存储是否可以保证集群稳定性?

无法保证。使用本地存储意味着 Pod 被绑定到一个特定的节点，NebulaGraph Operator 目前不支持在绑定的节点发生故障时进行故障转移。

19.9.3 扩缩容集群时，如何确保稳定性?

建议提前备份数据，以便故障发生时回滚数据。

19.9.4 Operator 文档中的 replica 和悦数图数据库内核文档中的 replica 是一样的吗?

二者是不同的概念。Operator 文档中的 replica 是 K8s 中的 Pod 副本，而内核文档中的 replica 是悦数图数据库中的分片副本。

19.9.5 如何查看 NebulaGraph 集群中各服务的日志?

NebulaGraph 集群日志没有被收集到 K8s 集群中，因此无法通过 `kubectl logs` 查看。用户可以通过进入容器并在容器内查看日志文件的方式来获取 NebulaGraph 集群各服务日志。

操作如下：

```
# 查找要进入的容器所在的 Pod 名称。其中 <cluster-name> 为集群名称。
kubectl get pods -l app.kubernetes.io/cluster=<cluster-name>

# 进入 Pod 中的容器，例如进入 nebula-graphd-0 容器。
kubectl exec -it nebula-graphd-0 -- /bin/bash

# 进入 /usr/local/nebula/logs 目录，查看日志文件。
cd /usr/local/nebula/logs
```

19.9.6 `host not found:nebula-<metad|storaged|graphd>-0.nebula.<metad|storaged|graphd>-headless.default.svc.cluster.local` 错误如何解决?

这个错误一般是由于 DNS 解析失败导致的，需检查是否修改了集群域名。如果修改了集群域名，需要同步修改 Operator 配置文件中的 `kubernetesClusterDomain` 字段。同步修改 Operator 配置文件的操作如下：

1. 查看 Operator 配置文件。

```
[abby@master ~]$ helm show values nebula-operator/nebula-operator
image:
  nebulaOperator:
    image: vesoft/nebula-operator:v1.6.2
    imagePullPolicy: Always
  kubeRBACProxy:
    image: bitnami/kube-rbac-proxy:0.14.2
    imagePullPolicy: Always
  kubeScheduler:
    image: registry.k8s.io/kube-scheduler:v1.24.11
    imagePullPolicy: Always

imagePullSecrets: []
kubernetesClusterDomain: ""  # 集群域名，默认为 cluster.local。
```

2. 修改 `kubernetesClusterDomain` 字段的值为集群域名。

```
helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=<nebula-operator-system> --version=1.6.2 --set kubernetesClusterDomain=<cluster-domain>
```

为 Operator 所在的命名空间，为更新后的集群域名。

Copyright © Yueshu Graph Technology - All rights reserved.

19.9.6 host not found:nebula-<metad|storaged|graphd>-0.nebula.<metad|storaged|graphd>-headless.default.svc.cluster.local错误如何解决?

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

20. NebulaGraph Bench

NebulaGraph Bench 是悦数图数据库的性能测试工具，一款利用 LDBC 数据集对悦数图数据库进行性能测试的工具。

20.1 适用场景

- 生成测试数据并导入悦数图数据库。
- 对悦数图数据库集群进行性能测试。

20.2 更新说明

Release

详细使用说明请参见 NebulaGraph Bench。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

21. 常见问题 FAQ

本文列出了使用悦数图数据库 3.5.1 时可能遇到的常见问题，用户可以使用文档中心或者浏览器的搜索功能查找相应问题。

21.1 关于历史兼容性

版本兼容性

悦数图数据库 3.5.1 与 历史版本 （包括悦数图数据库 1.x 和 2.x) 的数据格式、客户端通信协议均双向不兼容。 使用老版本客户端连接新版本服务端，会导致服务进程退出。 数据格式升级参见升级悦数图数据库历史版本至当前版本。 客户端与工具均需要下载对应版本。

21.2 关于执行报错

21.2.1 如何处理错误信息 `-1005:GraphMemoryExceeded: (-2600)`

这个错误是由 Memory Tracker 发出的，它在观察到内存使用超过了设定阈值时发出警报。这个机制可以帮助避免服务进程被系统的 OOM（Out of Memory）killer 终止。解决步骤：

1. 检查内存使用情况：首先，你需要查看命令执行过程中的内存用量。如果内存使用量确实很高，那么这个错误可能是预期中的情况。
2. 检查 Memory Tracker 的配置：如果内存使用量并不高，检查 Memory Tracker 的相关配置。这包括 `memory_tracker_untracked_reserved_memory_mb`（未被跟踪的预留内存量，单位为 MB），`memory_tracker_limit_ratio`（内存限制比例），以及 `memory_purge_enabled`（是否启用内存清理）。关于 Memory Tracker 的配置，参见 memory tracker 配置。
3. 优化配置：根据实际情况，调整这些配置。例如，如果可用内存限制过低，可以调高 `memory_tracker_limit_ratio` 的值。

21.2.2 如何处理错误信息 `SemanticError: Missing yield clause`

从悦数图数据库 3.0.0 开始，查询语句 `LOOKUP` 、 `GO` 、 `FETCH` 必须用 `YIELD` 子句指定输出结果。详情请参见YIELD。

21.2.3 如何处理错误信息 `Host not enough!`

从 3.0.0 版本开始，在配置文件中添加的 Storage 节点无法直接读写，配置文件的作用仅仅是将 Storage 节点注册至 Meta 服务中。必须使用 `ADD HOSTS` 命令后，才能正常读写 Storage 节点。详情参见管理 Storage 主机。

21.2.4 如何处理错误信息 `To get the property of the vertex in 'v.age', should use the format 'var.tag.prop'`

从 3.0.0 版本开始， `pattern` 支持同时匹配多个 Tag，所以返回属性时，需要额外指定 Tag 名称。即从 `RETURN 变量名.属性名` 改为 `RETURN 变量名.Tag名.属性名` 。

21.2.5 如何处理错误信息 `Storage Error E_RPC_FAILURE`

报错原因通常为 Graph 服务向 Storage 服务请求了过多的数据，导致 Storage 服务超时。请尝试以下解决方案：

- 修改配置文件：在 `nebula-graphd.conf` 文件中修改 `--storage_client_timeout_ms` 参数的值，以增加 Storage client 的连接超时时间。该值的单位为毫秒（ms）。例如，设置 `--storage_client_timeout_ms=60000` 。如果 `nebula-graphd.conf` 文件中未配置该参数，请手动增加。提示：请在配置文件开头添加 `--local_config=true` 再重启服务。
- 优化查询语句：减少全库扫描型的查询，无论是否用 `LIMIT` 限制了返回结果的数量；用 GO 语句改写 MATCH 语句（前者有优化，后者无优化）。
- 检查 Storaged 是否发生过 OOM。(`dmesg |grep nebula`)。
- 为 Storage 服务器提供性能更好的 SSD 或者内存。
- 重试请求。

Copyright © Yueshu Graph Technology - All rights reserved.

21.2.6 如何处理错误信息 `The leader has changed. Try again later`

已知问题，通常需要重试 1-N 次 (N==partition 数量）。原因为 meta client 更新 leader 缓存需要 1-2 个心跳或者通过错误触发强制更新。

如果登录悦数图数据库时，出现该报错，可以考虑使用 `df -h` 查看磁盘空间，检查本地磁盘空间是否已满。

21.2.7 编译 Exchange、Connectors、Algorithm 时无法下载 SNAPSHOT 包

现象：编译时提示 `Could not find artifact com.vesoft:client:jar:xxx-SNAPSHOT` 。

原因：本地 maven 没有配置用于下载 SNAPSHOT 的仓库。maven 中默认的 central 仓库用于存放正式发布版本，而不是开发版本（SNAPSHOT）。

解决方案：在 maven 的 setting.xml文件的 `profiles` 作用域内中增加以下配置：

```
<profile>
   <activation>
      <activeByDefault>true</activeByDefault>
   </activation>
   <repositories>
      <repository>
          <id>snapshots</id>
          <url>https://oss.sonatype.org/content/repositories/snapshots/</url>
          <snapshots>
             <enabled>true</enabled>
          </snapshots>
    </repository>
   </repositories>
</profile>
```

