# 基于图形数据库技术的文献资源关联网络构建*

王莉
（中国科学技术信息研究所，北京 100038）

摘要：文献资源不仅规模庞大，而且数据之间存在多种类型的关联关系。采用传统的关系数据库存储暴露出越来越多的问题。文章提出一种基于Neo4j的文献资源模型映射方法，能够快速实现文献资源关联网络的持久化。

关键词：Neo4j；文献资源关联网络；图形数据库；数据模型
中图分类号：G254
DOI：10.3772/j.issn.1673—2286.2014.05.010

## 1 引言

文献作为记录人类知识的载体，集中反映了人类在认识和改造世界过程中积累的知识财富，是一种重要的信息资源。对文献资源的描述与组织一直是图书情报机构的核心能力。传统的方法以详细描述资源实体为目标，侧重于创建独立的、可理解的书目数据。随着网络化、数字化的发展，人们开始关注文献之间、文献内部数据单元之间，以及数据单元与外部世界之间的关系，文献资源的组织从分级树形结构演变为以关系揭示为基础的网状结构，并且超越传统图情领域，融入互联网世界，成为数据网的一部分。

网状结构中数据内部的依赖和复杂度显著增加，采用传统的关系数据库不仅存在大量的数据冗余，而且难于动态更新。图形数据库采用节点和边的形式刻画数据及数据之间的关联，能够准确表达文献资源关联网络。本文提出一种基于图形数据库技术的数据表示与存储方法，重点研究领域对象模型向图模型的转换，快速实现文献资源关联网络的持久化。

## 2 相关研究综述

本研究涉及图书情报和计算机技术两大领域，分领域综述可以清晰地看到领域应用和支撑技术两条发展主线。

### 2.1 书目数据语义化相关研究

编目是图书情报机构描述与组织资源的基本方式，主要基于载体表现和文献单元层次而进行，以编目条例作为资源描述规范，结合MARC格式作为数据交换标准，实现书目控制的目的。随着互联网的飞速发展，网络信息资源类型日益丰富，MARC不断升级调整以适应新的资源环境，同时，业界开始探索如何建立一个更简单的元数据模型和架构。

笔者对业界关于数据模型的研究进行梳理，归纳出三类主流的研究方法。一是以图书馆为中心，从用户对书目记录的需求出发，构建一个描述资源及资源间关系的概念模型。这一类研究以IFLA主持的FRBR研究为代表。FRBR模型对传统编目产生了巨大冲击，在语义网、关联数据发展浪潮下，IFLA编目部2011/2013战略规划中提出，参与开发ISBD、FRBR、FRAD、FRSAD的命名域，将IFLA标准与模型置于语义网环境中。借助FRBR模型，编目实践迈向语义网时代[1]。第二类研究在开放关联大环境下重新审视图书馆数据，旨在通过一个更普遍的描述框架使得图书馆界融入和参与到更广泛的信息社会中，其典型代表是美国国会图书馆提出的书目框架倡议及BIBFRAME草案。BIBFRAME

* 本研究得到国家十二五科技支撑计划课题“信息资源自动处理、智能检索与STKOS应用服务集成”（编号：2011BHA10B05）资助。

模型以网络为基础架构，试图将MARC21格式拆解并重构，塑造一个包含MARC信息的普遍的书目描述框架。虽然BIBFRAME模型仍然是面向书目数据的，但是这种对书目数据关联数据化的思考，将引导图书馆融入更大的数据网。2012年9月，Schema.org书目扩展小组成立，旨在在图书馆数据模型与Schema.org之间建立桥梁，让Schema.org更好地适应图书馆数据。德国国家图书馆的实践[2]则代表了开放关联环境下的另一种思潮：当书目在万维网中被应用与复用时，我们应该认识到，图书馆所提供的服务将越来越多地有赖于外部资源信息，对于图书馆来说，可以放弃一些工作（对于部分信息来说，缩减自己组建数据而利用第三方信息），更多地关注我们的力量（例如：高质量信息的创建、知识组织、规范控制）。第三类研究以W3C图书馆关联数据孵化小组的工作为代表。2011年10月孵化小组发布系列LLD（Library Linked Data）报告。报告将图书馆及相关领域的应用模式分为书目数据、规范数据、数字对象等8大类，通过对57个实例的描述与总结，虽然没有提出明确的数据模型，但是为图书馆关联数据应用提供了非常有价值的指南。文献[3]在LLD报告的基础上，系统评述了图书馆数据的类型与应用，提出图书馆关联数据的宏观语义模型。文献[4]从资源发现、数据融合与语义检索、学术研究与交流、跨机构的关联数据开放与复用等方面对图书馆关联数据应用进行了分析，并且指出：“图书馆应充分利用关联数据源中的关联关系，实现有序地组织、集成和关联知识单元，进行知识内容的关联和深层展示。”

