

一种基于 Neo4j 图数据库的模糊查询研究与实现

李 雪

(南京航空航天大学 计算机与技术学院,江苏 南京 211106)

摘 要:在数据库中进行查询时,数据库无法对形如“高的”、“很近”这种不精确的表达进行处理,关系数据库对于这方面的研究已经提出了很多方法,但是数据量呈指数级增长,关系数据库已经无法对这样庞大的数据量进行存储和处理。Neo4j 图数据库不需要在表中存储结构化的数据,而是只需要将所有的节点通过关系连接起来,以网络的形式存储,将所有的节点通过关系形成一个很大的图,这样对数据节点和关系进行增删以及查找就非常容易。针对以上不足,提出一种使用 Neo4j 图数据库进行模糊查询扩展的方法。该方法通过添加一种领域特定语言对 Neo4j 图数据库的查询语言 Cypher 进行扩展,主要是基于图中节点的属性,对 Cypher 查询语言的 WHERE 子句进行扩展,通过隶属函数以及 α 截集相关知识,对不精确的查询条件进行去模糊化,并且将它们转换成精确的结果区间,然后通过这个确定的结果区间在数据库中进行查询,得到了所需的结果。最后设计了一个可以在 Neo4j 图数据库进行模糊查询的系统,并验证了该系统的可行性以及有效性。

关键词:模糊查询;Neo4j 图数据库;Cypher 查询语言;查询语言扩展;隶属函数; α 截集

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2018)11-0016-06

doi:10.3969/j.issn.1673-629X.2018.11.004

Research and Implementation of a Fuzzy Query Based on Neo4j Graph Database

LI Xue

(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract: When we query in a database, the database can't handle the inexact expression, such as "high" or "very close". Many works have been proposed for dealing with fuzzy data and queries about the relationship database, but the growth of data is exponentially increasing, the relational databases have been unable to store and process these huge data. Neo4j graph database does not need to store structured data in the table, but just need to connect all the nodes through the relationship and store in the form of network. All the nodes through the partnership are formed a great figure, which makes it easy to add and delete data nodes and relationships. Aiming at the above deficiencies, we propose a method of using Neo4j graph database for fuzzy query extension. By adding a domain specific language for extending the Cypher query language of Neo4j graph database, mainly based on the attributes of nodes in the graph, we extend WHERE clause of the Cypher query language. Through the knowledge of membership functions and alpha cut sets, we can defuzzy the inaccurate query conditions, transform them into precise result intervals, and then query through the determined result interval in the database. Therefore, we can get the results we need. Finally, a fuzzy query system can be carried out in the Neo4j graph database, and its feasibility and validity are also verified.

Key words: fuzzy query; Neo4j graph database; Cypher query language; query language extension; membership functions; Alpha cut sets

0 引言

在日常生活中,人们想要得到某种结果时,并不能很明确地表达他们的查询要求。例如,寻找一个“高个的年轻人”,其中“高”和“年轻”这样的词语就是模

糊的概念。但是一般的数据库管理系统并不能对这样的模糊语句进行查询,因此需要找寻一种有效的方法将模糊的查询条件转化成精确的查询区间,使其可以在数据库管理系统中执行。

收稿日期:2018-01-03

修回日期:2018-05-05

网络出版时间:2018-06-29

基金项目:国家自然科学基金(61370075)

作者简介:李 雪(1991-),女,硕士研究生,CCF 会员(E200054521G),研究方向为数据与知识工程。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20180629.1706.056.html>

随着信息网络的飞速发展,各种数据量呈指数级增长,传统的关系数据库无法对这些数据进行很好的处理。为了解决数据量增多以及关系型数据库在处理复杂结构这方面的不足,引入了 NoSQL 数据库,它被认为是管理大量的图数据或者复杂数据的最佳选择^[1]。