21.2.8 如何处理错误信息 `[ERROR (-1004)]: SyntaxError: syntax error near`？

大部分情况下，查询语句需要有 `YIELD` 或 `RETURN` ，请检查查询语句是否包含。

21.2.9 如何处理错误信息 `can’t solve the start vids from the sentence`

查询引擎需要知道从哪些 VID 开始图遍历。这些开始图遍历的 VID，或者通过用户指定，例如：

```
> GO FROM ${vids} ...
> MATCH (src) WHERE id(src) == ${vids}
# 开始图遍历的 VID 通过如上办法指定
```

或者通过一个属性索引来得到，例如：

```
# CREATE TAG INDEX IF NOT EXISTS i_player ON player(name(20));
# REBUILD TAG INDEX i_player;

> LOOKUP ON player WHERE player.name == "abc" | ... YIELD ...
> MATCH (src) WHERE src.name == "abc" ...
# 通过点属性 name 的索引，来得到 VID
```

否则，就会抛出这样一个异常 `can’t solve the start vids from the sentence` 。

21.2.10 如何处理错误信息 `Wrong vertex id type: 1001`

检查输入的 VID 类型是否是 `create space` 设置的 `INT64` 或 `FIXED_STRING(N)` 。详情请参见 create space。

21.2.11 如何处理错误信息 `The VID must be a 64-bit integer or a string fitting space vertex id length limit.`

检查输入的 VID 是否超过限制长度。详情请参见 create space。

21.2.12 如何处理错误信息 `edge conflict` 或 `vertex conflict`

Storage 服务在毫秒级时间内多次收到插入或者更新同一点或边的请求时，可能返回该错误。请稍后重试。

Copyright © Yueshu Graph Technology - All rights reserved.

21.2.13 如何处理错误信息 `RPC failure in MetaClient: Connection refused`

报错原因通常为 metad 服务状态异常，或是 metad 和 graphd 服务所在机器网络不通。请尝试以下解决方案：

- 在 metad 所在服务器查看下 metad 服务状态，如果服务状态异常，可以重新启动 metad 服务。
- 在报错服务器下使用 `telnet meta-ip:port` 查看网络状态。
- 检查配置文件中的端口配置，如果端口号与连接时使用的不同，改用配置文件中的端口或者修改配置。

21.2.14 如何处理 `nebula-graph.INFO` 中错误日志 `StorageClientBase.inl:214] Request to "x.x.x.x":9779 failed: N6apache6thrift9transport19TTransportExceptionE: Timed Out`

报错原因可能是查询的数据量比较大，storaged 处理超时。请尝试以下解决方法：

- 导入数据时，手动 compaction，加速读的速度。
- 增加 Graph 服务与 Storage 服务的 RPC 连接超时时间，在 `nebula-graphd.conf` 文件里面修改 `--storage_client_timeout_ms` 参数的值。该值的单位为毫秒（ms），默认值为 60000 毫秒。

21.2.15 如何处理 `nebula-storaged.INFO` 中错误日志 `MetaClient.cpp:65] Heartbeat failed, status:Wrong cluster!` 或者 `nebula-metad.INFO` 含有错误日志 `HBProcessor.cpp:54] Reject wrong cluster host "x.x.x.x":9771!`

报错的原因可能是用户修改了 metad 的 ip 或者端口信息，或者 storage 之前加入过其他集群。请尝试以下解决方法：

用户到 storage 部署的机器所在的安装目录（默认安装目录为 `/usr/local/nebula`）下面将 `cluster.id` 文件删除，然后重启 storaged 服务。

21.2.16 如何处理报错信息 `Storage Error: More than one request trying to add/update/delete one edge/vertex at he same time.`

上述报错表示当前版本不允许同时对一个点或者一条边进行并发操作。如出现此报错请重新执行操作命令。

21.3 关于设计与功能

21.3.1 返回消息中 `time spent` 的含义是什么？

将命令 `SHOW SPACES` 返回的消息作为示例：

```
nebula> SHOW SPACES;
+--------------------+
| Name               |
+--------------------+
| "basketballplayer" |
+--------------------+
Got 1 rows (time spent 1235/1934 us)
```

- 第一个数字 `1235` 表示数据库本身执行该命令花费的时间，即查询引擎从客户端接收到一个查询，然后从存储服务器获取数据并执行一系列计算所花费的时间。
- 第二个数字 `1934` 表示从客户端角度看所花费的时间，即从客户端发送请求、接收结果，然后在屏幕上显示结果所花费的时间。

21.3.2 为什么在正常连接悦数图数据库后，`nebula-storaged` 进程的端口号一直显示红色？

`nebula-storaged` 进程的端口号的红色闪烁状态是因为 `nebula-storaged` 在启动流程中会等待 `nebula-metad` 添加当前 Storage 服务，当前 Storage 服务收到 Ready 信号后才会正式启动服务。从 3.0.0 版本开始，Meta 服务无法直接读写在配置文件中添加的 Storage 服务，配置文件的作用仅仅是将 Storage 服务注册至 Meta 服务中。用户必须使用 `ADD HOSTS` 命令后，才能使 Meta 服务正常读写 Storage 服务。更多信息，参见管理 Storage 主机。

21.3.3 为什么悦数图数据库的返回结果每行之间没有横线分隔了？

这是 NebulaGraph Console 2.6.0 版本的变动造成的，不是悦数图数据库内核的变更，不影响返回数据本身的内容。

Copyright © Yueshu Graph Technology - All rights reserved.

21.3.4 关于悬挂边

悬挂边 (Dangling edge) 是指一条边的起点或者终点在数据库中不存在。

悦数图数据库 3.5.1 的数据模型中，由于设计允许图中存在“悬挂边”； 没有 openCypher 中的 MERGE 语句。 对于悬挂边的保证完全依赖应用层面。 详见 INSERT VERTEX, DELETE VERTEX, INSERT EDGE, DELETE EDGE。

21.3.5 可以在 `CREATE SPACE` 时设置 `replica_factor` 为偶数（例如设置为 2）吗?

不要这样设置。

Storage 服务使用 Raft 协议（多数表决），为保证可用性，要求出故障的副本数量不能达到一半。

当机器数量为 1 时， `replica_factor` 只能设置为 `1` 。

当机器数量足够时，如果 `replica_factor=2` ，当其中一个副本故障时，就会导致系统无法正常工作；如果 `replica_factor=4` ，只能有一个副本可以出现故障，这和 `replica_factor=3` 是一样。以此类推，所以 `replica_factor` 设置为奇数即可。

建议在生产环境中设置 `replica_factor=3` ，测试环境中设置 `replica_factor=1` ，不要使用偶数。

21.3.6 是否支持停止或者中断慢查询

支持。

详情请参见终止查询。

21.3.7 使用 `GO` 和 `MATCH` 执行相同语义的查询，查询结果为什么不同?

原因可能有以下几种:

- `GO` 查询到了悬挂边。
- `RETURN` 命令未指定排序方式。
- 触发了 Storage 服务中 `max_edge_returned_per_vertex` 定义的稠密点截断限制。
- 路径的类型不同，导致查询结果可能会不同。
- `GO` 语句采用的是 `walk` 类型，遍历时点和边可以重复。
- `MATCH` 语句兼容 openCypher，采用的是 `trail` 类型，遍历时只有点可以重复，边不可以重复。

因路径类型不同导致查询结果不同的示例图和说明如下。

从点 A 开始查询距离 5 跳的点，都会查询到点 C（`A->B->C->D->E->C` ），查询 6 跳的点时， `GO` 语句会查询到点 D（`A->B->C->D->E->C->D` ），因为边 `C->D` 可以重复查询，而 `MATCH` 语句查询为空，因为边不可以重复。

所以使用 `GO` 和 `MATCH` 执行相同语义的查询，可能会出现 `MATCH` 语句的查询结果比 `GO` 语句少。

Copyright © Yueshu Graph Technology - All rights reserved.

关于路径的详细说明，请参见维基百科。

21.3.8 如何统计每种 Tag 有多少个点，每个 Edge type 有多少条边?

请参见 show-stats。

21.3.9 如何获取每种 Tag 的所有点，或者每种 Edge type 的所有边?

1. 建立并重建索引。

```
> CREATE TAG INDEX IF NOT EXISTS i_player ON player();
> REBUILD TAG INDEX i_player;
```

2. 使用 `LOOKUP` 或 `MATCH` 语句。例如：

```
> LOOKUP ON player;
> MATCH (n:player) RETURN n;
```

更多详情请参见 INDEX、LOOKUP 和 MATCH。

21.3.10 能不能用中文字符做标识符，比如图空间、Tag、Edge type、属性、索引的名称?

能，详情参见关键字和保留字。

21.3.11 获取指定点的出度（或者入度）?

