# 2019 开源数据架构平台的产品订阅服务及组织协同实施 PPT提纲

# （MongoDB + Neo4j医疗卫生领域实践）



### 总起（10-15页）

#### 医疗卫生领域背景介绍

#### 卫生领域参考模型

#### 项目介绍：

##### His/Lis/CDS临床数据仓库等系统介绍

##### 切入点1：电子病历？——传统数据库在半结构化数据存储面临的问题（也就是之前是怎么做的）——引出Mongodb解决方案

##### 切入点2：不良反应发现的业务及需求——引出Neo4j解决方案

### MongoDB解决大量半结构化数据存储的实践（10-20页）

#### 相关业务及面对问题详述

#### 文档型数据库简介及意义——为什么我们要用mongo解决上述问题

#### 使用MongoDB的解决方案

#### MongoDB产品介绍

##### 行业地位

##### 同类产品比较

##### 技术架构

##### 产品生态

#### MognoDB项目实践

##### 部署方案（最佳实践）

##### 基于项目的优化和改造

###### 索引优化

###### 存储引擎优化

###### Etc…

### Neo4j在不良反应发现系统的实践（10-20页）

#### 不良反应发现的需求

#### 不良反应发现的业务设计

##### 建模设计

##### 数据流程

##### 最终形态

#### 承转——上述设计使用Neo4j落地

#### Neo4j产品介绍

##### Neo4j的总体架构

1. **Neo4j图数据库基础**
2. **图数据库介绍**

图数据库源起欧拉和图理论，也可称为面向/基于图的数据库，对应的英文是Graph Database。图数据库的基本含义是以“图”这种数据结构存储和查询数据，而不是存储图片的数据库。它的数据模型主要是以节点和关系（边）来体现，也可处理键值对。它的优点是快速解决复杂的关系问题。

说得正式一些，图可以说是顶点和边的集合，或者说更简单一点儿，图就是一些节点（node）和关联这些节点的联系（relationship）的集合。图将实体表现为节点，实体与其他实体连接的方式表现为关系。Nodes 和 Relationships 包含key/value形式的属性。Nodes通过Relationships所定义的关系相连起来，形成关系型网络结构。如图1：

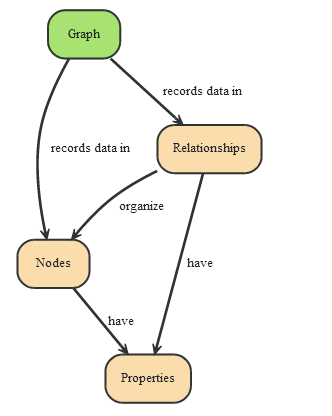


图1

1. **Neo4j概述**

Neo4j是由Java实现的开源NoSQL图数据库，是世界领先的开源图形数据库。 它完全由使用Java语言通过新技术的发展。Neo4j实现了专业数据库级别的图数据模型的存储。与普通的图处理或内存级数据库不同，Neo4j提供了完整的数据库特性，包括ACID事务的支持、集群支持、备份与故障转移等，这使其适合于企业级生产环境下的各种应用。

另外，Neo4j一些特殊功能使得Neo4j在用户、开发人员和DBA中非常受欢迎，如图2：

* **一个本地化的图数据库：**Neo4j自底向上构建成一个图数据库，它的体系结构旨在优化快速管理，存储和遍历节点和关系。在Neo4j中，关系是数据库中最重要的元素，他代表节点之间的相互联系。众所周知，在关系数据库领域中，“关系”用于多个不同表之间的连续操作，这种操作的性能下降与关系的数量呈指数级别，但在Neo4j中测试用于从一个节点指向另一个节点，其性能却是线性级别的。
* **界面友好：**Neo4j提供了一个查询与展示一体化的Web操作界面，对于图数据模型Neo4j使用D3.js做数据可视化，非常形象的展示了数据模型的节点和关系。
* **声明式图查询语言：**Cypher是一种声明式图数据库查询语言，它表现力丰富，查询效率高，其地位和作用与关系型数据库中的SQL语言类似。Cypher还有良好的扩展性，用户可以定制自己的查询方式（如自定义过程）。
* **ACID事务：**Neo4j通过ACID事务提供真正的数据安全，Neo4j使用事务来保证数据在硬件故障或系统崩溃的情况下不会丢失。
* **高性能：**Neo4j使用多副本主从复制的方式构建高可靠性集群，支持大数据集合并且可以不断扩展其容量，可存储数百万亿个实体。也就是说，Neo4j可部署在一个可容错、可扩展的集群上。此外Neo4j还提供了热备份和性能监控功能。
* **代码开源：**Neo4j将源代码公布到GitHub，任何人都可以去Neo4j的GitHub主页下下载源代码。