众所周知,图通常可以使用一种很自然的方法来表述这个世界^[2]。图被应用于许多领域,可以用来表示或存储数据,在不同的场合,图亦拥有很多形式的表示方法或者使用方法^[3]。理所当然地,图数据库概念的诞生很快吸引了众多专家学者的关注与研究,它可以有效地管理网络实体中每个节点的特定属性,以及每条边和实体之间的联系^[4]。Neo4j 是图数据库中比较受欢迎的一种数据库,可以很容易地对关系进行插入删除,在各个领域应用广泛^[5]。

国内外对模糊查询进行了大量研究,王青松根据模糊数据自身的各种特性,通过聚类的方法自动生成隶属函数,避免了人为设定隶属函数的主观性^[6]。王慧和陈逸菲等不仅对经典关系数据库的查询语言进行了扩展,使其可以处理模糊查询条件,同时还通过对隶属度的计算将权重的概念引入模糊查询,满足了人们查询时各种偏好的设定^[7-8]。郑知卉等使用正向法和反向法对查询条件进行处理,将其转化成精确的查询区间^[9-10]。

对 NoSQL 模糊查询研究最多的是 A Castelltort 等,他们对 NoSQL 图数据库 Neo4j 的查询语言 Cypher 进行扩展,得到 Cypherf 语言使其可以进行模糊查询,并分别对基于属性、节点和关系的查询语言的扩展进行了研究与介绍^[11]。在此基础上,他们还引入了必要的框架来支持 NoSQL 图形数据库中的近似查询,通过定义和使用 Scala,提出 Fuzzy4S 框架和 Cypherf 模糊声明性查询语言^[11]来定义和运行 NoSQL 图形数据库的近似查询。其中 Fuzzy4S 通过领域特定语言(DSL)提供一个框架来定义近似的语言变量。然后,该 DSL 可以用于查询带有 Cypherf 的 NoSQL 图形数据库,该数据库扩展了现有的声明性语言来管理近似查询,该方法将整个链从最终用户的声明性查询级别嵌入到数据库引擎内的实现问题中^[12]。

Arnaud Castelltort 等还通过对 NoSQL 图数据库进行扩展,即扩展用于管理带有 DSL 的不精确查询(定义了灵活的操作符和使用 Scala 的操作),以及在 NoSQL 图形数据库中使用 Cypherf 声明性的灵活查询语言来处理 and 描述复杂的欺诈行为^[13]。他们还提出使用合适的工具用来在 Neo4j 的 Cypher 语言中表达模糊查询,用模糊性来帮助解决模糊模式匹配^[14]。

SQL 模型,即 FNoSQL(Fuzzy NoSQL),用来处理大数据的同时也扩展了 NoSQL^[15]。Olivier Pivert 等提出了一个可以灵活查询模糊数据库的系统 SUGAR,该系统是建立在 NoSQL 图数据库管理系统中的,支持 Cypher 查询,由转换模块和分数计算器模块组成,并且引进了 Fudge 语言来实施,该语言可以灵活地处理模糊的或者精确的数据,可以用来处理模糊图数据库中的偏好查询^[16-17]。

1 相关基础知识

1.1 隶属函数

对于论域 $U \in [0, 1]$ 上的模糊集合 A , 可以表示为:任意一个元素 $x \in U$, 都有与其对应的一个隶属函数 $\mu_A(x) \in [0, 1]$, 使得:

$$\begin{cases} \mu_A(x): U \rightarrow [0, 1] \\ x \rightarrow \mu_A(x) \end{cases} \quad (1)$$

其中, $\mu_A(x)$ 表示隶属度 x 属于模糊集合 A 的程度。

例如: 设论域 $U = [0, 100]$, 模糊集 A 表示“young”, 模糊集 B 表示“old”, 则

$$\mu_{\text{young}}(x) = \begin{cases} 1 & 0 \leq x \leq 25 \\ \frac{1}{1 + (\frac{x-25}{5})^2} & 25 < x \leq 100 \end{cases} \quad (2)$$

$$\mu_{\text{old}}(x) = \begin{cases} 0 & 0 \leq x \leq 50 \\ \frac{1}{1 + (\frac{x-50}{5})^2} & 50 < x \leq 100 \end{cases} \quad (3)$$