通过梳理可以看到，图书馆数据模型的研究尽管存在多种不同的研究方法和角度，但是其成果产出都呈现关联化特性。在关联数据运动中，图书馆不仅仅是发布者，也是消费者，通过与外部数据集建立关联，图书馆数据不再是终点，而是跨越数据类型得到其他有用数据的起点[3]。

### 2.2 图形数据库相关应用

计算机领域解决数据模型的存储问题，在众多存储技术中，关系数据库长期占据统治地位。随着Web 2.0应用、社交网络的兴起，数据内部依赖和复杂度增加，关系数据库暴露出越来越多的问题，NoSQL运动迅速兴起[5]。NoSQL即“不仅仅是SQL”（Not only SQL），代表了一类非关系型数据库解决方案，例如对象数据库、文档数据库、键值对数据库和图形数据库。在具有网络模型结构特征的应用中，采用图形数据库进行存储是最适合的方案之一。

图形数据库的理论基础是图论。图论以图为研究对象，是数学的一个分支。图论中的图是由若干给定的点及连接两点的线所构成的图形，点代表事物，连接两点的线表示相应两个事物间具有的某种关系。图论早期起源于数学游戏，二十世纪六十年代以后，由于高速计算机的广泛使用，在计算机科学的发展和推动下，图论得到飞速发展，其巨大用途得到认可，相关的理论与算法也逐渐从研究走向应用。近年来出现了一些可用于产品环境的高性能图形数据库[6]，例如Neo4j、Infinite Graph、DEX、InfoGrid、HyperGraphDB、Trinity、FlockDB以及AllegroGraph等，其中，Neo4j是目前比较主流的一款基于Java实现的开源软件，其内核是一种极快的图形引擎，具有恢复、两阶段提交、支持XA事务等数据库产品特性。Neo4j在语义网和RDF、关联数据、GIS、社交网络、基因分析和深度推荐算法等领域具有巨大的应用前景，自2003年起，它已经被作为24/7的产品使用[7]。

## 3 文献资源关联网络框架

文献资源关联网络不仅仅是建立文献之间的关联，更重要的是建立文献及文献内部数据的关联，其基础是文献资源描述和组织方法，只有在一个普遍的描述框架下展开更细粒度、更完整的描述与分析，才能发现和识别更多的关系，进而向外部世界延伸，使最大程度地利用数据中的价值成为可能。笔者并不试图去建立一个普遍的描述框架，而是将研究重点放在对关联模型持久化的技术探索上。因此，在本研究中笔者借鉴BIBFRAME“将MARC21格式用于数据网络建模”的工作思路，以国家科技图书文献中心（以下简称NSTL）文献资源为样例，对其数据著录格式进行拆解进而重构，形成NSTL文献资源关联框架，图1展示了其包含的主要实体及实体间的基本关系。

NSTL文献资源关联框架包含资源、主体、活动、主题四组实体。“资源”通过划分“作品”和“实例”明确概念性内容和物理表现。“作品”是抽象实体，可以理解为一个著录对象，记录了与作品的知识本质相关的数据。“实例”是抽象作品的物理表达，可以是不同的格式/载体，也可以具体到存放的物理位置。例如，对于