一个点的“出度”是指从该点出发的“边”的条数。入度，是指指向该点的边的条数。

```
nebula > MATCH (s)-[e]->() WHERE id(s) == "given" RETURN count(e); #出度
nebula > MATCH (s)<-[e]-() WHERE id(s) == "given" RETURN count(e); #入度
```

因为没有对出度和入度的进行特殊处理（例如索引或者缓存），这是一个较慢的操作。特别当遇到超级节点时，可能会发生OOM。

21.3.12 是否有办法快速获取“所有”点的出度和入度?

没有直接命令。

可以使用 NebulaGraph Algorithm。

21.4 关于运维

21.4.1 运行日志文件过大时如何回收日志?

悦数图数据库的运行日志默认在 `/usr/local/nebula/logs/` 下，正常 INFO 级别日志文件为 `nebula-graphd.INFO, nebula-storaged.INFO, nebula-metad.INFO`，报警和错误级别后缀为 `.WARNING` 和 `.ERROR`。

悦数图数据库使用 glog 打印日志。glog 没有日志回收的功能，用户可以：

- 使用 crontab 设置定期任务回收日志文件，详情请参见 Glog should delete old log files automatically。
- 使用 logrotate 实现日志轮询。使用 logrotate 管理日志前需修改相应悦数图数据库服务的配置，将 `timestamp_in_logfile_name` 参数的值改成 `false`。

21.4.2 如何查看悦数图数据库版本

服务运行时：`nebula-console` 中执行命令 `SHOW HOSTS META`，详见 SHOW HOSTS

服务未运行时：在安装路径的 `bin` 目录内，执行 `./<binary_name> --version` 命令，可以查看到 version 和 GitHub 上的 commit ID，例如：

Copyright © Yueshu Graph Technology - All rights reserved.

```
$ ./nebula-graphd --version
```

- Docker Compose 部署

  查看 Docker Compose 部署的悦数图数据库版本，方式和编译安装类似，只是要先进入容器内部，示例命令如下：

```
docker exec -it nebula-docker-compose_graphd_1 bash
cd bin/
./nebula-graphd --version
```

- RPM/DEB 包安装

  执行 `rpm -qa |grep nebula` 即可查看版本。

21.4.3 修改 Host 名称后，旧的 Host 一直显示 `OFFLINE` 怎么办?

`OFFLINE` 状态的 Host 将在一天后自动删除。

21.4.4 如何查看 dmp 文件?

dmp 文件是错误报告文件，详细记录了进程退出的信息，可以用操作系统自带的 gdb 工具查看。Coredump 文件默认保存在启动二进制的当前目录下（默认为 `/usr/local/nebula` 目录），在悦数图数据库服务 crash 时，系统会自动生成。

1. 查看 Core 文件进程名字，pid 一般为数值。

```
$ file core.<pid>
```

2. 使用 gdb 调试。

```
$ gdb <process.name> core.<pid>
```

3. 查看文件内容。

```
$(gdb) bt
```

例如：

```
 $ file core.1316027
 core.1316027: ELF 64-bit LSB core file, x86-64, version 1 (SYSV), SVR4-style, from '/home/workspace/fork/nebula-debug/bin/nebula-metad --flagfile /home/k', real uid: 1008, effective uid:
1008, real gid: 1008, effective gid: 1008, execfn: '/home/workspace/fork/nebula-debug/bin/nebula-metad', platform: 'x86_64'

 $ gdb /home/workspace/fork/nebula-debug/bin/nebula-metad core.1316027

 $(gdb) bt
 #0  0x00007f9de58fecf5 in __memcpy_ssse3_back () from /lib64/libc.so.6
 #1  0x0000000000eb2299 in void std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::_M_construct<char*>(char*, char*, std::forward_iterator_tag) ()
 #2  0x0000000000ef71a7 in nebula::meta::cpp2::QueryDesc::QueryDesc(nebula::meta::cpp2::QueryDesc const&) ()
 ...
```

如果用户不清楚 dmp 打印出来的相关信息，可以将打印出来的内容，带上操作系统版本、硬件配置、Core文件产生前后的错误日志和可能导致错误的操作贴在 悦数图数据库 论坛上寻求帮助。

21.4.5 如何通过 systemctl 设置开机自动启动悦数图数据库服务?

1. 执行 `systemctl enable` 启动 metad、graphd 和 storaged 服务。

```
[root]# systemctl enable nebula-metad.service
Created symlink from /etc/systemd/system/multi-user.target.wants/nebula-metad.service to /usr/lib/systemd/system/nebula-metad.service.
[root]# systemctl enable nebula-graphd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/nebula-graphd.service to /usr/lib/systemd/system/nebula-graphd.service.
[root]# systemctl enable nebula-storaged.service
Created symlink from /etc/systemd/system/multi-user.target.wants/nebula-storaged.service to /usr/lib/systemd/system/nebula-storaged.service.
```

2. 配置 metad、graphd 和 storaged 的 service 文件，设置服务自动拉起。

Caution

Copyright © Yueshu Graph Technology - All rights reserved.

在配置service文件时需注意以下几点： - PIDFile、ExecStart、ExecReload、ExecStop 参数的路径需要与服务器上的一致。 - RestartSec 为重启前等待的时长(秒)，可根据实际情况修改。 - （可选）StartLimitInterval 为无限次重启，默认是 10 秒内如果重启超过 5 次则不再重启，设置为 0 表示不限次数重启。 - （可选）LimitNOFILE 为服务最大文件打开数，默认为1024，可根据实际情况修改。

配置 metad 服务的 service 文件：

```
$ vi /usr/lib/systemd/system/nebula-metad.service

[Unit]
Description=Nebula Graph Metad Service
After=network.target

[Service ]
Type=forking
Restart=always
RestartSec=15s
PIDFile=/usr/local/nebula/pids/nebula-metad.pid
ExecStart=/usr/local/nebula/scripts/nebula.service start metad
ExecReload=/usr/local/nebula/scripts/nebula.service restart metad
ExecStop=/usr/local/nebula/scripts/nebula.service stop metad
PrivateTmp=true
StartLimitInterval=0
LimitNOFILE=1024

[Install]
WantedBy=multi-user.target
```

配置 graphd 服务的 service 文件：

```
$ vi /usr/lib/systemd/system/nebula-graphd.service
[Unit]
Description=Nebula Graph Graphd Service
After=network.target

[Service]
Type=forking
Restart=always
RestartSec=15s
PIDFile=/usr/local/nebula/pids/nebula-graphd.pid
ExecStart=/usr/local/nebula/scripts/nebula.service start graphd
ExecReload=/usr/local/nebula/scripts/nebula.service restart graphd
ExecStop=/usr/local/nebula/scripts/nebula.service stop graphd
PrivateTmp=true
StartLimitInterval=0
LimitNOFILE=1024

[Install]
WantedBy=multi-user.target
```

配置 storaged 服务的 service 文件：

```
$ vi /usr/lib/systemd/system/nebula-storaged.service
[Unit]
Description=Nebula Graph Storaged Service
After=network.target

[Service]
Type=forking
Restart=always
RestartSec=15s
PIDFile=/usr/local/nebula/pids/nebula-storaged.pid
ExecStart=/usr/local/nebula/scripts/nebula.service start storaged
ExecReload=/usr/local/nebula/scripts/nebula.service restart storaged
ExecStop=/usr/local/nebula/scripts/nebula.service stop storaged
PrivateTmp=true
StartLimitInterval=0
LimitNOFILE=1024

[Install]
WantedBy=multi-user.target
```

3. reload 配置文件。

```
[root]# sudo systemctl daemon-reload
```

4. 重启服务。

```
$ systemctl restart nebula-metad.service
$ systemctl restart nebula-graphd.service
$ systemctl restart nebula-storaged.service
```


Copyright © Yueshu Graph Technology - All rights reserved.

21.5 关于连接

21.5.1 防火墙中需要开放哪些端口?

如果没有修改过配置文件中预设的端口，请在防火墙中开放如下端口：

服务类型	端口
Meta	9559, 9560, 19559
Graph	9669, 19669
Storage	9777 ~ 9780, 19779

如果修改过配置文件中预设的端口，请找出实际使用的端口并在防火墙中开放它们。

更多端口信息，参见公司产品端口全集。

21.5.2 如何测试端口是否已开放?