1.2 α 截集

论域 U 中所有的元素 x 所对应的隶属函数 $\mu_A(x)$ 的值都不小于 α 的一个集合称为模糊集合 A 的 α 截集:

$$A_\alpha = \{x | \mu_A(x) \geq \alpha, \forall x \in U, \alpha \in [0, 1]\} \quad (4)$$

其中, α 是置信水平(阈值)。

1.3 语气算子

在日常生活中,经常会说“很好”“非常大”“比较容易”等句子,其中“好”、“大”、“容易”等词都是基本单词,用“很”、“非常”、“比较”等词放在它们的前面,对它们的语气程度进行了调整。这种放在基本单词之前,可以对语气程度进行调整的词称为语气算子。

定义 1: 设 H_α 为语气算子, $H_\alpha A = A^\alpha$, 如果 α 小于 1, 称 H_α 为散漫化算子, 如果 α 大于 1, 称 H_α 为集中化算子。其中集中化算子会将原词的影响范围变窄, 而散漫化算子正好相反。

假设原词的隶属函数为 $\mu(x)$, 可以有以下规定:

(1) “很”、“非常”等词加上基本词的隶属函数为

$[\mu(x)]^2$;

(2) “稍微”、“略微”、“有点”等词加上基本词的隶属函数为 $[\mu(x)]^{0.5}$;

(3) “极”等词加上基本词的隶属函数为 $[\mu(x)]^4$ 。

比如非常老的隶属函数为:

$$\mu_{\text{非常老}} = \begin{cases} 0 & 0 \leq x \leq 50 \\ \frac{1}{[1 + (\frac{x-50}{5})^{-2}]^2} & 50 < x \leq 100 \end{cases} \quad (5)$$

1.4 模糊化算子

“差不多”、“几乎”、“大约”等词和语气算子不同,它们可以放在原词前,将原词的词义模糊化,这些词统称为模糊化算子。考虑到这些词意思相近,只将“大约”作为标准的模糊化算子。

定义 2: 可以用一个相似模糊关系 $\mu_E(x,y)$, $x,y \in U$ 在模糊化算子和被修饰词的隶属函数 $\mu(y)$ 之间作运算,来实现对隶属函数 $\mu(y)$ 的模糊化,记作:

$$F(A)(y) = (E \circ A)(y) = \bigvee_{y \in U} (\mu_E(x,y) \wedge \mu_A(y)) \quad (6)$$

其中,将 δ ($\delta > 0$) 作为一个参数对模糊范围进行调节, E 为相似模糊关系,记为:

$$E(x,y) = \begin{cases} e^{-\left(\frac{x-y}{\delta}\right)^2}, & |x-y| < \delta \\ 0, & |x-y| \geq \delta \end{cases} \quad (7)$$

2 模糊查询语言扩展

要在 Neo4j 图数据库上实现模糊查询,关于模糊性的处理可以基于三方面,分别是基于节点(图数据里面的一个个记录),关系(用来连接节点的一条条边)和属性(也叫作数据的值)。选择一个电影演职员关系图(部分)(见图 1)进行介绍。

a	m	d
{born: 1956, name: Tom Hanks}	{tagline: A story of love, lava and burning desire., title: Joe Versus the Volcano, released: 1990}	{born: 1950, name: John Patrick Stanley}
{born: 1956, name: Tom Hanks}	{tagline: At odds in life... in love on-line., title: You've Got Mail, released: 1998}	{born: 1941, name: Nora Ephron}
{born: 1956, name: Tom Hanks}	{tagline: Everything is connected, title: Cloud Atlas, released: 2012}	{born: 1967, name: Andy Wachowski}
{born: 1956, name: Tom Hanks}	{tagline: Everything is connected, title: Cloud Atlas, released: 2012}	{born: 1965, name: Tom Tykwer}
{born: 1956, name: Tom Hanks}	{tagline: Everything is connected, title: Cloud Atlas, released: 2012}	{born: 1965, name: Lana Wachowski}