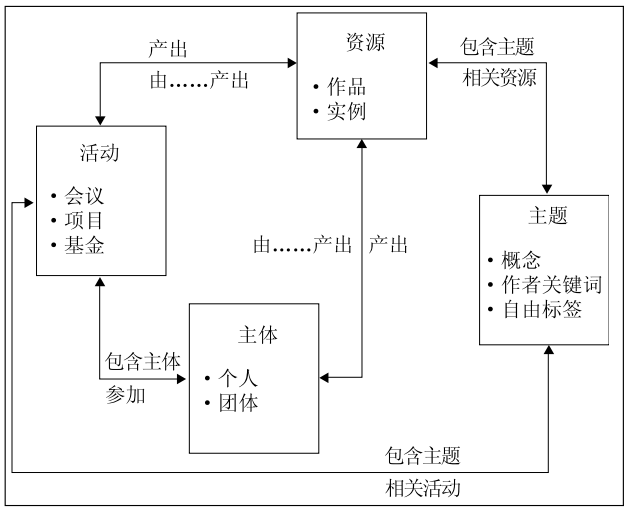


图1 NSTL文献资源关联模型

一本图书来说，“实例”可以是某个收藏单位存放的印本，也可以是PDF文件的访问链接。主体表示资源产出责任，包括个人和团体。“主体”可以是一个“作品”的创作者，也可以是一个“实例”的生产者或收藏者。“活动”包括会议、项目、基金、演讲等，活动的参加者即“主体”，而“作品”则可能是某项活动的学术产出。“主题”包括概念、对象、事件和地点，与“作品”和“活动”直接相关。图2展示了从一篇期刊文献著录信息（来自NSTL文献数据仓储）中抽象出的关联图。

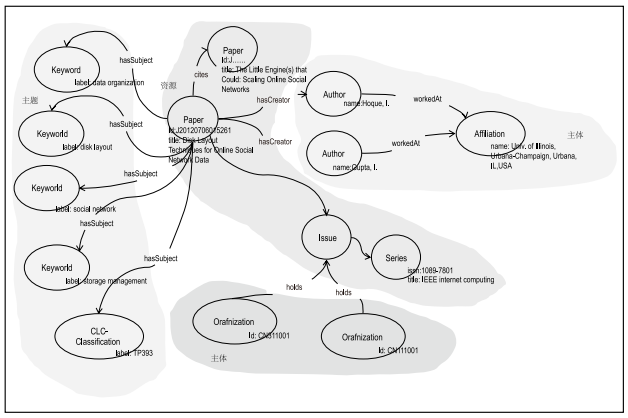


图2 期刊文献关联示例：Disk Layout Techniques for Online Social Network Data （doi:10.1109/MIC.2012.40）

## 4 关联网络构建方法

文中第3节给出的关联框架可以看作一个简单的领域对象模型，关联网络的构建则是从领域应用模型到数据存储格式的映射过程。Neo4j中基本的数据模型由节点（Node）、关系（Relationship）和属性（Property）构成。节点类似于对象实例，拥有唯一的ID（由Neo4j自动生成）。不同节点通过各种不同的关系关联起来。关系类似于有向图中的边，由起始节点、终止节点和类型三个要素组成，其方向性进一步厘清了节点之间的语义关系。节点和关系都可以定义属性，采用key-value值对表示。沿用图2的例子，文献和作者是不同的实体，两者之间建立撰写关系，用Cypher语言[8]描述见示例1。

示例1：

```
(paper {title:〝Disk Layout Techniques for Online Social Network Data〞,
doi:〝10.1109/MIC.2012.40〞) –[:CREATEDBY]->
(author {name: 〝Hoque, I.〞})
```

其中，“(paper {title:〝Disk Layout Techniques for Online Social Network Data〞,
doi:〝10.1109/MIC.2012.40〞)”标记了paper节点，“{}”中是为该节点定义的title和doi两个属性，采用“key:value”形式表示。同样，“(author {name: 〝Hoque, I.〞})”标记的是author节点。“[:CREATEDBY]”标记了撰写关系。整个语句描述了两个节点之间的有向关系，具有很强的表达能力。这只是一个非常简单的示例。数据建模的关键就是将节点和领域实体对应起来，从简单节点、节点间的直接关系构建开始，节点不断成长，节点间关系逐渐丰富，最终形成一个复杂图。以下讨论该过程涉及的一些核心问题。

### 4.1 建立面向业务应用的数据模型

节点、关系和属性是Neo4j的基本数据结构，虽然利用这种简单结构就可以灵活地表达复杂网络关系，但是在实际应用中，仍然有必要建立面向业务的数据模型，对节点进行分类管理。简单地说，面向业务的数据模型是对实体/对象“类”的描述，节点则是“某个类”下的具体实例。得到一个业务数据模型很简单，可以直接从领域分析中的E-R模型演化，关键在于如何转换为具体的数据结构并发挥作用。当采用关系数据库时，这个问题非常简单，表结构的定义类似于对实体/对象“类”的描述，如具备哪些特征、特征的取值范围等；具体的实例则以行的方式存储在表中。在Neo4j中，并没有“表

