

Table of Contents

介绍	1.1
存储结构	1.2

neo4j-guide

• 下载

<https://neo4j.com/download/>

• 文档

<https://neo4j.com/docs/>

• 安装

解压

```
cd /neo4j-community-3.2.2/bin
(sudo) ./neo4j start
./neo4j stop
./neo4j status
./cypher-shell    (进入命令行)
username:neo4j
password:neo4j
(退出)    :exit
控制台http://127.0.0.1:7474 (在此修改初始密码)
允许远程连接: vim /neo4j-community-3.2.2/conf/neo4j.conf
dbms.connectors![:default_listen_address=0.0.0.0(去掉注释)
```

• 优势

- 自带一套易于学习的查询语言（名为Cypher）
- 不使用schema，因此可以满足你的任何形式的需求
- 与关系型数据库相比，对于高度关联的数据（图形数据）的查询快速要快上许多
- 它的实体与关系结构非常自然地切合人类的直观感受
- 支持兼容ACID的事务操作
- 提供了一个高可用性模型，以支持大规模数据量的查询，支持备份、数据局部性以及冗余
- 提供了一个可视化的查询控制台，你不会对它感到厌倦的

Neo4j 社区版和企业版的技术特性区别（版本 3.x）

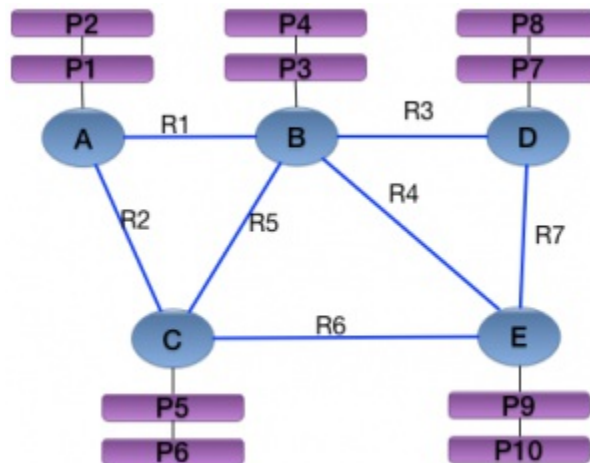
特性	企业版特性	社区版
数据库集群	多数据库集群可以提供横向的可扩展性，以支持大规模并发的数据写入和查询，以及高可用性和灾难恢复。	无
数据库热备份	在数据库运行时进行备份，确保 7x24 的可用性。	手工备份，需要停止数据库服务
在线存储空间再使用	回收并再使用由于数据删除而释放的空间。	无
LDAP/AD 集成	与企业级用户身份认证和访问控制系统集成。	无
基于数据库角色的访问控制	为数据库管理员提供指定用户角色的功能，支持的角色包括只读、写入、管理员等。	无
高级数据库监控，性能指标和 Cypher 查询状态跟踪	提供监控数据库运行状况的整套指标，以及对正在运行的查询进行状态跟踪和管理(例如强制结束查询的运行)。	无
JMX 日志和监控	包含 Graphite JMX 客户端	无
更全面的日志	包括 HTTP, GC, Security 日志	仅 HTTP 日志
更快的运行时刻 Cypher 执行	企业版的运行内核进行专门的优化，这使得 Cypher 查询的执行更加高效。	无
无限制的图节点、关系和属性	对节点、关系和属性的数目没有限制。	节点：344 亿 关系：344 亿
节点键	数据库模式特性：使用一个或多个节点属性来唯一标识一个节点(类似关系数据库中的复合主键)。	无
属性存在性限制	数据库模式特性：允许定义限制(Constraint)要求某个属性总是存在	无
更高效的写入锁	对于超过 4 个 CPU 核心的系统提供更加高效的写入锁机制以增加并发处理能力。	只针对最多 4 个 CPU 核心
极大优化的并行图算法执行过程	对于超过 4 个 CPU 核心的系统提供高度优化的并行图算法执行过程。	只针对最多 4 个 CPU 核心
支持更先进和广泛的硬件类型	支持超过 4 个以上的 CPU 核心； 支持 IBM Power 8 系统主机	只针对最多 4 个 CPU 核心
企业级运营管理员工具	提供管理员命令行界面，包括 neo4j-shell、neo4j-import(高速数据导入)等	无

<http://blog.csdn.net/GraphWay>

• 数据模型

Neo4j是图数据库的一种，使用图模型存储数据。有三大要素：Node,Relationship,Property

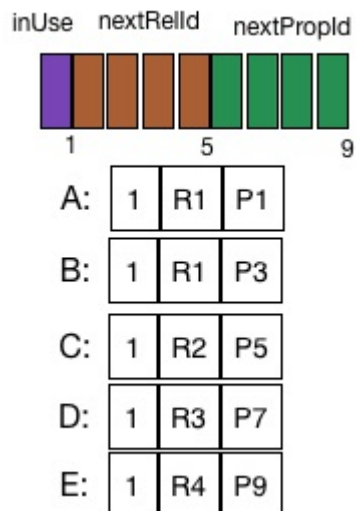
- Node即实体，数据的主要载体，用Property描述
- Relationship表示Node之间的关联关系
- Property用以增加Node和Relationship的描述信息
- Node和Relationship都有“类别”的概念