图 1 电影演职员关系

图 1 显示了在数据库中的演职员和电影的例子,包含了两种节点,分别是导演或者参演电影的演职员,以及上映的电影;其中有“DIRECTED”和“ACTED_IN”这两种关系;关系和节点都通过属性

进行具体的描述,其中演职员的属性有:“name”、“born”,电影的属性有:“released”、“title”以及“tagline”。

在此根据电影演职员例子查询“年老的演员出演的 2000 年左右上映的电影”,其中模糊条件分别是“年老”以及“2000 年左右”,将它们分解后逐个进行分析。

2.1 简单条件模糊查询

其中“年老”不是精确的表述,是一种模糊性的概念,但是怎么算是年老? 以平时的生活经验可知,50 岁以下一般不属于年老,但是 50 岁以上的人该怎么判断他们哪些是年老的? 由式 3 所定义的关于“年老”的隶属函数可知,50 岁及以下可以肯定不是“年老”的人,但是对于 50 岁以上的人并不能绝对地说他是或者不是年老的人,而是要通过计算隶属函数,得到他们属于年老的“程度”是多少。

进行模糊查询一般根据这样一个过程:用户结合自己的需求输入自己所需的相关条件,根据用户输入的模糊条件找到其中模糊项相对应的隶属函数,根据隶属函数计算所得结果和查询条件进行匹配,看是否满足条件。其中对隶属函数的计算就分为两种方法,即根据文献[9-12]介绍的正向法和反向法实现模糊条件的去模糊化,然后将模糊的查询条件转换成精确的 CQL 查询,这样就可以在当前的 Neo4j 数据库中对带有模糊性的条件进行查询。

正向法:需要把所有的模糊性条件都带进相对的隶属函数中,根据隶属函数计算出所有的隶属度,系统会自动生成相应的表来存储这些结果,然后根据题意对隶属度表中的所有隶属度进行求交集或者求并集,得到总的隶属度,将这个隶属度和之前设定好的阈值进行对比,如果匹配度大于或者等于阈值,那么这个条件就是满足的。

在这边,将阈值设定为 α ,然后将所有条件选项所对应的数值带入隶属函数,计算出每个选项所对应的隶属度,具体的计算过程如下所示:

$$\begin{cases} \mu(x_1) = \alpha_1 \\ \mu(x_2) = \alpha_2 \\ \vdots \\ \mu(x_n) = \alpha_n \end{cases} \quad (8)$$

对求得的所有隶属度进行求交集或者并集得到总匹配度,具体计算方法如下:

逻辑“AND”的总隶属度计算方法为:

$$M = \min[\mu(x_1), \mu(x_2), \dots, \mu(x_n)] = \min(\alpha_1, \alpha_2, \dots, \alpha_n) \quad (9)$$

逻辑“OR”的总隶属度计算方法为:

$$M = \max[\mu(x_1), \mu(x_2), \dots, \mu(x_n)] = \max(\alpha_1, \alpha_2, \dots, \alpha_n) \tag{10}$$

然后将计算出来的总隶属度和原先设定好的阈值 α 进行比较,如果结果满足 $M \geq \alpha$,那么当前的记录是满足要求的;如果结果不满足 $M \geq \alpha$,则当前记录不满足条件。

以上方法简单易懂,计算容易,适用于只有少量数据的时候,但是对于要研究的大数据来说计算量尤其复杂,需要扫描数据库中的所有数据,并且计算出这些数据对应的模糊项的隶属度,而且最后还需要自动分配一个表来存储这些结果,表占用了很多的内存空间,会大大降低查询速度,因此更倾向于使用反向法。