结构”这一层次的描述，而是直接定义实例层数据，每个实例拥有自己的属性，即使是相同类型的实体也可以根据需要拥有不同类型不同数量的属性特征，没有类型定义的约束，只保留有意义的属性。这种实现方式给Neo4j带来巨大的灵活性，同时庞大的节点空间也遇到管理维护上的问题。

解决方法一：利用节点标签“label”实现节点的分组。

实际应用中，开发人员通常会给节点增加一个“type”属性，用来记录其类型；Neo4j设计者从这一常见开发模式中析出一个非常特殊的“label”项。如同其名称的字面含义一样，label是一个标签，而不是属性，不具备“keys/values”值对特征，它提供的是一种对节点进行分组的方法。简单地说，在Neo4j中可以采用label来标识一组节点集合，进而在该集合上执行建索引、定义约束、和查询等操作。例如，我们在定义文献节点的时候，可以采用CREATE (paper: JournalArticle)，表示建立一个paper节点，通过标签JournalArticle表示该节点是期刊论文。该节点与具有同样标签JournalArticle的文献节点构成一个集合。示例2展示了如何在节点集合上进行索引和查询操作。

示例2：

```
1） CREATE INDEX ON : JournalArticle(doi)
2） MATCH (n:JournalArticle {doi:’ .....’ })
ETURN n
```

当执行查询命令时，仅在JournalArticle标记的节点集合中查找，不需要搜索整图，能够极大地提高查询效率。

需要注意的是，label用来标识一个节点集合，可以为所属节点的属性定义某些限制、增加索引。它类似关系数据库中表的名称，而并非严谨的“表结构”。一个节点可以不赋予label，也可以赋予一个或多个label。归根结底，label机制提供的只是一种对节点进行分组的方法，建立在分组上的管理需要采用其他机制来实现。

在Neo4j中处理关系时，允许设置关系的类型，这一设置是为图的遍历做准备，虽然采用了关键字“type”，但其作用仍然只是“标签”，其性质与节点的“label”相同。

方法二：定义专门的类

除了利用数据库自身的机制外，当使用Neo4j内嵌应用模式时，还可以从应用程序架构入手寻找合适的解决方法。Neo4j支持Java、Python、Scala、PHP、Clojure等多种编程内嵌。以Java为例，只需要下载jars包并引入项目工程中，就可以创建并操作本地Neo4j数据库。Neo4j教程[9]中介绍了一种“将实体封装到节点上”的方法。文章[10]进一步阐述了如何用该方法实现业务应用，其基本思路是：首先，创建一个普通接口，在接口实现类里定义underlyingNode属性，这样，节点与领域实体就对应起来。接着，采用Factory模式完成节点实例及关系的创建工作。在创建关系时，利用Neo4j提供的参考节点（reference node）和子参考节点（subreference node）机制，将关系和对象之间的引用对应起来。

编程实现时除了直接使用底层API之外，更好的方式是采用第三方框架，例如Spring、Grails、Griffon、Django等。Spring框架中的Spring Data Neo4j提供了对Neo4j的良好支持，将Neo4j数据库中创建、读取、更新、删除、索引、图遍历等操作进行了封装，提供更加抽象易懂的API。在实现业务应用时，开发人员可以直接通过注解来声明节点类（@NodeEntity）以及节点类之间的关系（@RelationshipEntity），构建的数据存储模型清晰易懂。

Grails是在Spring、Hibernate等标准Java框架之上构建的一套Web应用开发开源框架，其Neo4j插件最新版本为1.1.1。图3描述了在Grails中如何将领域实体映射到Neo4j节点空间。Neo4j自动生成的参考节点是

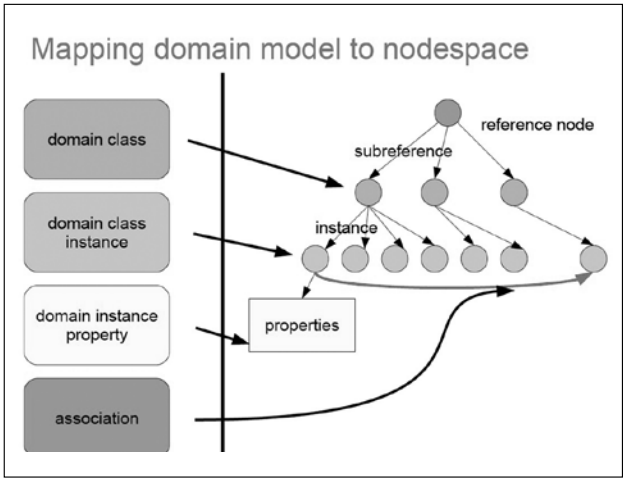


图3 Grails：从领域模型到节点空间的映射[11]