- Relationship是有方向的

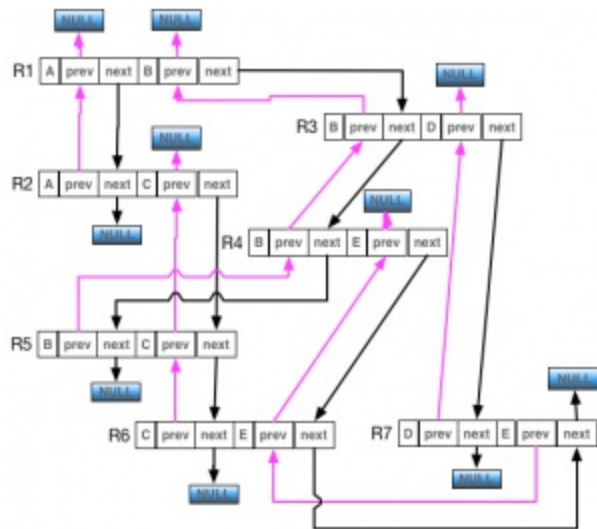
• Node存储结构

Node只存储只存储它第一个Property和它的第一个Relationship的指针。Property比较简单，用一个单向链表来存储，遍历这个链表即可找到Node的所有Property(Relationship的Property也是同样的存储结构)。所以，对于同一类型Node，较多被访问的Property应该被放在链表的前面（在时间上先被添加到Node）。



• Relationship存储结构

存储Relationship是一个双向链表。实际上，一个Relationship肯定是连接两个Node的，所以，它会存储2组指针，分别指向2个Node的Relationship。如下图所示，R1是A和B的Relationship，首先它会存储2个Node(A&B)，然后A组的"next"指向与A相连的下一个Relationship(R2)，B组的"next"指向与B相连的下一个



Relationship。"prev"同理。

• graph.db存储文件

Neo4j的数据文件存放在\$NEO4J_HOME/data/databases/graph.db下面

◦ 存储 **node** 的文件

■ 存储节点数据及其序列Id

neostore.nodestore.db: 存储节点数组，数组的下标即是该节点的ID neostore.nodestore.db.id : 存储最大的ID 及已经free的ID

■ 存储节点label及其序列Id

neostore.nodestore.db.labels : 存储节点label数组数据，数组的下标即是该节点label的ID
neostore.nodestore.db.labels.id

◦ 存储 **relationship** 的文件

■ 存储关系数据及其序列Id

neostore.relationshipstore.db 存储关系 record 数组数据 neostore.relationshipstore.db.id

■ 存储关系组数据及其序列Id

neostore.relationshipgroupstore.db 存储关系 group数组数据 neostore.relationshipgroupstore.db.id

■ 存储关系类型及其序列Id

neostore.relationshiptypestore.db 存储关系类型数组数据 neostore.relationshiptypestore.db.id

■ 存储关系类型的名称及其序列Id

neostore.relationshiptypestore.db.names存储关系类型 token 数组数据
neostore.relationshiptypestore.db.names.id

◦ 存储 **label** 的文件

■ 存储label token数据及其序列Id

neostore.labeltokenstore.db 存储label token 数组数据 neostore.labeltokenstore.db.id

- 存储label token名字数据及其序列Id

neostore.labeltokenstore.db.names 存储 label token 的 names 数据
neostore.labeltokenstore.db.names.id

◦ 存储 **property** 的文件

- 存储属性数据及其序列Id

neostore.propertystore.db 存储 property 数据 neostore.propertystore.db.id

- 存储属性数据中的数组类型数据及其序列Id

neostore.propertystore.db.arrays 存储 property (key-value 结构)的Value值是数组的数据。
neostore.propertystore.db.arrays.id

- 属性数据为长字符串类型的存储文件及其序列Id

neostore.propertystore.db.strings 存储 property (key-value 结构)的Value值是字符串的数据。
neostore.propertystore.db.strings.id

- 属性数据的索引数据文件及其序列Id

neostore.propertystore.db.index 存储 property (key-value 结构)的key 的索引数据。
neostore.propertystore.db.index.id

- 属性数据的键值数据存储文件及其序列Id

neostore.propertystore.db.index.keys 存储 property (key-value 结构)的key 的字符串值。
neostore.propertystore.db.index.keys.id

◦ 其他的文件

- 存储版本信息

neostore neostore.id

- 存储 schema 数据

neostore.schemastore.db neostore.schemastore.db.id

- 活动的逻辑日志

nioneo_logical.log.active

- 记录当前活动的日志文件名称

active_tx_log