反向法:同样也需要事先设定好阈值(α 截集),将阈值代入模糊查询条件所对应的隶属函数反向计算得到相应的精确的结果区间,然后用这个精确的结果区间取代原先的模糊查询条件,就可以以常规的方式在 Neo4j 中实现模糊查询。由于反向法不需要对所有的数据条件进行计算,而是可以根据隶属函数直接进行转换,对于查询庞大数据量时具有很大的优势,而 Neo4j 是区别于关系数据库的云数据库,里面数据尤其多,所以可以使用反向法进行去模糊化处理。

通过设定的 α 截集将带有自然语言表述的模糊查询语言转换成精确的查询语句,假设阈值 $\alpha = 0.5$,则有如下方程组:

$$\begin{cases} y = \mu_{old}(x) \\ y \geq 0.5 \end{cases} \tag{11}$$

求解得到的结果为: $x \in [55, 100]$ 。由此可知,年老的演员年龄为 55 到 100 岁之间。以上成功地将查询“年老”的演员这一模糊条件转换成了查询年龄在 $[55, 100]$ 这一精确的区间。去模糊化后可以对 Cypher 查询语言进行扩展来实现这一查询,具体查询语句如下:

```
查询年老的演员:  
MATCH (p:Person)  
WHERE 2017-p.born=OLD(0.5)  
RETURN DISTINCT p.name
```

其中,OLD(α) 是将 Cypher 查询语言扩展后的 API,主要为了对是否符合“年老”这一模糊条件进行判断,其中 α 是阈值,此处设置为 0.5。并且由于数据库中并没有直接给出具体的年龄,只是给了每个人的出生年,所以需要“2017-p.born”进行转换,将它转换成具体的年龄。

2.2 带有语言变量的模糊查询

还有很多时候直接给出的条件不一定就是模糊的,也有可能要查询的条件本身是精确的,但是经过一些修饰词将它模糊化了,比如要查询“大约 2000 年上映的电影”,其中 2000 年是一个具体的值,但是加上大约就将这个词模糊化了。对于“大约”这一模糊化算子,根据 1.4 节相关知识,可以用一个相似模糊关系来对模糊范围进行调节,这个相似模糊关系为式 7,其中 y 是要查询的大约年份, δ ($\delta > 0$) 是一个调节模糊范围的参数,同样假设此处的阈值是 0.5,即 $E(x, y) \geq 0.5$,计算可得具体的范围是: $x \in [y - \delta \sqrt{\ln 2}, y + \delta \sqrt{\ln 2}]$ 。在这个例子中,设定 $\delta = 10$,查询上映年份在 2000 左右这一模糊查询就可以转换成查询上映年份在 $[2000 - 10 \sqrt{\ln 2}, 2000 + 10 \sqrt{\ln 2}]$,即查询 1992 年到 2008 年上映的电影。可以对 Cypher 查询语句进行扩展来实现查询,具体语句如下:

```
查询 2000 年左右上映的电影:  
MATCH (m:Movie)  
WHERE m.released=GENERAL(0.5,10,2000)  
RETURN DISTINCT m.title
```

对于这个查询语句也使用了一个扩展的 API: GENERAL(α, δ, y),其中 α 是隶属函数的阈值, y 是要查询的条件,结果会根据这三个值的改变而进行改变。

以上两个扩展得到了两个查询条件的结果,最后需要查询演员出演的电影,而不仅仅只是将两个扩展的查询条件进行简单的求交集,需要使用 (p)-[:ACTED_IN]->(m) 语句对它们之间的关系进行限定,然后才能对演员的关系进行筛选,得到需要的结果。具体扩展的查询语句如下:

```
复合条件查询:  
MATCH (p:Person), (m:Movie), (p)-[:ACTED_IN]->(m)  
WHERE 2017-p.born=OLD(0.5) and m.released=GENERAL(0.5,10,2000)  
RETURN p.name, m.title
```