图的起点，首先引入子参考节点，每一个实体类都用一个子参考节点表示，通过SUBREFERENCE关系挂接在参考节点上。领域对象实例，也就是应用域中的真实数据，对应每一个具体的节点，挂接在子参考节点上，用关系INSTANCE连接。最后将实例属性映射为节点属性。由于Neo4j中的节点属性只能是基本数据类型，在做属性映射时可能需要作相应的转换。

## 4.2 关系的处理

在创建节点类/节点的过程中，节点类/节点之间的关系也同步建立起来。一般来说，不同实体类型之间的关系在模型分析过程中大都可以明确定义，不同类型节点（实例）之间的关系属于其中的一种或多种；而同类型节点（相类实体下的不同实例）之间的关系大都是未知的，需要在检索和遍历过程中通过关联路径发现。因此，创建关系的基本原则是：关注领域模型分析，将那些稳定并频繁出现在不同实体类之间的关系固化下来，可以作为常量定义在单独的类中。

在实现业务应用的时候，即使面对模型分析中那些明确的关系，也没有必要全部建立，筛选时应主要考虑双向关系和简单隐含关系的处理。现实世界中有很多关系是双向的（也可称为无向），比如合著（coauthor）。但是，在Neo4j中所有关系都是有向的，在这种场景下，最好的建模方式是在两个节点间创建一个有向的coauthor关系，同时在检索过程中忽略关系的方向。所谓简单隐含关系，是指一个关系暗示另外一种关系的情况，例如引用（cite）和被引用（cited），只需要选择创建一种关系即可。

此外，应用场景分析对关系的创建也非常重要，不同的应用场景可能需要建立不同粒度的关系描述[12]。例如创作主体和作品之间的关系可以统一称为PRODUCEDBY，也可以进一步区分著、编、译等不同的责任方式。在业务实现的时候，我们通常希望尽可能详尽地描述并区分节点之间的关系，因此建立一套细粒度的关系类型。当进行图遍历操作时，指定关系类型能够有效地缩小遍历图的大小，提高遍历速度。然而，在现实中也可能遇到需要在更大范围的图中检索的情况，这时可以将所有的关系类型都包括在检索语句中。但是，当关系类型非常多时，会造成检索语句冗长，而且执行速度也不可控。更好的方法是在节点间增加一个粗粒度的、更普遍的关系类型，以适应不同的应用场景。

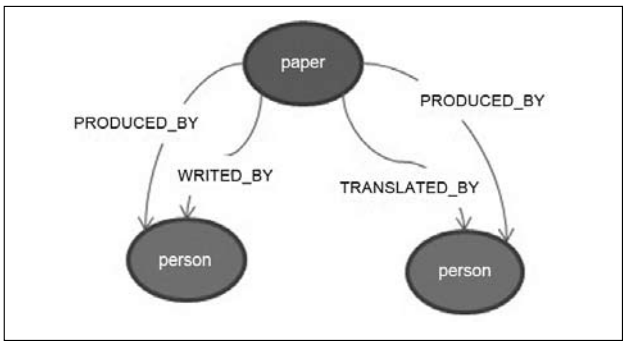


图4 针对不同应用场景建立不同关系

## 4.3 给图建立索引

现实世界中主要有两种查询场景：一是通过一个或多个给定的条件找到与其相符的目标；二是已知某个事物，查找与其有关联的信息。前者是传统的检索方式，Neo4j采用集成外部索引系统Lucene和Solr的方法实现了对所有节点属性的全文检索。创建索引的过程非常简单，在查询时也不需要指定使用某个索引，Neo4j会自行调度。后一种应用场景是一种遍历操作（traversal），也是Neo4j的长项，采用某种特定算法，从一个或一些起始节点开始查询与其关联的节点。

Neo4j支持非常复杂的图遍历操作，当采用Cypher查询时，非常简单，只需要通过START（起点）、MATCH（匹配模式）、WHERE（过滤条件）、RETURN（返回）四个关键字描述如何从图中寻找即可。当采用Java实现时，Neo4j提供一套功能强大的API，由TraversalDescription、Evaluator、Traverser、Uniqueness、Order、BranchSelector、Path、PathExpander/RelationshipExpander、Expander等主要接口组成遍历框架[13]。实现时需要通过TraversalDescription接口指定遍历方式，包括路径、顺序、唯一性、决策器和遍历起点等信息。