用户可以使用如下 telnet 命令检查端口状态：

```
telnet <ip> <port>
```

> **Note**
>
> 如果无法使用 telnet 命令，请先检查主机中是否安装并启动了 telnet。

示例：

```
// 如果端口已开放：
$ telnet 192.168.1.10 9669
Trying 192.168.1.10...
Connected to 192.168.1.10.
Escape character is '^]'.

// 如果端口未开放：
$ telnet 192.168.1.10 9777
Trying 192.168.1.10...
telnet: connect to address 192.168.1.10: Connection refused
```

21.5.3 会话数量达到 `max_sessions_per_ip_per_user` 所设值上限时怎么办?

当会话数量达到上限时，即超过 `max_sessions_per_ip_per_user` 所设值，在连接悦数图数据库的时候，会出现以下报错：

```
Fail to create a new session from connection pool, fail to authenticate, error: Create Session failed: Too many sessions created from 127.0.0.1 by user root. the threshold is 2. You can change it by modifying 'max_sessions_per_ip_per_user' in nebula-graphd.conf
```

用户可以通过以下三种方式解决问题：

1. 执行 `KILL SESSION` 命令，关闭不需要的 Session。详情参见终止会话。
2. 在 Graph 服务的配置文件 `nebula_graphd.conf` 中增大 `max_sessions_per_ip_per_user` 的值，增加可以创建的会话总数。重启 Graph 服务使配置生效。
3. 在 Graph 服务的配置文件 `nebula_graphd.conf` 中减小 `session_idle_timeout_secs` 参数的值，降低空闲会话的超时时间。重启 Graph 服务使配置生效。

Copyright © Yueshu Graph Technology - All rights reserved.

21.6 关于扩容、缩容

21.6.1 如何增加或减少 Meta、Graph、Storage 节点的数量

- 悦数图数据库 3.5.1 未提供运维命令以实现自动扩缩容，但可参考以下步骤实现手动扩缩容：
- Meta 的扩容和缩容： Meta 不支持自动扩缩容。

> **Note**
>
> 用户可以使用脚本工具 迁移 Meta 服务，但是需要自行修改 Graph 服务和 Storage 服务的配置文件中的 Meta 设置。

- Graph 的缩容：将该 Graph 的 IP 从 client 的代码中移除，关闭该 Graph 进程。
- Graph 的扩容：在新机器上准备 Graph 二进制文件和配置文件，在配置文件中修改或增加已在运行的 Meta 地址，启动 Graph 进程。
- Storage 的缩容：执行 `SUBMIT JOB BALANCE DATA REMOVE <ip:port>` 和 `DROP HOSTS <ip:port>` 。完成后关闭 Storage 进程。

> **Caution**
>
> - 执行该命令迁移指定 Storage 节点中的分片数据前，需要确保其他 Storage 节点数量足够以满足设置的副本数。例如，如果设置了副本数为 3，那么在执行该命令前，需要确保其他 Storage 节点数量大于等于 3。
> - 如果待迁移的 Storage 节点中有多个图空间的分片，需要分别在每个图空间内执行该命令以迁移该 Storage 节点中的所有分片。

- Storage 的扩容：在新机器上准备 Storage 二进制文件和配置文件，在配置文件中修改或增加已在运行的 Meta 地址，然后注册 Storage 到 Meta 并启动 Storage 进程。详情参见注册 Storage 服务。

  Storage 扩缩容之后，根据需要执行 `SUBMIT JOB BALANCE DATA` 将当前图空间的分片平均分配到所有 Storage 节点中和执行 `SUBMIT JOB BALANCE LEADER` 命令均衡分布所有图空间中的 leader。运行命令前，需要选择一个图空间。

- 使用悦数运维监控，在可视化页面对 Graph 和 Storage 进行快速扩缩容，详情参见集群操作-扩缩容。
- 使用 NebulaGraph Operator 扩缩容集群，详情参见使用 Kubectl 部署悦数图数据库集群和使用 Helm 部署悦数图数据库集群。

21.6.2 如何在 Storage 节点中增加或减少磁盘

目前，Storage 不支持动态识别新的磁盘。可以通过以下步骤在分布式集群的 Storage 节点中增加或减少磁盘：

1. 执行 `SUBMIT JOB BALANCE DATA REMOVE <ip:port>` 将待增加或减少磁盘的 Storage 节点迁到其他 Storage 节点中。

> **Caution**
>
> - 执行该命令迁移指定 Storage 节点中的分片数据前，需要确保其他 Storage 节点数量足够以满足设置的副本数。例如，如果设置了副本数为 3，那么在执行该命令前，需要确保其他 Storage 节点数量大于等于 3。
> - 如果待迁移的 Storage 节点中有多个图空间的分片，需要分别在每个图空间内执行该命令以迁移该 Storage 节点中的所有分片。

2. 执行 `DROP HOSTS <ip:port>` 移除待增加或减少磁盘的 Storage 节点。
3. 在所有 Storage 节点配置文件中通过 `--data_path` 配置新增或减少磁盘路径，详情参见 Storage 配置文件。
4. 执行 `ADD HOSTS <ip:port>` 重新添加待增加或减少磁盘的 Storage 节点。
5. 根据需要执行 `SUBMIT JOB BALANCE DATA` 将当前图空间的分片平均分配到所有 Storage 节点中和执行 `SUBMIT JOB BALANCE LEADER` 命令均衡分布所有图空间中的 leader。运行命令前，需要选择一个图空间。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22. 附录

22.1 Release Note

22.1.1 悦数图数据库 3.5.1 release notes

优化

- 当 `MATCH` 语句查询不存在的属性时不再进行全表扫描。
- 支持 `MATCH...STARTS WITH` 语句的下推。

缺陷修复

- 修复连续重启 drainer 导致无法正常停止 drainer 的问题。
- 修复单个大查询可能导致 Graph 服务崩溃的问题。
- 修复 `FIND ALL PATH` 语句导致内存不足的问题。
- 修复 `MATCH` 语句中添加路径变量导致 `all()` 函数下推优化失效的问题。
- 修复低版本 Linux 内核使用 Boost 生成 UUID 时报错的问题。
- 修复 `MATCH...shortestpath()` 语句进行循环检测时报错的问题。
- 修复使用管道符（`|`）删除边时 Graph 服务崩溃的问题。
- 修复 `MATCH` 语句进行多跳查询时边属性无法显示的问题。
- 修复 `shortestPath()` 函数导致 Graph 服务崩溃的问题。
- 修复 `FIND ALL PATH` 语句查找点的自环时未返回自环路径的问题。
- 修复 `GO` 的复合语句中多个子句使用同一个变量时，重复执行该复合语句会返回不同结果的问题。
- 修复 `CREATE SPACE...AS` 语句克隆出的图空间中，新建索引会覆盖旧索引的问题。
- 修复 `GO...UNION ALL` 语句在某些场景下报错的问题。

历史版本

历史版本

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22.1.2 悦数运维监控更新说明

v3.5.1

- 优化
- 安装 License Manager 时检测安装目录是否为空。

- 缺陷修复
- 修复删除集群时不会删除 node_exporter 的问题。
- 修复多个 SSH 连接不释放的问题。
- 修复批量导入节点报错的问题。
- 修复 3.4 版本升级到 3.5 版本时的兼容问题。
- 修复未通过本软件部署 License Manager 时加载 License Key 报错的问题。

v3.5.0

- 功能
- 支持通过悦数运维监控一键部署 License Manager (LM)。
- 支持全量备份至本地。
- 增加慢查询分析功能。
- 支持配置集群诊断的打分公式。
- 更新配置增加新增配置、查看当前配置的生效值以及查看不一致配置功能。
- 支持在通知设置中为 webhook 配置 body。
- 支持自定义监控面板。

- 优化
- 集群拓扑一致性：扩缩容后无需用户手动刷新与授权。
- 集群总览页面优化。
- 集群间数据同步优化。
- 扩容新增节点配置默认与集群中第一个节点保持一致。
- 优化集群诊断报告内容。
- 支持在 `config.yaml` 文件中修改 `Prometheus` 服务的端口号。

v3.4.2

- 优化
- 支持在备份恢复页面查看数据备份恢复的进度。
- 内置悦数图数据库 v3.4.1 安装包。

Copyright © Yueshu Graph Technology - All rights reserved.

v3.4.1

- 缺陷修复
- 修复 RPM 包因为权限问题无法执行 `nebula-agent` 的问题。
- 修复 goconfig 文件夹权限问题导致无法查看集群导入信息的问题。
- 修复当许可证过期时间小于 `30` 天且 `gracePeriod` 大于 `0` 时页面提示错误的问题。

v3.4.0

- 功能
- 支持查看悦数图数据库集群运行日志。
- 支持查看悦数图数据库集群审计日志。
- 支持作业管理。
- 备份恢复支持增量备份。
- 内置 dashboard.service 脚本，支持一键管理悦数运维监控服务和查看悦数运维监控版本。
- 新增产品反馈页面。

