目录

[第一章 绪论 2](#_Toc481005581)

[第二章 Neo4j数据库 3](#_Toc481005582)

[第三章 Neo4j的查询语言Cypher 5](#_Toc481005583)

[第四章 基于Neo4j的图算法性能研究 7](#_Toc481005584)

[第五章 结论 7](#_Toc481005585)

[致谢词 7](#_Toc481005586)

# 第一章 绪论

* 1. 研究背景及课题意义

随着网络时代的到来，现代技术飞速发展，数据达到了前所未有的爆炸式增长，而随之出现许多问题，比如服务器的容量无法承载巨大的数据业务量，传统的关系型数据库无法满足多种不同形式和内容的信息的存储等等。随着数据量的增加，数据的交叉互联越来越多，数据的结构越来越多样化，传统的数据处理方法和数据查询技术显得力不从心，因此如何更好地处理大数据已经引起了学术界和工业界的广泛关注。

在大数据的时代，新兴技术如云计算、社交网络、物联网等迅速发展，传统的关系型数据库RDBS在大规模的数据存储上需要较高成本的系统维护，容易造成数据丢失等问题，建立在传统关系型数据库上的系统的伸缩性和可用性较明显不足，语义描述的准确性、推理能力较差，NoSQL数据库应运而生。

NoSQL即是“Not Only SQL”，是一种非关系型数据库，目前主要分为四大类：键值存储数据库、列存储数据库、文档型数据库以及图形数据库，NoSQL分为四层：接口层、数据逻辑模型层、数据分布层以及数据持久层。面对性能和可扩展性的问题，NoSQL数据库并没有一个统一的架构，基本放弃了关系模型处理交叉互联数据的方式，为海量数据的存储提供了很好的解决方法。NoSQL支持分布式存储，克服了传统关系型数据库的缺点，可部署在廉价的硬件上，透明地扩展节点，这吸引了学术界和工业界的关注。学术界的主要研究侧重于如何在大规模图数据库中实现查找等操作。工业界则主要研究如何将实际的关系（例如社交网络、股票分析、网络舆情等）转换为图的操作，并利用新颖的图算法实现具体的功能。

图形数据库作为NoSQL的一个分支，相对于其他分支，它更适合用来表达图结构的数据，而Neo4j是目前最流行的图形数据库，在社会计算、物流、社交关系、数据中心管理、产品目录等领域上都能适用，是处理复杂数据的理想选择。综上，本选题将研究分析在大数据时代下，新型图数据库Neo4j是如何应用图形理论来存储与表达实体之间的信息及关系，使得大数据的存储与搜索更加直观、效率更高。

* 1. 国内外研究状况

大数据是指利于最先进的工具及方法，进行获取、存储、分析、解释的庞大且复杂的数据，随着大数据的迅速发展，社交网络、生物工程、智能家居以及电子商务等发生了翻天覆地的变化。大数据作为云计算、虚拟技术和数据中心的逻辑衍生物，现如今已经渗透到国家、社会以及各个行业，在政治军事、经济金融、信息通讯以及环境生态中成为了十分重要的生产因素。大数据对数据库技术的要求越来越高，如高性能、大容量、易扩展，这对NoSQL系统的发展起到了一定程度的促进作用。

在20世纪90年代初，NoSQL起源于Berkeley的原型系统，Berkeley数据库是一个由Key-Value键值对组织的哈希数据库，这种数据库的主要设计宗旨是稳定、简单、灵活和高性能。NoSQL于2007年开始得到快速发展，这一年里，谷歌和亚马逊分别发不了关于BigTable的报告和关于Dynamo的说明文档，阐述了当时的NoSQL数据库的设计宗旨。NoSQL发展至今，已出现了十多种较为流行的NoSQL产品，比如Neo4j、CouchDB、Redis、MongoDB、Riak、Membase、Cassandra、HBase等，而我国的NoSQL领域也从2009年开始活跃起来，代表性的有豆瓣开源数据存储系统BeansDB、人人网的UCG海量存储系统Nuclear和盛大创新院的TCDatabase等。相比起国外的开发的NoSQL，国内的开发尚不成熟，在实际生产中并没有得到很好地应用。因此，国内比较大型的互联网企业更倾向于选择国外的NoSQL产品，比如淘宝、大众点评和视觉中国选用MongoDB，新浪采用Redis，而盛大在MongoDB的基础上设计并推出了MongoDB云服务,被称为“盛大云”。