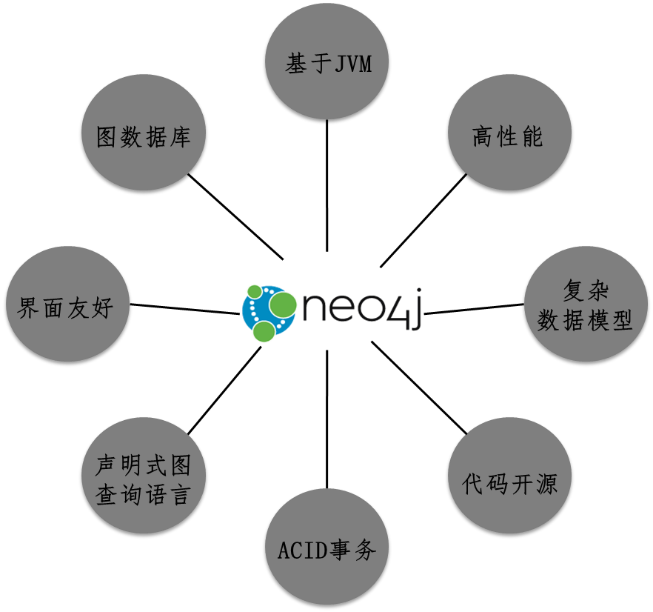


图2

1. **Neo4j图数据中基本元素与概念**
2. **节点**

节点（Node）是图数据库中的一个基本元素，用以表示一个实体记录，就像关系型数据库中的一条记录一样，在Neo4j中节点可以包含多个属性（Property）和多个标签（Label），如图3所示。



图3

1. **关系**

关系（Relationship）同样是图数据库中的基本元素，当数据库中已经存在节点后，需要将节点连接起来构成图。关系就是用来连接两个节点。关系也称为图论的边（Edge），其始端和末端都必须是节点，关系不能指向空也不能从空发起。关系和节点一样可以包含多个属性，但关系只能有一个类型（Type），如图4所示。一个节点可以被多个关系指向或作为关系的起始点，如图5所示，多个关系指向同一节点。

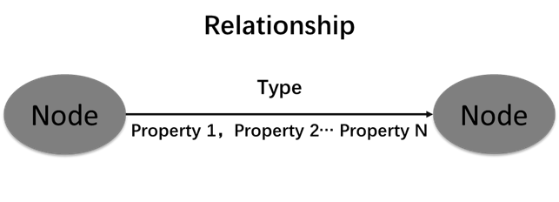
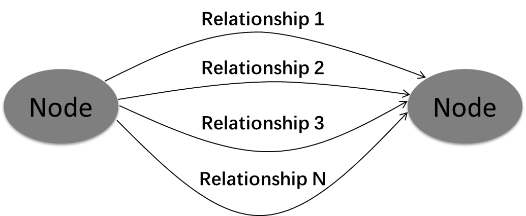


图5

图4

在图的遍历操作中我们可以指定关系遍历的反向或者指定为无方向，因此在创建关系的时候不必为两个节点创建相互指向的关系，而是在遍历时不指定遍历方向即可。

**\*\* 特别注意：一个节点可以存在指向自己的关系。如图6：**

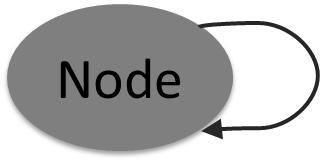
****

图6

1. **属性**

上面提到节点和关系都可以有多个属性。属性是由键值对组成的，就像Java的哈希表一样，属性名类似变量名，属性值类似变量值。属性值可以是基本额数据类型，或者由基本书库类型组成的数组。

需要注意的是属性值没有null的概念，如果一个属性不要了可以直接将整个键值对都移除，在使用Cypher或Java API时，可用IS NULL关键字判断属性是否存在。

1. **路径**

当使用节点和关系创建了一个图后，在此图中任意两个节点间都是可能存在路径的，路径也有长度的概念，也就是路径中关系的条数。

如果两个简单的节点，中间只存在一条关系，那这条路径的长度就是1.