- 优化
- 创建集群时自动检测安装包是否适配操作系统。
- 批量导入节点时支持指定悦数图数据库安装目录。
- 删除集群时支持同时删除安装目录。
- 导入集群和服务监控中显示依赖服务。
- 告警规则静默支持中途取消。
- 支持强杀 Graph 服务进程。
- 支持展示和修改多个服务的配置信息。
- 支持修改 Meta 服务配置。
- 操作记录支持记录更新配置和删除备份操作。
- LDAP 开启后支持自动注册。
- 任务中心日志信息更加详细。
- 浏览器兼容提示。
- 悦数图数据库许可证到期提醒。
- 支持红旗操作系统 Asianux Linux 7 (Core)。
- 优化连接数据库、创建集群、扩缩容、批量导入节点等多处交互。
- 优化接口报错提示。
- 节点监控的总览页中显示监控指标名称。
- 优化 `num_queries` 等监控指标的计算方式，调整为时序聚合显示。

- 缺陷修复
- 修复服务监控的总览页面中选择监控时间范围不生效的问题。
- 修复缩容时删除空节点没有删除对应悦数图数据库文件的问题。
- 修复切换诊断报告语言时，同时切换了全局语言的问题。
- 修复某个导入集群任务阻塞导致其他导入任务一直处于等待状态的问题。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22.1.3 悦数图探索版本更新说明

v3.5.1

- 缺陷修复
- 修复错误链接。
- 修复错误文案。
- 移除弃用的工具组件。

v3.5.0

- 功能
- 支持通过配置 NFS 使用工作流。
- 允许用户对产品进行个性化自定义，包括页面 Logo 和产品名称。
- 数据导入支持历史任务重新导入，数据源类型新增支持云存储和 SFTP。
- 支持新版 License。

Copyright © Yueshu Graph Technology - All rights reserved.

v3.4.0

- 功能
- 支持查看 Schema 的创建语句。
- 全局设置页面增加 Beta 功能开关。
- 新增产品反馈页面。

- 优化
- 优化慢查询，取消超时限制。
- 切换页面后控制台保留历史记录。
- 控制台支持使用 `#` 添加注释。
- 创建模版时支持使用 `#` 或 `//` 添加注释。
- 更新全局设置页面。
- 支持可视化修改 IP 白名单。
- 画布中默认展示 VID。
- 浏览器兼容提示。
- 连接信息展示内核版本。
- 内置数据集增加索引。
- 优化登录页。
- 优化 Workflow：
- 增加算法说明。
- 优化图算法的参数配置。
- 优化结果的展示。
- 优化交互：
- 节点筛选
- Tag 查询
- 路径查询
- 优化展示：
- 优化 Schema 统计信息展示。
- 优化力导向图布局。
- 优化可视化查询结果导入画布后的布局。
- 优化悬挂边上的点的展示。
- 优化控制台界面。
- 优化提示：
- 优化引导提示。
- 优化报错提示。

- 缺陷修复
- 修复无法查看导入任务日志的问题。
- 修复 `demo_basketballplayer` 数据集中缺失部分边数据的问题。
- 修复页面崩溃的问题。
- 修复 Workflow 中图计算结果导入画布后无法展示点的详细信息的问题。

Copyright © Yueshu Graph Technology - All rights reserved.

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22.2 悦数图数据库学习路径

本文介绍悦数图数据库学习路径，用户可以通过路径中的文档及视频由浅入深地学习悦数图数据库。

22.2.1 1. 关于悦数图数据库

1.1 什么是悦数图数据库?

文档	视频	PPT
什么是悦数图数据库	悦数图数据库介绍、万亿级别的悦数图数据库	悦数图数据库 3年回顾

1.2 图相关术语

视频
图世界的那些概念、术语
图数据库简述

1.3 数据模型

文档
数据模型

1.4 路径

文档	视频
路径	路径类型

1.5 产品架构

文档	视频
Meta 服务	-
Graph 服务	-
Storage 服务	-

22.2.2 2. 快速入门

2.1 安装悦数图数据库

文档	视频
使用 RPM/DEB 包	-
使用 TAR 包	-
使用 Docker	-
使用源码	图数据库入门教程（零）通过编译源码来安装悦数图数据库
-	如何选择部署方式?

Copyright © Yueshu Graph Technology - All rights reserved.

2.2 启动悦数图数据库

文档

启停悦数图数据库

2.3 连接悦数图数据库

文档

连接悦数图数据库

2.4 使用 **nGQL** 命令

文档

nGQL 命令汇总

22.2.3 3. 进阶操作

3.1 部署多机集群

文档

使用 RPM/DEB 包部署悦数图数据库多机集群

3.2 升级集群版本

文档

升级悦数图数据库

3.3 配置**Nebula**

文档

配置 Meta

配置 Graph

配置 Storage

配置 Linux 内核

3.4 配置日志

文档

日志配置

Copyright © Yueshu Graph Technology - All rights reserved.

3.5 运维与管理

- 账号鉴权和授权

  文档

  本地身份验证

  OpenLDAP

  管理用户

  内置角色

- 平衡分片分布

  文档

  Storage 负载均衡

- 监控

  文档

  Nebula 指标

  RocksDB 统计数据

- 数据快照

  文档

  创建快照

- SSL 加密

  文档

  SSL 加密

3.6 性能调优

文档	视频
图建模	Nebula 高性能图 schema 设计
系统建模	-
Compaction 策略	-

Copyright © Yueshu Graph Technology - All rights reserved.

3.7 周边工具

- 可视化

可视化工具	文档	视频
数据可视化	Nebula Studio	Nebula Studio 图探索功能和Nebula Studio 可视化建模
数据监控和运维	悦数运维监控和 Nebula Dashboard 社区版	悦数运维监控
数据分析	悦数图探索	悦数图探索

- 数据导入与导出

导入与导出	文档	视频
数据导入	Nebula Importer	Nebula Importer
数据导入	Nebula Spark Connector	-
数据导入	Nebula Flink Connector	-
数据导入	Nebula Exchange 社区版	悦数图数据库数据导入工具——Exchange、Exchange 导入 SST 数据
数据导出	Nebula Exchange 企业版	-

- 备份与恢复

文档	视频
Nebula BR	-

- 性能测试

文档
Nebula Bench

- 集群运维

文档	视频
Nebula Operator	Nebula Operator、 云原生的演进

- 图算法

文档	视频
Nebula Algorithm	Nebula Algorithm 介绍

- 客户端

文档
Nebula Console
Nebula CPP
Nebula Java
Nebula Python
Nebula Go

Copyright © Yueshu Graph Technology - All rights reserved.

22.2.4 4. 高阶操作

文档

API & SDK

22.2.5 5. 最佳实践

文档 **&** 视频

LDBC 数据导入及 nGQL 实践

基于悦数图数据库的 Betweenness Centrality 算法

百亿级图数据在快手安全情报的应用与挑战

美团图数据库平台建设及业务实践

Akulaku 的智能风控实践

微信使用悦数图数据库的定制化开发

知乎使用悦数图数据库实践

悦数图数据库在微众银行的数据治理业务的实践

图数据库在安全风控场景的应用 @BOSS 直聘

更多文档 、视频

22.2.6 6. 常见问题

文档

常见问题 FAQ

22.2.7 视频

用户也可以观看视频快速了解悦数图数据库的相关概念和操作。

悦数图数据库 **Academy** 系列课程

- 第一课：图的概念（03 分 45 秒）

- 第二课：图的结构（02 分 24 秒）

请访问 Bilibili 空间，查看更多视频。

热点视频

- Foesa 小学姐课堂——悦数图数据库那些磨人的概念（04 分 20 秒）

- Foesa 小学姐课堂—— `path` 的三种类型（03 分 09 秒）

- Foesa 小学姐课堂——悬挂边 （02 分 27 秒）

Copyright © Yueshu Graph Technology - All rights reserved.

- Foesa 小学姐课堂——自环（02 分 53 秒）

- Nebula Explore Demo Show（02 分 53 秒）

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22.3 产品端口全集

以下是悦数图数据库内核及周边工具使用的默认端口信息:

Copyright © Yueshu Graph Technology - All rights reserved.