图形数据库是NoSQL的一个分支体系，是将社交关系等数据描述为点（Vertex）和边（Edge）及他们的属性（Property），每一张图（Graph）都可以看做是一个结构化数据。随着社交网络、电子商务等领域的发展，在使用关系型数据库时会遇到许多非常复杂的设计问题，深度遍历和查询大量复杂互相关联的数据的效果并不理想，而选用图数据库的特点刚好弥补了关系型的缺陷。根据美国数据库知识网站DB-engines发布的2015年3月数据库流行度排行榜，自2013年2月开始，图数据库的增长幅度高达5倍，而在数据库流行度排行榜中，位列前三的图型数据库分别是Neo4J（No.23），Titan（No.53），OrientDB（No.63）。

* 1. 主要研究内容

本选题主要研究分析在大数据时代下，新型图数据库Neo4j是如何应用图形理论来存储与表达实体之间的信息及关系，使得大数据的存储与搜索更加直观、效率更高。具体研究内容如下：

（1）搭建仿真平台，部署Neo4j服务器，通过图表建模研究Neo4j的数据模型；

（2）在Neo4j服务器模式下使用查询语言Cypher；

（3）在嵌入式模式下，研究使用Neo4j API，搭建测试环境，集成图算法，对各个算法进行功能测试与性能测试；

（4）对比Neo4j与传统关系型数据库对数据的存储能力。

1.4 论文组织结构

# 第二章 Neo4j数据库

2.1 Neo4j数据库的特点

Neo4j作为NoSQL数据库的一个重要的分支，是一个高性能的图数据库。图数据库不是存储图片的数据库，而是应用“图”这种数据结构来存储和管理数据，其数据模型主要体现为节点（Nodes）和关系（Relationships），Nodes通过Relationships所定义的关系相连起来，形成关系型网络结构。目前有几种流行的图数据库，有Neo4j，AllegroGraph，InfiniteGraph，HyperGraphDB，DEX，InfoGrid，Trinity，具体对比分析如下图：



注：“1M”是指对应图形数据库可以支持100万个免费节点。RDF图形可以被看作一种特殊属性的图形

由表可知，相比起其他的图形数据库，Neo4j有较大的优势，支持Java，开发手册质量较好，便携性良好，开放源代码，能扩张亿级以上的节点和关系。Neo4j作为图形数据库，并不是将数据保存在表或者集合中，而是将节点及节点之间的关系保存起来，除此之外，节点还可以设置零或者多个标签，每个关系都对应着一种类型，并且关系是从一个节点指向另一个节点。

Neo4j自带一套易于学习的查询语言Cypher，不使用schema，可以满足任何形式的需求；与传统的关系型数据库相比，对于高度关联的数据的查询速度要快上许多；Neo4j的实体与关系结构十分自然，切合人类的直观感受；它支持兼容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；它还提供了一个可视化的查询控制台。因此，本课题选择Neo4j来存储分析数据及集成和研究经典图算法。

2.2 Neo4j的系统架构

2.2.1 系统结构

Neo4j的系统结构如下图所示：



Neo4j的图形数据以记录文件的形式存储在磁盘上。为了提高检索速度，Neo4j使用了一个两层缓存机制：文件系统缓存和对象缓存。通过访问文件系统缓存而不是直接访问一个旋转的硬盘，这会大大提升性能。在硬盘上，关系包含大部分信息，每个节点只保存其关系的第一个引用。而在对象缓存中，节点保存所有关系，并且关系变得非常简单，只需保留其所有属性。一个节点的关系根据RelationshipType进行分类，并且很好地快速遍历。事务/逻辑日志包含提交了的事务文件，从而确保与ACID的兼容，在需要时提供恢复数据库的能力。HA模块描述了系统的群集功能，企业版的Neo4j具有HA部件，其能力主要表现在硬件出现故障时有较强的应变能力和容错能力，并且它提供了横向扩展Neo4j的方法读密集型数据的场景。在Neo4j结构中处于顶层的是编程API，包括Cypher、遍历API、核心API。核心API是必需的API，它期望用户对低层图形数据的布局有一个准确的理解，并提供显式的指令指出如何与低层图形交互。遍历API对开发者来说是比较友好的API，用户可以直接给出目标对图进行遍历。而Cypher则是最直观和最友好的声明的、人性化图形查询语言。