1. **遍历**

遍历一张图就是按照一定的规则，根据它们之间的关系，依次访问所有关联的节点的操作。

对于遍历操作不必自己实现，因为Neo4j提供了一套高效的遍历API，可以指定便利规则，然后让Neo4j自动按照遍历规则遍历并返回遍历的结果。遍历规则可以是广度优先，也可以是深度优先。

1. **Neo4j 组成之browser**

和大多数数据库一样,neo4j是server-client的数据库,支持http和bolt 2种协议。neo4j自带一个基于浏览器的客户端，启动neo4j服务后，只需在浏览器输入server\_ip:设定端口，即可使用.。neo4j browser自带教程和示例数据库初始化脚本。方便使用。常用的命令：

* + - **:help** help命令显示各种帮助提示
    - **:play** 交互式学习命令. 例如，**:play movie graph** 进入基于电影数据库的教程.
    - **:param** 设置变量，**:param usrname => "zhangsan"**, 设置变量之后可以使用变量MATCH (n:Person) WHERE n.name = $usrname
    - **:params** 显示当前已经设置的所有变量. 也可以使用**:params {name: 'Stella', age: 24}** 覆盖目前的变量，但是这个命令没用类型安全。

1. **Neo4j组成之Cypher**
2. **Cypher简介**

Cypher是一种声明式图数据库查询语言，它具有丰富的表现力，能高效地查询和更新图数据。Cyper通过一系列不同的方法和建立于确定的实践为表达查询而激发的。许多关键字如like和order by是受SQL的启发。模式匹配的表达式来自于SPARQL。正则表达式匹配实现实用Scala programming language语言。

它的焦点在于从图中如何找回（what to retrieve），而不是怎么去做。这使得在不对用户公布的实现细节里关心的是怎么优化查询。

Neo4j使用Cypher查询图形数据，和SQL很相似，Cypher语言的关键字不区分大小写，但是属性值，标签，关系类型和变量是区分大小写的。

1. **Cypher组成**

这个查询语言包含以下几个明显的部分：

* START：在图中的开始点，通过元素的ID或所以查找获得。
* MATCH：图形的匹配模式，束缚于开始点。
* WHERE：过滤条件。
* RETURN：返回所需要的

##### Neo4j的细节架构

1. **Neo4j 数据库管理相关管理工具**
2. **导入工具**

* **简单导入示例：**

以电影、演员和角色图数据集为例，先将**path\_to\_target\_directory**修改为数据库文件目录，默认安装下**path\_to\_target\_directory**指向**<neo4j-home>/data/databases/graph.db**文件。其中电影节点文件为**movies.csv**，每部电影都有一个编号id，便于外部数据源的引用，另外每部电影都有电影名和年份属性，并为每个节点添加了**Movie**和**Sequel**标签。该文件内容示例如下：

movieId:ID,title,year:int,:LABEL

tt0133093,"The Matrix",1999,Movie

tt0234215,"The Matrix Reloaded",2003,Movie;Sequel

tt0242653,"The Matrix Revolutions",2003,Movie;Sequel

演员节点文件为**actors.csv**，每个演员有两个属性：编号（姓名的缩写）、姓名，并带有Actor标签，该文件内容示例如下：

personId:ID,name,:LABEL

keanu,"Keanu Reeves",Actor

laurence,"Laurence Fishburne",Actor

carrieanne,"Carrie-Anne Moss",Actor

角色文件为roles.csv，保存演员与电影之间的关系，START\_ID为演员节点中的编号，END\_ID为电影中的编号，role字段为该演员在这部电影中所扮演的角色名，TYPE字段为关系类型（在本例中为ACTED\_IN）。该文件内容示例如下：

:START\_ID,role,:END\_ID,:TYPE

keanu,"Neo",tt0133093,ACTED\_IN

keanu,"Neo",tt0234215,ACTED\_IN

keanu,"Neo",tt0242653,ACTED\_IN

laurence,"Morpheus",tt0133093,ACTED\_IN

laurence,"Morpheus",tt0234215,ACTED\_IN

laurence,"Morpheus",tt0242653,ACTED\_IN

carrieanne,"Trinity",tt0133093,ACTED\_IN

carrieanne,"Trinity",tt0234215,ACTED\_IN

carrieanne,"Trinity",tt0242653,ACTED\_IN

接下来调用数据导入neo4j-import命令如下：

neo4j\_home**$ ./**bin**/**neo4j-import **--**into path\_to\_target\_directory **--**nodes movies.csv **--**nodes actors.csv **--**relationships roles.csv