3 系统实现与实验分析

3.1 实验系统

3.1.1 实验环境

文中的原型系统使用 Java 的 Eclipse 集成环境开发完成,实现了一个可以在 Neo4j 数据库进行模糊查询的系统,具体的硬件环境如下:处理器是 Intel(R) Core(TM) i5-4590 CPU @ 3.30 GHz;8 GB RAM;

界面显示结果所使用的软件环境如下:操作系统

是 Windows 10 专业版 64 位;具体代码通过 Java 的 Eclipse 集成环境编写;数据库使用 Neo4j 3.2.6 社区版。

3.1.2 数据集

实验数据采用 Neo4j 数据库系统自带的“Movie Graph”演职员关系图,主要是数据库中的演职员和电影的例子,包含了两种节点,分别是参演电影的演员或者其他相关的工作人员,以及上映的电影。数据集中所有的关系和节点都通过属性值进行具体的描述,其中演职员“Person”的属性有姓名和出生日期,电影“Movie”的属性有电影名称、上映时间以及电影简介。节点通过关系连接,并且节点通过标签进行说明,其中标签是没有属性的。

在 Neo4j 图数据库中进行模糊查询实现的原型系统如图 2 所示。

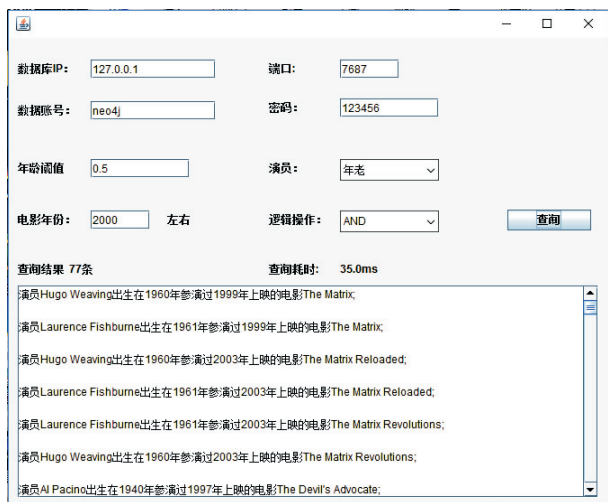


图 2 原型系统

3.2 实验分析

基于 Neo4j 图数据库的模糊查询扩展方法主要是对 Cypher 查询语言进行扩展,根据隶属函数和 α 截集相关知识在查询语言上扩展一个 API,使用户可以自行输入自己想要查询的条件,通过这个 API 连接到数据库中,将模糊查询条件经过去模糊化,转换成精确的取值区间,然后在数据库中进行查询,得到所需要的结果。为了验证原型系统的性能,选取不同的查询条件分别在 Neo4j 数据库和设计的模糊系统中进行查询,对它们的响应时间进行分析比较。

实验分别从常规条件下的查询语句和模糊条件下的查询语句中随机选择 4 条查询用例进行测试,相应的查询语句如表 1 所示。

对表 1 的测试用例分别进行系统响应时间测试。由于系统性能的影响,所有的测试用例都查询十二次,然后去掉最长时间和最短时间,取十次响应时间的平均值进行

表 1 测试用例

模糊查询	对应的常规查询
年老的演员出演的 2000 年左右上映的电影($\alpha=0.5$)	1962 年以前出生的演员出演的 1992 年到 2008 年之间上映的电影
年老的演员出演的 2000 年左右上映的电影($\alpha=0.8$)	1957 年以前出生的演员出演的 1992 年到 2008 年之间上映的电影
年轻的演员出演的 2000 年左右上映的电影($\alpha=0.8$)	1989.5 年以后出生的演员出演的 1992 年到 2008 年之间上映的电影
年轻的演员或者 2000 年左右上映的电影($\alpha=0.8$)	1989.5 年以后出生的演员或者 1992 年到 2008 年之间上映的电影