图遍历是隐含关系发现的主要手段，在构建数据模型时，也可以充分利用这一机制，将那些比较明确的简单隐含关系固化下来，在Cypher中体现为路径模式，在Java实现中可以用path context或遍历模板（或称“遍历器”）的形式定义。例如，合著关系（coauthor）是不同作者节点之间通过共同作品表现出来的，在Cypher查询中可以采用(a:person)-[:WRITES]->( )<-[:WRITES]-(b:person)路径刻画。Java实现则可以定义一个COAUTHOR_TRAVERSAL遍历器，进而通过

“path:COAUTHOR_TRAVERSAL”方式让其他不同的遍历共享。

## 5 Neo4j建模实验

以NSTL期刊论文数据为样例，进行文献资源关联网络构建实验。

实验环境：1）硬件配置：普通PC电脑，AMD FX™-4300 Quad-Core Processor 3.80GHz，4GB内存；2）操作系统：Windows 7专业版 32位；3）Neo4j数据库版本2.0.M06。

数据模型：

Neo4j UI管理界面展示：

1）数据库概貌

选择NSTL期刊论文数据1371条，建模之后得到节点8056个。

表1 从期刊论文数据构建的Neo4j数据模型

| 节点类 | | | |
|---|---|---|---|
| | **标签** | **主要属性** | **说明** |
| 1 | JournalArticle | Key:string Title:string Doi:string url:string | 期刊论文 |
| 2 | JournalList | Key:string Vol:string | 期刊卷期 |
| 3 | Journal | Key:string ISSN:string | 期刊 |
| 4 | Person | Key:string Name:string | 个人 |
| 5 | Organization | Key:string Name:string | 机构 |
| 6 | Keyword | Key:string Name:string | 关键词 |
| 7 | CLC | Key:string Name:string | 中图分类法 |
| 关系 | | | |
| | **描述（形如：起始节点-[关系名]->终止节点）** | | **说明** |
| 1 | (n:Journal) - [:INCLUDE]->(m:JournalList) | | 期刊与卷期之间的包含关系 |
| 2 | (n:JournalList) - [:INCLUDE]->(m:JournalArticle) | | 卷期与论文之间的包含关系 |
| 3 | (n: Person) - [:WRITE]->(m: JournalArticle) | | 个人与论文之间的撰写关系 |
| 4 | (n: Person) - [:EDIT]->(m: JournalArticle) | | 个人与论文之间的编著关系 |
| 5 | (n: Person) - [:TRANSLATE]->(m: JournalArticle) | | 个人与论文之间的翻译关系 |
| 6 | (n: Person) - [:PRODUCE]->(m: JournalArticle) | | 个人与论文之间的产出关系 |
| 7 | (n: Organization) - [:PUBLISHE]->(m: Journal) | | 期刊与机构之间的出版关系 |
| 8 | (n:JournalList) - [:STORE]->(m:Organization) | | 期刊与机构之间的收藏关系 |
| 9 | (n:Author) - [:WORKS_IN]->(m:Organization) | | 个人与机构之间的雇佣关系 |
| 10 | (n:JournalArticle) - [:CITE]->(m:JournalArticle | | 论文之间的引用关系 |
| 11 | (n:JournalArticle)-[:HAS_KEYWORD]->(m:Keyword) | | 论文与关键词之间的所属关系 |
| 12 | (n:Journal)-[:HAS_CLC]->(m:CLC) | | 期刊与中图分类之间的所属关系 |
| 基本路径 | | | |
| | **描述（形如：起始节点-[:关系名]->终止节点）** | | **说明** |
| 1 | (n:JournalArticle) - [:INCLUDE]->(m:JournalList) - [:INCLUDED_BY]->(m:Journal) | | 文献出处 |
| 2 | (n1:JournalArticle) - [:CITE]->(m:JournalArticle) <- [:CITE]-(n2:JournalArticle) | | 论文n1和n2之间的共引关系 |
| 3 | (n1:JournalArticle)< - [:CITE]-(m:JournalArticle) - [:CITE]->(n2:JournalArticle) | | 论文n1和n2之间的共被引关系 |
| 4 | (n1:Person)<-[:WRITE]-(m:JournalArticle) -[:WRITE]->(n2:Person) | | 作者n1和n2之间的合著关系 |
| 5 | (n1:JournalArticle)-[:WRITE]->(n:Person) <-[:WRITE]-(n2:JournalArticle) | | 论文n1和n2之间的同作者关系 |