2.2.2 储存结构

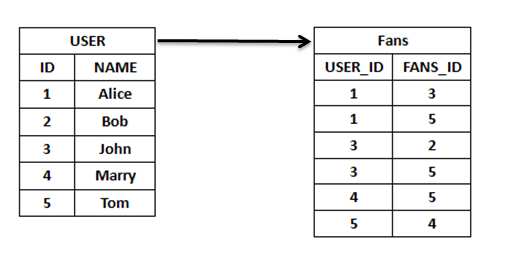
Neo4j数据库中的节点数据存储在名为neostore.nodestore.db的文件中。大多数Neo4j存储文件中，节点存储是一个固定大小的记录存储，其中每个记录长度为9字节。固定的记录大小可以快速查找存储文件中的节点，例如一个ID为100的节点，那么它的记录在文件中开始的第900个字节，因此数据库可以直接计算记录位置，时间空间复杂度为O（1），而不需要以O（log n）为代价执行搜索。在一个节点记录中，第一个字节是用于标识数据库当前是否使用该记录或是否可以回收存储新记录，接下来的四个字节是此节点的第一个关系连接，最后四个节点字节是此节点的第一个属性的ID。

Neo4j数据库中的关系数据存储在磁盘上名为neostore.relationshipstore.db的文件中。与节点存储文件一样，关系存储是固定大小的记录存储，每个记录长度为33字节，每个记录包含该记录是否可用的标识、关系的头节点、关系的尾节点、关系类型、头节点的前一个关系ID、头节点的后一个关系ID、尾节点的前一个关系ID、尾节点的后一个关系ID、关系的最近属性ID。

Neo4j数据库中的属性数据存储在磁盘上名为neostore.propertystore.db的文件中，它还包含属性类型和属性索引文件（neostore.propertystore.db.index）。一个属性默认占41字节，它包含该记录是否可用的标识、前一个属性ID、后一个属性ID和属性块，属性块格式为属性类型（占8B）+属性值（占8B），属性键与属性值分别存储在不同的文件中，属性记录属于动态存储格式。

2.3 Neo4j的数据模型

Neo4j是一个无架构的数据库，不同于传统的关系型数据库管理系统，它的数据模型即存储在数据中，无需明确定义表和关系，一个节点可以有任意属性，并且任何节点均可以与其他任何节点建立任意关系。图是一组节点和连接这些节点的关系，图形数据存储在节点和关系在属性的形式，属性是键值对表示数据。在图形理论中，可以表示一个带有圆的节点，节点之间的关系用一个箭头标记表示，因此，Neo4j对数据库将要存储的数据可以很容易地进行一致性的描述，以期望的方式构架查询。以一个简单的社交网络作为例子，一个用户有一个或多个粉丝，在传统关系型数据库中，会有两个表用来存储数据，如下：



在Neo4j图数据库中，只需要将用户描述为节点，将粉丝跟随用户描述为节点之间的关系，可以得到如下图的数据模型：



对比以上两张图， Neo4j的数据模型具有更强的描述性，我们可以更加清晰的看出谁是谁的粉丝。这个社交网络中查询Alice的粉丝，使用传统的关系型数据库进行查询，可以很容易地得到结果是John和Tom，当查询Alice的粉丝的粉丝时，则问题变得复杂起来，而将查询深度延伸到3、4或5时，由于链表递归查询，时间空间复杂度变得非常高，查询的代价十分高昂。但是，在Neo4j的图数据模型中，节点之间的关系即存在的路径，查询或者遍历都是涉及到路径，故查询的代价较低。总之，Neo4j是描述性的，而不是规定性的，节点作为数据模型的实体，而关系是节点之间的连接。

2.4 Neo4j的应用模式

Neo4j有两种应用模式，一种是嵌入式模式，一种是服务器模式。

2.4.1 嵌入式模式

Neo4j的嵌入式模式，并不是指硬盘上的实际物理数据库嵌入到应用程序中，而是嵌入到Neo4j引擎（类和相关的过程）直接运行和管理Neo4j数据库。目前最通用的嵌入式情景涉及在一个基于Java的应用中直接嵌入Neo4j，如下图所示：



嵌入式模式需要适当的JAR文件捆绑，或者当应用程序开始时，这个JAR文件可用，通过实例化GraphDatabaseService接口上的适当实例，获得对库的访问后，应用程序就能够利用引用与Neo4j交互。在项目中需要使用Maven来创建对Neo4j的依赖，Maven是一个基于Java的开发框架，用于管理代码的设置、创建、编译以及依赖等解决方案的许多方面。以下为嵌入式Neo4j依赖的程序：

