存储格式

- 1.0存储格式：https://nebula-graph.com.cn/posts/nebula-graph-storage-engine-overview/
- 2.0存储格式：https://blog.csdn.net/weixin_44324814/article/details/114631038
- 存储格式
    - Key里有PartID原因：用于 **Partition 重新分布(balance) 时方便根据前缀扫描整个 Partition 数据**

代码结构

代码结构及模块说明

Nebula Graph

- https://nebula-graph.com.cn/posts/nebula-graph-source-code-reading-01/

```
├── cmake
├── conf
├── LICENSES
├── package
├── resources
├── scripts
├── src
│   ├── context
│   ├── daemons
│   ├── executor
│   ├── optimizer
│   ├── parser
│   ├── planner
│   ├── scheduler
│   ├── service
│   ├── session
│   ├── stats
│   ├── util
│   ├── validator
│   └── visitor
└── tests
    ├── admin
    ├── bench
    ├── common
    ├── data
    ├── job
    ├── maintain
    ├── mutate
    ├── query
    └── tck
```

- conf/：查询引擎配置文件目录
- package/：graph 打包脚本

- resources/：资源文件
- scripts/：启动脚本
- src/：查询引擎源码目录
    - src/context/：查询的上下文信息，包括 AST（抽象语法树），Execution Plan（执行计划），执行结果以及其他计算相关的资源。
    - src/daemons/：查询引擎主进程
    - src/executor/：执行器，各个算子的实现
    - src/optimizer/：RBO（基于规则的优化）实现，以及优化规则
    - src/parser/：词法解析，语法解析，：AST结构定义
    - src/planner/：算子，以及执行计划生成
    - src/scheduler/：执行计划的调度器
    - src/service/：查询引擎服务层，提供鉴权，执行 Query 的接口
    - src/session/：Session 管理
    - src/stats/：执行统计，比如 P99、慢查询统计等
    - src/util/：工具函数
    - src/validator/：语义分析实现，用于检查语义错误，并进行一些简单的改写优化
    - src/visitor/：表达式访问器，用于提取表达式信息，或者优化
- tests/：基于 BDD 的集成测试框架，测试所有 Nebula Graph 提供的功能

Nebula Storage

```
├── cmake
├── conf
├── docker
├── docs
├── LICENSES
├── package
├── scripts
└── src
    ├── codec
    ├── daemons
    ├── kvstore
    ├── meta
    ├── mock
    ├── storage
    ├── tools
    ├── utils
    └── version
```

- conf/：存储引擎配置文件目录
- package/：storage 打包脚本
- scripts/：启动脚本
- src/：存储引擎源码目录
    - src/codec/：序列化反序列化工具
    - src/daemons/：存储引擎和元数据引擎主进程
    - src/kvstore/：基于 raft 的分布式 KV 存储实现
    - src/meta/：基于 KVStore 的元数据管理服务实现，用于管理元数据信息，集群管理，长耗时任务管理等
    - src/storage/：基于 KVStore 的图数据存储引擎实现
    - src/tools/：一些小工具实现
    - src/utils/：代码工具函数

Nebula Common

```
├── cmake
│   └── nebula
├── LICENSES
├── src
│   └── common
│       ├── algorithm
│       ├── base
│       ├── charset
│       ├── clients
│       ├── concurrent
│       ├── conf
│       ├── context
│       ├── cpp
│       ├── datatypes
│       ├── encryption
│       ├── expression
│       ├── fs
│       ├── function
│       ├── graph
│       ├── hdfs
│       ├── http
│       ├── interface
│       ├── meta
│       ├── network
│       ├── plugin
│       ├── process
│       ├── session
│       ├── stats
│       ├── test
│       ├── thread
│       ├── thrift
│       ├── time
│       ├── version
│       └── webservice
└── third-party
```

Nebula Common 仓库代码是 Nebula 内核代码的工具包，提供一些常用工具的高效实现。一些常用工具包相信各位工程师一定也是了然于心。这里只对其中和图数据库密切相关的目录进行说明。

- src/common/clients/：meta，storage 客户端的 CPP 实现
- src/common/datatypes/：Nebula Graph 中数据类型及计算的定义，比如 string，int，bool，float，Vertex，Edge 等。
- rc/common/expression/：nGQL 中表达式的定义
- src/common/function/：nGQL 中的函数的定义
- src/common/interface/：graph、meta、storage 服务的接口定义

编译&开启服务

```
mkdir build
cd build

cmake -DENABLE_BUILD_STORAGE=on -DENABLE_TESTING=OFF -DCMAKE_BUILD_TYPE=Debug -
DNEBULA_COMMON_REPO_TAG=v2.0.0 -DNEBULA_STORAGE_REPO_TAG=v2.0.0 ..

make -j8

sudo make install-all

sudo /usr/local/nebula/scripts/nebula.service start all

sudo /home/yundao/project/nebulagraph/nebula-console -port 9669 -u a -p b
```

执行图操作

- 官方文档（快速入门）：https://docs.nebula-graph.com.cn/2.0/2.quick-start/4.nebula-graph-crud/

```
CREATE SPACE IF NOT EXISTS test_space
USE test_space

CREATE TAG player(name string, age int);
CREATE EDGE follow(degree int);

INSERT VERTEX player(name, age) VALUES "p1":("Tim Duncan", 42);
INSERT VERTEX player(name, age) VALUES "p2":("Tony Parker", 36);
INSERT EDGE follow(degree) VALUES "p1" -> "p2":(95);

GO FROM "p1" OVER follow;
FETCH PROP ON player "player100"; //查询VID为player100的球员的属性。

// rdf
CREATE TAG rdfplay(name string, age int);
INSERT RDFVERTEX (123) rdfplayer(name, age) VALUES "p1":("Tim Duncan", 42),"p2":
("Tom", 21);
```