序号	所属产品/服务	类型	默认端口	说明
1	悦数图数据库	TCP	9669	Graph 服务的 RPC 守护进程监听端口（通常用于客户端连接 Graph服务）。
2	悦数图数据库	TCP	19669	Graph 服务的 HTTP 端口。
3	悦数图数据库	TCP	19670	Graph 服务的 HTTP/2 端口。（3.x 后已弃用该端口）
4	悦数图数据库	TCP	9559	Meta 服务的 RPC 守护进程监听端口。（通常由 Graph 服务和 Storage 服务发起请求，用于获取和更新图数据库的元数据信息。
5	悦数图数据库	TCP	9560	Meta 服务之间的 Raft 通信端口。
6	悦数图数据库	TCP	19559	Meta 服务的 HTTP 端口。
7	悦数图数据库	TCP	19560	Meta 服务的 HTTP/2 端口。（3.x 后已弃用该端口）
8	悦数图数据库	TCP	9777	Storage 服务中，Drainer 服务占用端口（仅在企业版集群中暴露）。
9	悦数图数据库	TCP	9778	Storage 服务中，Admin 服务占用端口。
10	悦数图数据库	TCP	9779	Storage 服务的 RPC 守护进程监听端口。（通常由 Graph 服务发起请求，用于执行数据存储相关的操作，例如读取、写入或删除数据。）
11	悦数图数据库	TCP	9780	Storage 服务之间的 Raft 通信端口。
12	悦数图数据库	TCP	19779	Storage 服务的 HTTP 端口。
13	悦数图数据库	TCP	19780	Storage 服务的 HTTP/2 端口。（3.x 后已弃用该端口）
14	悦数图数据库	TCP	8888	备份和恢复功能的 Agent 服务端口。Agent 是集群中每台机器的一个守护进程，用于启停悦数图数据库服务和上传、下载备份文件。
15	悦数图数据库	TCP	9789、9790、9788	全文索引中 Raft Listener 的端口，从 Storage 服务读取数据，然后将它们写入 Elasticsearch 集群。 也是集群间数据同步中 Storage Listener 的端口。用于同步主集群的 Storage 数据。端口 9790、9788 由端口 9789 加一减一后自动生成。
16	悦数图数据库	TCP	9200	悦数图数据库使用该端口与 Elasticsearch 进行 HTTP 通信，以执行全文搜索查询和管理全文索引。
17	悦数图数据库	TCP	9569、9570、9568	集群间数据同步功能中 Meta Listener 的端口，用于同步主集群的 Meta 数据。端口 9570、9568 由端口 9569 加一减一后自动生成。
18	悦数图数据库	TCP	9889、9890、9888	集群间数据同步功能中 Drainer 服务端口。用于同步 Storage、Meta 数据给从集群。端口 9890、9888 由端口 9889 加一减一后自动生成。
19	NebulaGraph Studio	TCP	7001	Studio 提供 Web 服务占用端口。
20	悦数运维监控	TCP	8090	Nebula HTTP Gateway 依赖服务端口。为集群服务提供 HTTP 接口，执行 nGQL 语句与悦数图数据库数据库进行交互。
21	悦数运维监控	TCP	9200	Nebula Stats Exporter 依赖服务端口。收集集群的性能指标，包括服务 IP 地址、版本和监控指标（例如查询数量、查询延迟、心跳延迟 等）。
22	悦数运维监控	TCP	9100	

Copyright © Yueshu Graph Technology - All rights reserved.

序号	所属产品/服务	类型	默认端口	说明
				Node Exporter 依赖服务端口。收集集群中机器的资源信息，包括 CPU、内存、负载、磁盘和流量。
23	悦数运维监控	TCP	9091	Prometheus 服务的端口。存储监控数据的时间序列数据库。
24	NebulaGraph Dashboard	TCP	7003	Dashboard 社区版 提供 Web 服务占用端口。
25	悦数运维监控	TCP	7005	悦数运维监控提供 Web 服务占用端口。
26	悦数运维监控	TCP	9093	Alertmanager 服务的端口。接收 Prometheus 告警，发送告警通知给悦数运维监控。
27	悦数图探索	TCP	7002	悦数图探索提供的 Web 服务占用端口。
28	License Manager	TCP	9119	License Manager (LM) 服务的端口。LM 服务用于管理 License（仅在企业版集群中使用）。

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22.4 生态工具概览

Note

内核版本号命名规则为 `X.Y.Z`，表示 `大版本 X，中版本 Y，小版本 Z`。关于生命周期说明：

- 企业版：一个 `Y` 版本的开发周期为 6 个月，其维护与支持周期为 3 年；
- 社区版：一个 `Y` 版本的开发周期为 3 个月，其维护与支持周期至下一个 `Y` 版本发布；
- 文件命名中含有 `RC` 表示仅用于预览的非正式版本（`Release Candidate`），其维护周期仅至下个 RC 或者正式版本发布，其客户端、数据兼容性等均无保证。
- 文件命名中含有 `nightly`、`SNAPSHOT` 或者日期的为日常开发版本，无质量保证和维护周期。
- 版本号不一定连续，其数字大小仅代表偏序关系。

Compatibility

对于客户端的升级要求为：

- 内核从 `X.Y.Z1` 升级成 `X.Y.Z2`：表示内核完全前向兼容，通常用于 bugfix，建议尽快升级内核小版本。此时客户端可以不升级；
- 内核从 `X.Y1.*` 升级成 `X.Y2.*`：表示存在 API、语法、返回值部分不兼容，通常用于增加功能、提升性能、代码优化等。需要客户端相应升级至 `X.Y2.*`；
- 内核从 `X1.*.*` 升级成 `X2.*.*`：表示存储格式、API 、语法等存在大的不兼容。需要使用工具升级内核数据；客户端必须升级；
- 默认内核及客户端都不支持降级：不可从 `X.Y.Z2` 降级成 `X.Y.Z1`；
- 2.x 版本的生态工具完全不支持在悦数图数据库 3.x 版本中使用。

22.4.1 悦数运维监控

悦数运维监控是一款用于监控和管理悦数图数据库多集群中机器和服务状态的可视化工具，支持在可视化界面进行集群创建、集群导入、数据平衡、扩容缩容等操作。详情参见什么是悦数运维监控。

悦数图数据库版本	悦数运维监控版本
v3.5.1	v3.5.1

22.4.2 悦数图探索

悦数图探索是一款可以通过 Web 访问的图探索可视化工具，搭配悦数图数据库内核使用，用于与图数据进行可视化交互。即使没有图数据操作经验，用户也可以快速成为图专家。详情参见什么是悦数图探索。

悦数图数据库版本	悦数图探索版本
v3.5.1	v3.5.1

22.4.3 NebulaGraph Stats Exporter

nebula-stats-exporter将监控数据导入Prometheus。

悦数图数据库版本	**Stats Exporter** 版本
v3.5.1	v3.3.0

Copyright © Yueshu Graph Technology - All rights reserved.

22.4.4 NebulaGraph Exchange

NebulaGraph Exchange（简称 Exchange）是一款 Apache Spark™ 应用，用于在分布式环境中将集群中的数据批量迁移到悦数图数据库中，能支持多种不同格式的批式数据和流式数据的迁移。详情请参见什么是 NebulaGraph Exchange。

悦数图数据库版本	**Exchange** 版本	**Exchange** 企业版本
v3.5.1	v3.5.0	v3.5.0

22.4.5 NebulaGraph Operator

NebulaGraph Operator（简称 Operator）是用于在 Kubernetes 系统上自动化部署和运维悦数图数据库集群的工具。依托于 Kubernetes 扩展机制，悦数图数据库将其运维领域的知识全面注入至 Kubernetes 系统中，让悦数图数据库成为真正的云原生图数据库。详情请参考什么是 NebulaGraph Operator。

悦数图数据库版本	**Operator** 版本
v3.5.1	v1.6.2

22.4.6 NebulaGraph Importer

NebulaGraph Importer（简称 Importer）是一款悦数图数据库的 CSV 文件导入工具。Importer 可以读取本地的 CSV 文件，然后导入数据至悦数图数据库中。详情请参见什么是 NebulaGraph Importer。

悦数图数据库版本	**Importer** 版本
v3.5.1	v4.0.0

22.4.7 NebulaGraph Spark Connector

NebulaGraph Spark Connector 是一个 Spark 连接器，提供通过 Spark 标准形式读写悦数图数据库数据的能力。NebulaGraph Spark Connector 由 Reader 和 Writer 两部分组成。详情请参见什么是 NebulaGraph Spark Connector。

悦数图数据库版本	**Spark Connector** 版本
v3.5.1	v3.4.0

22.4.8 NebulaGraph Flink Connector

NebulaGraph Flink Connector 是一款帮助 Flink 用户快速访问悦数图数据库的连接器，支持从悦数图数据库中读取数据，或者将其他外部数据源读取的数据写入悦数图数据库。详情请参见什么是 NebulaGraph Flink Connector。

悦数图数据库版本	**Flink Connector** 版本
v3.5.1	v3.5.0

22.4.9 NebulaGraph Algorithm

NebulaGraph Algorithm（简称 Algorithm）是一款基于 GraphX 的 Spark 应用程序，通过提交 Spark 任务的形式使用完整的算法工具对悦数图数据库数据库中的数据执行图计算，也可以通过编程形式调用 lib 库下的算法针对 DataFrame 执行图计算。详情请参见什么是 NebulaGraph Algorithm。

悦数图数据库版本	**Algorithm** 版本
v3.5.1	v3.0.0

Copyright © Yueshu Graph Technology - All rights reserved.