图5 数据库概貌

2）节点及关系图形化展示

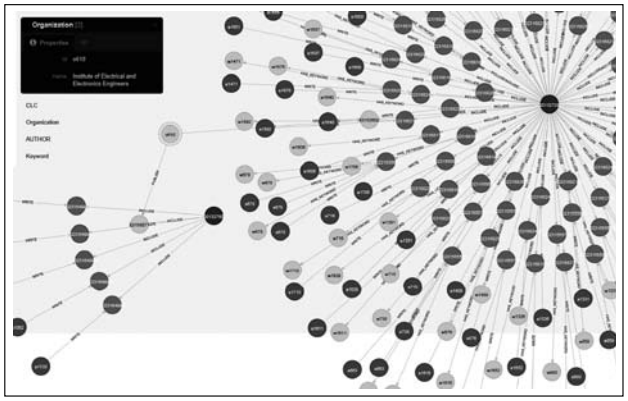

图6 从节点空间中查找与IEEE相关的信息（涉及263个节点及262个关系）

## 6 结语

图形数据库采用图的形式准确表达关联网络，能够有效解决关系数据库因为数据内部依赖和复杂度增加而暴露出的问题。本文提出一种基于图形数据库技术的数据表示与存储方法，重点研究领域对象模型向图模型的转换，以NSTL期刊论文数据为样例，完成Neo4j数据建模。该建模方法应用到NSTL智能检索平台的设计中。

## 参考文献

[1] 李恺.RDA和语义网[J].数字图书馆论坛,2010(12):8-15.

[2] 赵梦(译).作为关联开发数据的德国国家书目:应用与机遇[EB/OL]. [2014-01-23]. http://conference.ifla.org/past/ifla78/215-kettzh.pdf.

[3] 范炜.走向开放关联的图书馆数据[J].图书情报知识,2012(2):94-102.

[4] 黄永文.关联数据在图书馆中的应用研究综述[J].现代图书情报技术,2010(5):1-7.

[5] 孙立.关系数据库还是NoSQL数据库[EB/OL]. [2014-03-18]. http://www.infoq.com/cn/news/2011/01/relation-db-nosql-db.

[6] 邹磊,陈跃国.海量RDF数据管理[J].中国计算机学会通讯,2012(11):32-43.

[7] neotechnology graphs are everywhere [EB/OL]. [2014-03-18]. http://www.neotechnology.com/.

[8] Neo4j简体中文手册: Cypher查询语言[EB/OL]. [2014-03-18]. http://docs.neo4j.org.cn/cypher-query-lang.html.

[9] Neo4j简体中文手册: 领域实体[EB/OL]. [2014-03-18]. http://docs.neo4j.org.cn/tutorials-java-embedded-entities.html.

[10] Neo4j如何实现业务应用[EB/OL]. [2014-03-18]. http://www.neo4j.org.cn/2012/08/29/neo4j-in-active-project/.

[11] ARMBRUSTER S. Grails Goes Graph [EB/OL]. [2014-03-18]. http://download.irian.at/2012/f9u4mq2xdg/d2_f_1645_Stefan_Armbruster_grails_goes_graph.pdf.

[12] O'Reilly. Graph Databases [EB/OL]. [2014-03-18]. http://www.graphdatabases.com/.

[13] Traversal框架[EB/OL]. [2014-03-18]. http://www.neo4j.org.cn/2012/07/27/neo4j-traversal-framework/.

## 作者简介

王莉，女，硕士，中国科学技术信息研究所研究员，研究方向：数字图书馆。E-mail：wangli@istic.ac.cn。

Design of an Associated Network of Literature Resources Based on a Graph Database

WANG Li
（Institute of Scientific and Technical Information of China, Beijing 100038, China）

Abstract: Literature resources have a large data scale and complex relationships. Using relational database as storage means has resulted many problems. This paper presents a model mapping method of literature resources based on Neo4j. This method can quickly realize the associated network's persistence.

Keywords: Neo4j; Associated network of literature resources; Graph database; Data Model

（收稿日期：2014-03-18）