从图 3 可以看出,文中设计的系统的查询响应时间都比数据库本身的要大,但是相差均不超过 50 ms,时间差别很小,所以使用模糊系统进行查询是可以接受的。并且从图中可以看出,第三条语句响应时间明显要小一些,是因为它的查询结果很少,只有一条,所以查询结果的数量也是影响响应时间的因素之一。总的来说,图中显示了一般的 Cypher 查询语言和扩展后的具有模糊性的系统的执行之间没有什么大的区别,在系统中使用模糊语句的额外成本与使用 NoSQL 图形数据库是一致的。

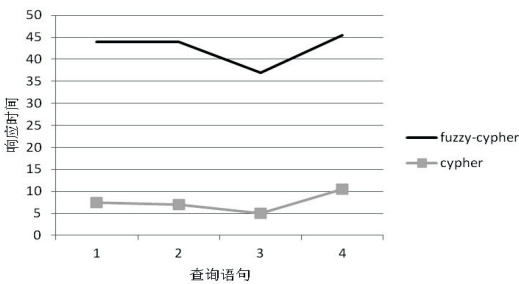


图 3 模糊系统和一般查询语言响应时间对比

4 结束语

对 Neo4j 图数据库的查询语言进行扩展使其可以进行模糊查询,有效解决了现如今大量数据情况下人们想要使用模糊的查询条件查询结果的问题。提出的模糊查询方法主要是对查询语言 Cypher 进行扩展,将这些扩展在低级 API 上实现。对于 Cypher 查询语言扩展方面主要对带有语言变量的模糊查询进行处理,说明了如何将模糊条件去模糊化转换成精确的数值区间进行查询。同时,文中也存在很多不足之处。因为对于不同的模糊集有不同的隶属函数,很难对所有的模糊集都设置一个相对较为准确的隶属函数,而且就算是同一个模糊集,由于所在条件不同,隶属函数的设定也有所区别,所以对该方法实现起来具有很大的工作量,难以运用到实际工程中。目前国内外相关研究还很少,还有很长的路要走。

参考文献:

[1] CATTELL R. Scalable SQL and NoSQL data stores[J]. ACM SIGMOD Record,2011,39(4):12-27.

[2] BARABASI A L. Linked: how everything is connected to everything else and what it means for business, science, and everyday life[M]. New York: Plume Book, 2009.

[3] ROBINSON I, WEBBER J, EIFREM E. Graph databases[J]. Oreilly Media, 2013, 7(4): 15-31.

[4] WOOD P T. Query languages for graph databases[J]. ACM SIGMOD Record, 2012, 41(1): 50-60.

[5] 王余蓝. 图形数据库 NEO4J 与关系型数据库的比较研究[J]. 现代电子技术, 2012, 35(20): 77-79.

[6] 王青松, 李 爽, 马瑞萍, 等. 基于模糊聚类分析的数据库模糊查询的研究[J]. 小型微型计算机系统, 2015, 36(6): 1199-1203.

[7] 王 慧. 关系型数据库的模糊查询方法研究与实现[D]. 南京: 南京信息工程大学, 2009.

[8] 陈逸菲, 叶小岭, 张颖超. 基于语言变量的关系数据库模糊查询[J]. 计算机工程, 2009, 35(9): 28-30.

[9] 樊新华. 基于关系数据库的模糊查询技术[J]. 计算机与数字工程, 2009, 37(10): 149-152.

[10] 郑知卉. 关系数据库模糊聚合查询方法研究[D]. 沈阳: 东北大学, 2012.

[11] CASTELLTORT A, LAURENT A. Fuzzy queries over NoSQL graph databases: perspectives for extending the cypher language[C]//15th international conference on information

processing and management of uncertainty in knowledge-based systems. [s. l.]: [s. n.], 2014: 384-395.