22.4.10 悦数图计算

悦数图计算是一款集成了开源高性能图计算框架 Plato 的应用程序，支持利用 Plato 对悦数图数据库数据库中的数据执行图计算。详情请参见什么是悦数图计算。

悦数图数据库版本	悦数图计算版本
v3.5.1	v3.5.0

22.4.11 NebulaGraph Console

NebulaGraph Console 是悦数图数据库的原生 CLI 客户端。如何使用请参见 NebulaGraph Console。

悦数图数据库版本	**Console** 版本
v3.5.1	v3.5.0

22.4.12 Backup&Restore

Backup&Restore是一款命令行界面（CLI）工具。通过Backup&Restore，用户可以备份悦数图数据库企业版数据，并基于备份目录进行数据恢复。

悦数图数据库版本	**Backup&Restore**版本
v3.5.1	v3.5.1

22.4.13 NebulaGraph Bench

NebulaGraph Bench 用于测试悦数图数据库的基线性能数据，使用 LDBC v0.3.3 的标准数据集。

悦数图数据库版本	**Bench** 版本
v3.5.1	v1.2.0

22.4.14 API、SDK

Compatibility

选择与内核版本相同 `X.Y.*` 的最新版本。

悦数图数据库版本	语言（**commit id**）
v3.5.1	C++
v3.5.1	Go
v3.5.1	Python
v3.5.1	Java
v3.5.1	HTTP

22.4.15 未发布

- Rust Client
- Node.js Client
- Object Graph Mapping Library (OGM, or ORM)

Copyright © Yueshu Graph Technology - All rights reserved.

- 测试

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

22.5 错误码

悦数图数据库运行出现问题时，会返回错误码。本文介绍错误码的详细信息。

Copyright © Yueshu Graph Technology - All rights reserved.

Note

- 返回 `0` 表示执行成功。

Copyright © Yueshu Graph Technology - All rights reserved.

错误名称	错误码	说明
`E_DISCONNECTED`	`-1`	连接断开
`E_FAIL_TO_CONNECT`	`-2`	无法建立连接
`E_RPC_FAILURE`	`-3`	RPC 失败
`E_LEADER_CHANGED`	`-4`	Raft leader 变更
`E_SPACE_NOT_FOUND`	`-5`	图空间不存在
`E_TAG_NOT_FOUND`	`-6`	Tag 不存在
`E_EDGE_NOT_FOUND`	`-7`	Edge type不存在
`E_INDEX_NOT_FOUND`	`-8`	索引不存在
`E_EDGE_PROP_NOT_FOUND`	`-9`	边属性不存在
`E_TAG_PROP_NOT_FOUND`	`-10`	Tag 属性不存在
`E_ROLE_NOT_FOUND`	`-11`	当前角色不存在
`E_CONFIG_NOT_FOUND`	`-12`	当前配置不存在
`E_MACHINE_NOT_FOUND`	`-13`	当前主机不存在
`E_LISTENER_NOT_FOUND`	`-15`	listener 不存在
`E_PART_NOT_FOUND`	`-16`	当前分区不存在
`E_KEY_NOT_FOUND`	`-17`	key 不存在
`E_USER_NOT_FOUND`	`-18`	用户不存在
`E_STATS_NOT_FOUND`	`-19`	统计信息不存在
`E_SERVICE_NOT_FOUND`	`-20`	没有找到当前服务
`E_DRAINER_NOT_FOUND`	`-21`	drainer 不存在
`E_DRAINER_CLIENT_NOT_FOUND`	`-22`	drainer 客户端不存在
`E_PART_STOPPED`	`-23`	当前 partition 已经被停止
`E_BACKUP_FAILED`	`-24`	备份失败
`E_BACKUP_EMPTY_TABLE`	`-25`	备份的表为空
`E_BACKUP_TABLE_FAILED`	`-26`	备份表失败
`E_PARTIAL_RESULT`	`-27`	multiget 无法获得所有数据
`E_REBUILD_INDEX_FAILED`	`-28`	重建索引失败
`E_INVALID_PASSWORD`	`-29`	密码无效
`E_FAILED_GET_ABS_PATH`	`-30`	无法获得绝对路径
`E_BAD_USERNAME_PASSWORD`	`-1001`	身份验证失败
`E_SESSION_INVALID`	`-1002`	无效会话
`E_SESSION_TIMEOUT`	`-1003`	会话超时
`E_SYNTAX_ERROR`	`-1004`	语法错误
`E_EXECUTION_ERROR`	`-1005`	执行错误
`E_STATEMENT_EMPTY`	`-1006`	语句为空

Copyright © Yueshu Graph Technology - All rights reserved.

错误名称	错误码	说明
`E_BAD_PERMISSION`	`-1008`	权限不足
`E_SEMANTIC_ERROR`	`-1009`	语义错误
`E_TOO_MANY_CONNECTIONS`	`-1010`	超出最大连接数
`E_PARTIAL_SUCCEEDED`	`-1011`	访问存储失败（仅有部分请求成功）
`E_NO_HOSTS`	`-2001`	主机不存在
`E_EXISTED`	`-2002`	主机已经存在
`E_INVALID_HOST`	`-2003`	无效主机
`E_UNSUPPORTED`	`-2004`	当前命令、语句、功能不支持
`E_NOT_DROP`	`-2005`	不允许删除
`E_CONFIG_IMMUTABLE`	`-2007`	配置项不能改变
`E_CONFLICT`	`-2008`	参数与 meta 数据冲突
`E_INVALID_PARM`	`-2009`	无效的参数
`E_WRONGCLUSTER`	`-2010`	错误的集群
`E_ZONE_NOT_ENOUGH`	`-2011`	listener 冲突
`E_ZONE_IS_EMPTY`	`-2012`	主机不存在
`E_SCHEMA_NAME_EXISTS`	`-2013`	Schema 名字已存在
`E_RELATED_INDEX_EXISTS`	`-2014`	与 Tag 或 Edge Type 相关的索引存在，不能被删除
`E_RELATED_SPACE_EXISTS`	`-2015`	仍有图空间在主机上，不能被删除
`E_STORE_FAILURE`	`-2021`	存储数据失败
`E_STORE_SEGMENT_ILLEGAL`	`-2022`	存储段非法
`E_BAD_BALANCE_PLAN`	`-2023`	无效的数据均衡计划
`E_BALANCED`	`-2024`	集群已经处于数据均衡状态
`E_NO_RUNNING_BALANCE_PLAN`	`-2025`	没有正在运行的数据均衡计划
`E_NO_VALID_HOST`	`-2026`	缺少有效的主机
`E_CORRUPTED_BALANCE_PLAN`	`-2027`	已经损坏的数据均衡计划
`E_IMPROPER_ROLE`	`-2030`	回收用户角色失败
`E_INVALID_PARTITION_NUM`	`-2031`	无效的分区数量
`E_INVALID_REPLICA_FACTOR`	`-2032`	无效的副本因子
`E_INVALID_CHARSET`	`-2033`	无效的字符集
`E_INVALID_COLLATE`	`-2034`	无效的字符排序规则
`E_CHARSET_COLLATE_NOT_MATCH`	`-2035`	字符集和字符排序规则不匹配
`E_SNAPSHOT_FAILURE`	`-2040`	生成快照失败
`E_BLOCK_WRITE_FAILURE`	`-2041`	写入块数据失败
`E_ADD_JOB_FAILURE`	`-2044`	增加新的任务失败
`E_STOP_JOB_FAILURE`	`-2045`	停止任务失败

Copyright © Yueshu Graph Technology - All rights reserved.