完成后则可以启动数据库，命令如下：

neo4j\_home**$ ./**bin**/**neo4j start

在Unix/Linux/OSX操作系统中，导入工具命令名为neo4j-import。与安装配置相关，导入工具可能在任意路径下调用，可能仅在安装目录中执行。而Windows系统则在安装目录下执行bin\neo4j-import命令即可。该命令有如下参数：

* --into <store-dir>：要导入到的数据库目录，该目录下不能包含已有数据库。
* --nodes[:Label1:Label2] "<file1>,<file2>,…​"：节点CSV标题和数据，如果带有多个文件将在逻辑上视为一个大文件。第一行必须为标题，每个数据源都有自己的标题。注意，一组文件必须用引号括起来。
* --relationships[:RELATIONSHIP\_TYPE] "<file1>,<file2>,…​"：关系CSV标题和数据。同样，多个文件将在逻辑上视为一个大文件，第一行必须为标题，每个数据源都有自己的标题，一组文件必须用引号括起来。
* --multiline-fields <true/false>：来自输入字段是否可以跨行，即是否包含换行符，默认不包含，即值为：false。
* --input-encoding <character set>：输入数据的编码字符集，必须是JVM中的可用字符集，如果未指定编码，则使用JVM的缺省字符集。
* --id-type <id-type>：指定节点/关系中编号字段的数据类型，可以为：STRING、INTEGER、ACTUAL。STRING：为字符串；INTEGER：为整数值；ACTUAL：以实际节点标识为其类型。默认值为：STRING。
* --stacktrace <true/false>：是否启用打印错误堆栈跟踪信息。
* --skip-bad-relationships <true/false>：导入是否跳过缺失节点编号的关系，即关系中未指定开始或结束节点编号，默认跳过，值为true。
* --db-config <path/to/neo4j.conf>：（高级选项）指定数据库特定配置的文件路径。与导入工具相关的配置参数有三个，分别为：dbms.relationship\_grouping\_threshold、unsupported.dbms.block\_size.strings、unsupported.dbms.block\_size.array\_properties。

**注：导入时不需要创建索引，导入完成后再添加。如果无法使用此工具进行数据导入，且加载的CSV文件为中小型，可以使用LOAD CSV方式。**

1. **Cypher Shell**

Cypher Shell是Neo4j数据库的一个命令行工具，可以用于数据库连接、调用Cypher语句进行数据查询、或定义相关模式和执行管理任务。Cypher Shell采用显式事务方式，允许将多个操作分组一并执行或回滚，通信方式采用加密的二进制Bolt协议。Cypher Shell位于bin目录中，运行时可带上一组参数，第一次运行Cypher Shell时，系统将提示您输入相关的安全信息，按照提示输入即可。其语法为：cypher-shell [-h] [-a ADDRESS] [-u USERNAME] [-p PASSWORD] [--encryption {true,false}] [--format {verbose,plain}] [--debug] [--fail-fast | --fail-at-end] [cypher]。各参数的具体说明如表1所示：

表1 Cypher Shell参数说明

|  |  |  |
| --- | --- | --- |
| 参数类型 | 参数名 | 描述 |
| 位置参数 | cypher | cypher执行后退出的可选字符串 |
| 可选参数 | -h, --help | 显示帮助消息并退出 |
| --fail-fast | 文件读取时，遇到第一个错误就退出并报告失败（这是缺省行为） |
| --fail-at-end | 文件读取时，在输入结束时退出并报告错误 |
| --format {verbose,plain} | 指定输出格式，verbose显示统计信息，plain仅显示数据，缺省值为：verbose |
| --debug | 打印调试信息（缺省值：false） |
| 连接参数 | -a ADDRESS,  --address | 要连接的IP地址和端口（缺省值：localhost:7687） |
| -u USERNAME,  --username | 连接的用户名，也可以使用环境变量NEO4J\_USERNAME指定（缺省为空） |
| -p PASSWORD,  --password | 连接的密码，也可以使用环境变量NEO4J\_PASSWORD指定（缺省为空） |
| --encryption {true,false} | 连接是否加密，必须与Neo4j的配置一致（缺省值：true） |