[12] CASTELLTORT A, MARTIN T. Handling scalable approximate queries over NoSQL graph databases: cypherf and the Fuzzy4S framework [J]. Fuzzy Sets & Systems, 2017, 5(40): 1-29.

[13] CASTELLTORT A, LAURENT A. Rogue behavior detection in NoSQL graph databases[J]. Journal of Innovation in Digital Ecosystems, 2016, 3(2): 70-82.

[14] CASTELLTORT A, LAURENT A. Fuzzy historical graph pattern matching a NoSQL graph database approach for fraud ring resolution[C]//IFIP international conference on artificial intelligence applications and innovations. [s. l.]: [s. n.], 2015: 151-167.

[15] SOUGUI I B, HIDRI M S, GRISSA-TOUZI A. No-FSQL: a graph-based fuzzy NoSQL querying model[J]. International Journal of Fuzzy Systems Applications, 2016, 5(2): 54-63.

[16] PIVERT O, SLAMA O, SMITS G, et al. SUGAR: a graph database fuzzy querying system[C]//IEEE tenth international conference on research challenges in information science. Grenoble, France: IEEE, 2016: 1-2.

[17] PIVERT O, SMITS G, THION V. Expression and efficient processing of fuzzy queries in a graph database context [C]//IEEE international conference on fuzzy systems. Istanbul, Turkey: IEEE, 2015: 1-8.

(上接第 15 页)

该优化算法主要应用于软材质实心打印材料,用于医学手术模拟训练和手动切割,因此,没有考虑模型的内部结构优化。未来需要考虑内部支撑结构的优化,从而适应更多的打印需求。

参考文献:

[1] 卢秉恒, 李涤尘. 增材制造(3D 打印)技术发展[J]. 机械制造与自动化, 2013, 42(4): 1-4.

[2] 刘利刚, 徐文鹏, 王伟明, 等. 3D 打印中的几何计算研究进展[J]. 计算机学报, 2015, 38(6): 1243-1267.

[3] 胡瑞珍, 黄 惠. 3D 打印启发下的模型实例化优化研究综述[J]. 计算机辅助设计与图形学学报, 2015, 27(6): 961-967.

[4] WANG Weiming, WANG Tuanfeng, YANG Zhouwang, et al. Cost-effective printing of 3D objects with skin-frame structures[J]. ACM Transactions on Graphics, 2013, 32(6): 177.

[5] KINDINGER J. Lightweight structural cores, specifying structural core[J]. ASM Handbook, 2001, 21: 180-183.

[6] LU Lin, SHARF A, ZHAO Haisen, et al. Build-to-last;

strength to weight 3D printed objects[J]. ACM Transactions on Graphics, 2014, 33(4): 97.

[7] CHEN Y. 3D texture mapping for rapid manufacturing[J]. Computer-Aided Design and Applications, 2007, 4(1): 761-771.

[8] 傅驰林, 刘 斌. 3D 打印模型节材优化[J]. 计算机辅助设计与图形学学报, 2017, 29(4): 742-750.

[9] 沈振宏. 三维打印支撑结构的研究与实现[D]. 南京: 南京航空航天大学, 2016.

[10] 陈 岩, 王士玮, 杨周旺, 等. FDM 三维打印的支撑结构的设计算法[J]. 中国科学: 信息科学, 2015, 45(2): 259-269.

[11] 欧立松. 面向三维打印的几何模型后处理技术研究[D]. 南京: 南京航空航天大学, 2015.

[12] 吴芬芬. 3D 打印物体的稳定平衡优化[D]. 合肥: 中国科学技术大学, 2016.

[13] 徐文鹏. 3D 打印中的结构优化问题研究[D]. 合肥: 中国科学技术大学, 2016.

[14] 李小丽, 马剑雄, 李 萍, 等. 3D 打印技术及应用趋势[J]. 自动化仪表, 2014, 35(1): 1-5.

[15] 王运赣, 王 宣. 三维打印技术[M]. 武汉: 华中科技大学出版社, 2013.