错误名称	错误码	说明
E_SAVE_JOB_FAILURE	-2046	保存任务信息失败
E_BALANCER_FAILURE	-2047	数据均衡失败
E_JOB_NOT_FINISHED	-2048	当前任务还没有完成
E_TASK_REPORT_OUT_DATE	-2049	任务报表失效
E_JOB_NOT_IN_SPACE	-2050	当前任务不在图空间内
E_JOB_NEED_RECOVER	-2051	当前任务需要恢复
E_JOB_ALREADY_FINISH	-2052	任务已经失败或完成
E_JOB_SUBMITTED	-2053	任务默认状态
E_JOB_NOT_STOPPABLE	-2054	给定任务不支持停止
E_JOB_HAS_NO_TARGET_STORAGE	-2055	leader 分布未上报，因此无法将任务发送到存储
E_INVALID_JOB	-2065	无效的任务
E_BACKUP_BUILDING_INDEX	-2066	备份终止（正在创建索引）
E_BACKUP_SPACE_NOT_FOUND	-2067	备份时图空间不存在
E_RESTORE_FAILURE	-2068	备份恢复失败
E_SESSION_NOT_FOUND	-2069	会话不存在
E_LIST_CLUSTER_FAILURE	-2070	获取集群信息失败
E_LIST_CLUSTER_GET_ABS_PATH_FAILURE	-2071	获取集群信息时无法获取绝对路径
E_LIST_CLUSTER_NO_AGENT_FAILURE	-2072	获取集群信息时无法获得 agent
E_QUERY_NOT_FOUND	-2073	query 未找到
E_AGENT_HB_FAILURE	-2074	agent 没有汇报心跳
E_HOST_CAN_NOT_BE_ADDED	-2082	该主机不能被添加，因为它不是一个 Storage 主机
E_ACCESS_ES_FAILURE	-2090	访问 Elasticsearch 失败
E_GRAPH_MEMORY_EXCEEDED	-2600	Graph 内存超出
E_CONSENSUS_ERROR	-3001	选举时无法达成共识
E_KEY_HAS_EXISTS	-3002	key 已经存在
E_DATA_TYPE_MISMATCH	-3003	数据类型不匹配
E_INVALID_FIELD_VALUE	-3004	无效的字段值
E_INVALID_OPERATION	-3005	无效的操作
E_NOT_NULLABLE	-3006	当前值不允许为空
E_FIELD_UNSET	-3007	字段非空或者没有默认值时，字段值必须设置
E_OUT_OF_RANGE	-3008	取值超出了当前类型的范围
E_DATA_CONFLICT_ERROR	-3010	数据冲突
E_WRITE_STALLED	-3011	写入被延迟
E_IMPROPER_DATA_TYPE	-3021	不正确的数据类型
E_INVALID_SPACEVIDLEN	-3022	VID 长度无效

Copyright © Yueshu Graph Technology - All rights reserved.

错误名称	错误码	说明
`E_INVALID_FILTER`	`-3031`	无效的过滤器
`E_INVALID_UPDATER`	`-3032`	无效的字段更新
`E_INVALID_STORE`	`-3033`	无效的 KV 存储
`E_INVALID_PEER`	`-3034`	peer 无效
`E_RETRY_EXHAUSTED`	`-3035`	重试次数耗光
`E_TRANSFER_LEADER_FAILED`	`-3036`	leader 转换失败
`E_INVALID_STAT_TYPE`	`-3037`	无效的统计类型
`E_INVALID_VID`	`-3038`	VID 无效
`E_LOAD_META_FAILED`	`-3040`	加载元信息失败
`E_FAILED_TO_CHECKPOINT`	`-3041`	生成 checkpoint 失败
`E_CHECKPOINT_BLOCKED`	`-3042`	生成 checkpoint 被阻塞
`E_FILTER_OUT`	`-3043`	数据被过滤
`E_INVALID_DATA`	`-3044`	无效的数据
`E_MUTATE_EDGE_CONFLICT`	`-3045`	并发写入同一条边发生冲突
`E_MUTATE_TAG_CONFLICT`	`-3046`	并发写入同一个点发生冲突
`E_OUTDATED_LOCK`	`-3047`	锁已经失效
`E_INVALID_TASK_PARA`	`-3051`	无效的任务参数
`E_USER_CANCEL`	`-3052`	用户取消了任务
`E_TASK_EXECUTION_FAILED`	`-3053`	任务执行失败
`E_PLAN_IS_KILLED`	`-3060`	执行计划被清除
`E_NO_TERM`	`-3070`	收到请求时心跳流程未完成
`E_OUTDATED_TERM`	`-3071`	收到旧 leader 的过时心跳（已选举出新的 leader）
`E_WRITE_WRITE_CONFLICT`	`-3073`	并发写入时与后到的请求发生冲突
`E_RAFT_UNKNOWN_PART`	`-3500`	未知的分区
`E_RAFT_LOG_GAP`	`-3501`	raft 日志落后
`E_RAFT_LOG_STALE`	`-3502`	raft 日志过期
`E_RAFT_TERM_OUT_OF_DATE`	`-3503`	心跳信息已经过期
`E_RAFT_UNKNOWN_APPEND_LOG`	`-3504`	未知的追加日志
`E_RAFT_WAITING_SNAPSHOT`	`-3511`	等待快照完成
`E_RAFT_SENDING_SNAPSHOT`	`-3512`	发送快照过程出错
`E_RAFT_INVALID_PEER`	`-3513`	无效的接收端
`E_RAFT_NOT_READY`	`-3514`	Raft 没有启动
`E_RAFT_STOPPED`	`-3515`	Raft 已经停止
`E_RAFT_BAD_ROLE`	`-3516`	错误的角色
`E_RAFT_WAL_FAIL`	`-3521`	写入 WAL 失败

Copyright © Yueshu Graph Technology - All rights reserved.

错误名称	错误码	说明
`E_RAFT_HOST_STOPPED`	`-3522`	主机已经停止
`E_RAFT_TOO_MANY_REQUESTS`	`-3523`	请求数量过多
`E_RAFT_PERSIST_SNAPSHOT_FAILED`	`-3524`	持久化快照失败
`E_RAFT_RPC_EXCEPTION`	`-3525`	RPC 异常
`E_RAFT_NO_WAL_FOUND`	`-3526`	没有发现 WAL 日志
`E_RAFT_HOST_PAUSED`	`-3527`	主机暂停
`E_RAFT_WRITE_BLOCKED`	`-3528`	写入被堵塞
`E_RAFT_BUFFER_OVERFLOW`	`-3529`	缓存溢出
`E_RAFT_ATOMIC_OP_FAILED`	`-3530`	原子操作失败
`E_LEADER_LEASE_FAILED`	`-3531`	leader 租约过期
`E_RAFT_CAUGHT_UP`	`-3532`	Raft 已经同步数据
`E_STORAGE_MEMORY_EXCEEDED`	`-3600`	Storage 内存超出
`E_LOG_GAP`	`-4001`	drainer 日志落后
`E_LOG_STALE`	`-4002`	drainer 日志过期
`E_INVALID_DRAINER_STORE`	`-4003`	drainer 数据存储无效
`E_SPACE_MISMATCH`	`-4004`	图空间不匹配
`E_PART_MISMATCH`	`-4005`	分区不匹配
`E_DATA_CONFLICT`	`-4006`	数据冲突
`E_REQ_CONFLICT`	`-4007`	请求冲突
`E_DATA_ILLEGAL`	`-4008`	数据非法
`E_CACHE_CONFIG_ERROR`	`-5001`	缓存配置错误
`E_NOT_ENOUGH_SPACE`	`-5002`	空间不足
`E_CACHE_MISS`	`-5003`	没有命中缓存
`E_POOL_NOT_FOUND`	`-5005`	写缓存失败
`E_NODE_NUMBER_EXCEED_LIMIT`	`-7001`	机器节点数超出限制
`E_PARSING_LICENSE_FAILURE`	`-7002`	解析证书失败
`E_UNKNOWN`	`-8000`	未知错误

最后更新: October 7, 2023

Copyright © Yueshu Graph Technology - All rights reserved.

https://ent-docs.nebula-graph.com.cn/3.5.1

Copyright © Yueshu Graph Technology - All rights reserved.