1. **一致性检查工具**

一致性检查可以使用neo4j-admin工具的check-consistency参数来检查数据库的一致性。该neo4j-admin工具位于bin目录中，语法调用为：neo4j-admin check-consistency --database=<database> [--report-dir=<directory>] [--additional-config=<file>] [--verbose]。命令中的参数详细介绍如表2所示：

表2 一致性检查命令参数说明

|  |  |
| --- | --- |
| 参数名 | 描述 |
| --database | 指定需要进行一致性检查的数据库名称。 |
| --report-dir | 写入报告的目录，默认为工作目录。 |
| --additional-config | 附加配置文件，用于一致性检查的附加配置。 |
| --verbose | 启用详细信息输出，包括：存储信息和内存使用信息。 |

一致性检查工具不能与当前正在使用的数据库一起使用，如果与正在运行的数据库一起使用，则自动停止并输出错误信息。如果一致性检查工具没有发现错误，则程序自动运行结束并不生成相应的报告；如果发现错误，程序将退出，退出代码为1，并将错误信息写入一个格式为inconsistencies-YYYY-MM-DD.HH24.MI.SS.report的报告文件中，此文件的位置在当前工作目录下，或由参数report-dir指定。

#### Neo4j项目实践

##### 基于项目的优化和改造

1. **Neo4j数据库运维与优化**
2. **内存调优**

Neo4j在启动时将自动配置内存相关参数的缺省值，并默认可使用机器上的所有内存。有三种类型的内存要考虑：操作系统内存、页面缓存和堆空间。

需要注意的是，操作系统内存是不能显式配置的，而是在指定页面缓存和堆空间后所“剩下的”。如果将页面缓存和堆空间配置为等于或大于可用内存，或者没有为操作系统留出足够的空间，操作系统内存将进行磁盘交换，这将严重影响性能。请确保操作系统内存配置为永远不需要交换。因此，内存调优可按照如下步骤进行：

* 计划操作系统内存大小

必须为服务器上与Neo4j数据库不相关的其它程序预留一些内存。此外，还需为index和schema目录留出足够的内存用作操作系统的文件缓冲区，因为如果索引不能全部装载在内存中，将影响索引查找的性能。1GB内存是Neo4j服务器的最低配置。基本的计算方法如下：

系统内存=1GB +（graph.db/index）+（graph.db/schema）

* 计划页面缓存大小调整

页面缓存用于缓存存储在磁盘上的Neo4j数据。确保将来自磁盘的所有或至少大部分图数据缓存到内存中，将有助于避免昂贵的磁盘访问并获得最佳性能。可通过简单方法确定页面缓存的大小：汇总含NEO4J\_HOME/data/databases/graph.db/\*store.db\* 所有文件大小，再增加20％的预留。指定页缓存的参数是：dbms.memory.pagecache.size（详情如表3所示），该参数设定允许Neo4j使用多少内存用于高速缓存。如果没有在启动时明确指定，Neo4j将给予默认配置：（机器可用内存 - JVM最大堆分配）\*50％。以下是估计页缓存大小的两种可行的方法：

* 对于现有的Neo4j数据库，查询所有\*store.db\*文件的大小，来确定页面缓存需容纳所有数据的大小，再增加20％。例如：在posix系统中从data/databases/graph.db目录下运行$ du -hc \*store.db\*命令即可获得数据大小。
* 对于新的Neo4j数据库，先导入一定比率的数据（例如总数据量的百分之一），然后将得到的存储大小乘以该比率（此处举例为100），再额外增加20％。

表3

|  |  |  |
| --- | --- | --- |
| 参数 | 可能值 | 效果 |
| **dbms.memory.pagecache.size** | 用于页面缓存的最大内存量，以字节为单位，或者更大的字节单位，例如100M、4G。 | 用于映射存储文件的内存量，以字节为单位。这将自动向下舍入到最近的整页。此值不能为零。对于极小和内存受限的部署，建议仍为页面缓存至少保留几兆字节。 |
| **unsupported.dbms.report\_configuration** | true或false | true表示：写入默认系统输出，主要是控制台或日志文件。 |

* 堆大小

可用堆内存大小是影响Neo4j性能的一个重要因素。一般来说，配置足够大的堆空间以维持并发操作是非常有用的。大多数Neo4j应用，堆大小设置为8-16G 之间即稳定运行。在文件NEO4J\_HOME/conf/neo4j.conf中设置dbms.memory.heap.initial\_size和dbms.memory.heap.max\_size参数即可，以兆字节（MB）为单位，比如：16000。建议将这两个参数设置为相同的值，以避免不必要的垃圾收集。

* 调整垃圾收集器

堆大小需合理分配，使得事务和查询状态从不使它到年老代。堆大小参数设置是：neo4j.conf文件中的dbms.memory.heap.max\_size（以MB为单位），dbms.memory.heap.initial\_size为堆的初始大小。如果未指定，则由JVM自行选择。JVM将根据需要自动增长堆，直到最大值。堆的增长需要一个完全的垃圾收集周期。

新老两代堆之间的大小比例由-XX:NewRatio=N开关参数控制，缺省情况下，N通常在2和8之间。比率为2意味着年老代大小除以年轻代大小等于2，即：堆内存的三分之二将专用于年老代。同理，比率为3则将四分之三的堆分配给年老代，比率为1则两代的大小相同。比率1是相当激进的，不过适合事务更改了大量数据的场景。如果运行需要保留大量数据的Cypher查询，例如在排序大型结果集时，设置大的年轻代大小也很重要。

如果年轻代太小，短暂的对象可能会被过早地移动到年老代，这被称为提前升级，将增加年老代垃圾收集周期的频率，从而降低数据库性能。如果年轻代太大，垃圾收集器可能会决定年老代没有足够的空间来容纳所有希望从新的到旧的对象。这将年轻代垃圾收集周期变成年老代垃圾收集周期，再次减慢数据库的速度。运行更多并发线程意味着在给定的时间跨度内可能发生更多的分配，从而额外增加年轻代的压力。

JVM中的压缩普通对象指针（OOP，Ordinary Object Pointer）功能仅支持32位。该功能可节省大量内存，但大于32GB的堆无法使用。因此，堆大小超过32GB收益可能很小，甚至带来副作用，除非增加非常的多（64GB或以上）。

Neo4j有一些长寿命的对象，它们留在年老代，有利于Java进程的生命周期。为了有效地处理它们，并且不会对垃圾收集暂停时间产生负面影响，我们建议使用并发垃圾收集器。如何调整特定的垃圾收集算法取决于JVM版本和工作负载。为获取有效的垃圾收集参数设置，建议在现有负载下测试几天或几周，堆碎片等问题可能需要更长的时间才能显现。为了获得良好的性能，这些是首先要研究的事情：

* 确保JVM不需要花太多时间来执行垃圾收集。目标是有一个足够大的堆，以确保重/峰值负载不会导致所谓的GC垃圾（Garbage Collection trashing）。当GC垃圾发生时，性能可能下降两个数量级。而堆太大也可能会降低性能，因此需要尝试一些不同的堆大小。
* 使用并发垃圾收集器。大多数情况下使用-XX:+UseG1GC开关参数系统可良好工作。Neo4j JVM需要足够的堆内存用于事务状态和查询处理，以及用于垃圾收集。因为堆内存需求是依赖于工作负载的，所以通常可配置1GB到高达32GB。
* 使用-server开关参数和一个大小合适的堆来启动JVM。专用服务器上的操作系统通常可以使用1到2GB的内存。机器的物理内存越多，操作系统需要的内存越多。

表3 JVM 相关配置参数

|  |  |
| --- | --- |
| 属性名 | 说明 |
| **dbms.memory.heap.initial\_size** | 初始堆大小（单位MB） |
| **dbms.memory.heap.max\_size** | 最大堆大小（单位MB） |
| **dbms.jvm.additional** | 附加文字JVM参数 |

1. **压缩存储**

在很多情况下，Neo4j可以压缩和内联存储属性值，例如短数组和字符串，其目的是节省磁盘空间和提高I/O操作性能。

* 短数组的压缩存储

Neo4j可用压缩的方式存储原始数组，采用“位剃削”算法，以减少存储数组中成员的位数。尤其是：

* 对于数组的每个成员，确定最左边设置位的位置。
* 确定数组中所有成员中最大的位置。
* 它将所有成员减少到该位数。
* 存储这些值，前缀为一个小标题。

这意味着，当在数组中包括单个负值时，将使用原始字节大小来存储。在以下情况下，可能会将结果内联到属性记录中：

* 压缩后小于24个字节。
* 数组成员不到64个。

例如，数组long[] {0L, 1L, 2L, 4L}内联，因为最大项（4）需要3位来存储，所以整个数组需4×3=12位存储。然而，数组long[] {-1L, 1L, 2L, 4L}将需要整个64位用于-1这个元素，因此它需要64×4=256字节，并采用动态存储。

* 短字符串的压缩存储

Neo4j将对字符串按短字符串来分类、管理和处理。因此需进行显式存储，将其内联在属性记录中，即不支持可变长度字符串存储，以减少磁盘空间。此外，字符串记录无需存储属性，可在单个查找中进行读写操作，从而提高性能并减少磁盘空间。

短字符串种类有：

* 数字，由数字0..9和标点符号（空格，句点，破折号，加号，逗号和撇号）组成。
* 日期，由数字0..9和标点符号（空格，破折号，冒号，斜杠，加号和逗号）组成。
* 十六进制（小写），由数字0..9和小写字母a-f组成。
* 十六进制（大写），由数字0..9和大写字母A-F组成。
* 大写，由大写字母A-Z组成，以及标点符号（空格，下划线，句点，破折号，冒号和斜杠）。
* 小写，如上，但字母为小写a-z。
* 电子邮件，包含小写字母a-z和标点（逗号，下划线，句点，破折号，加号和@符号）。
* URI，由小写字母a-z，数字0-9和大多数标点符号组成。
* 字母数字，由大写字母A-Z和小写字母a-z，数字0-9和标点符号（空格和下划线）组成。
* 阿尔法符号，由大写字母A-Z和小写字母a-z和标点符号（空格，下划线，句点，破折号，冒号，斜杠，加号，逗号，撇号，符号，管道和分号）组成。
* 欧洲字符，包括大多数重音的欧洲字符和数字加标点符号空间（破折号，下划线和句号），如latin1，但标点符号较少。
* Latin1。
* UTF-8。

除了字符串的内容之外，字符数还决定了字符串是否可以内联。每类都有字符数限制。限制详情请见表4所示：

表4 字符数限制表

|  |  |
| --- | --- |
| 字符串分类 | 字符数限制 |
| **数字，日期和十六进制** | 54 |
| **大写，小写和电子邮件** | 43 |
| **URI，字母数字和字母** | 36 |
| **欧洲字符** | 31 |
| **Latin1** | 27 |
| **UTF-8** | 14 |

这意味着最大的可内联字符串的长度为54个字符，必须为数字类。所有长度为14或更小的字符串将始终内联。还要注意，上述限制是针对参数PropertyRecord默认为41字节，如果通过修改源代码重新编译来更改该参数，则必须重新计算上述内容。

1. **Linux文件系统调优**

数据库在查询数据时通常会产生许多少量、随机的读操作，而在提交更改时经常会产生少量的顺序写操作。默认情况下，大多数Linux发行版使用完全公平排队（Completely Fair Queuing，CFQ）算法来调度IO请求，该算法在吞吐量和延迟之间作了较好的平衡。然而，期限调度器（Deadline scheduler）更适合于数据库的特定IO工作负载情形。期限调度器优先读操作请求，并尽快处理，以利于减少读操作的等待时间，而增加了写操作的等待时间。由于写操作通常是顺序的，增加其在IO队列中的延迟有益于合并重叠或相邻的写入请求，可有效地减少发送到驱动器的写入次数。Linux系统中IO调度程序可在运行时修改驱动器（sda）：

**echo** 'deadline' **>** **/**sys**/**block**/**sda**/**queue**/**scheduler

**$ cat** **/**sys**/**block**/**sda**/**queue**/**scheduler

noop **[**deadline**]** cfq

另一个推荐做法是禁用文件和目录的访问时间更新。从而，文件系统将不对此元数据进行写操作更新，从而提高写入性能。可在fstab文件中设置noatime，nodiratime挂载选项来实现。

### 产品订阅及NGO协同（5-10页）

#### 服务/产品及订阅说明

##### 服务

##### 产品

##### 订阅指南

#### NGO协同

##### NGO组织架构

##### 组织协同案例