<dependencies>

<dependency>

<groupId>org.neo4j</groupId>

<artifactId>neo4j-slf4j</artifactId>

<version>3.1.3</version>

</dependency>

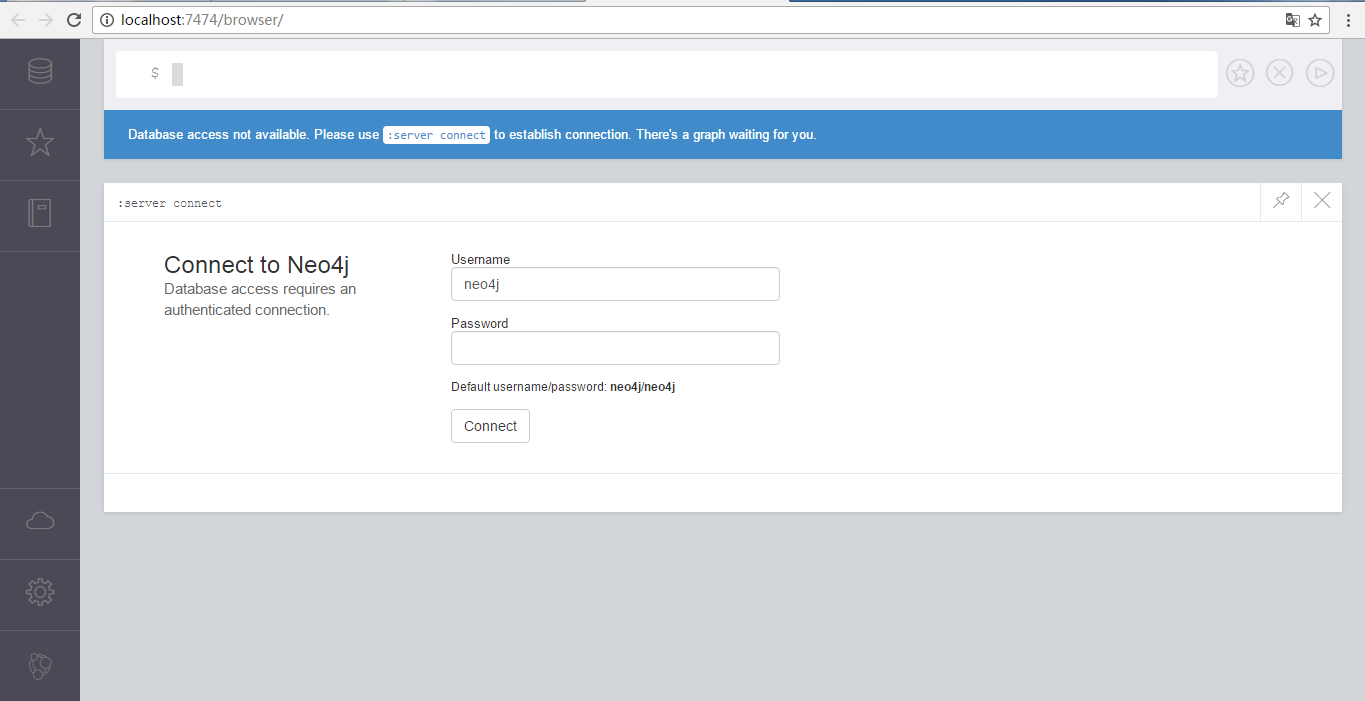
</dependencies>

</project>

2.4.2 服务器模式

与嵌入式模式不同，在服务器模式下运行Neo4j涉及要使用包含其自己的自身进程的Neo4j数据库进行访问和处理的所有类和逻辑，并且与使用它的客户端完全分离。例如，通过很多其他基于服务器的设置，客户端需要一种与服务器进程进行交互的机制，在Neo4j的情况下，这是通过使用明确定义的可扩展的基于HTTP的REST API来实现的。

对于服务器模式的安装，首先要确保机器上已经安装了最新的Oracle 7 JDK，并配置好环境变量；其次，在官网（http://neo4j.org/download）下载Neo4j的安装文件，安装好服务器，选择默认的数据库路径，启动服务器，通过访问http://localhost:7474，可看到Neo4j服务器的图形界面，本课题使用的版本为3.1.3，如下图：



2.5 本章小结

本章主要通过对比NoSQL中较流行的几种图数据库，阐述了Neo4j图数据库有完整的ACID支持、高可用性、轻易扩展到上亿级别的节点和关系、通过遍历工具高速检索数据等优点。主要介绍了Neo4j的结构，包括系统结构和存储结构。重点介绍了Neo4j的数据模型，举例说明了与传统关系型数据模型的区别。除此之外，还介绍了Neo4j的两种应用模式为嵌入式和服务器模式及其安装。

# 第三章 Neo4j的查询语言Cypher

3.1 Cypher语句的基本句法

Cypher对于neo4j数据库就相当于SQL对于MySQL数据库，Cypher是neo4j的查询语言

，它使用图形模式匹配作为主要的机制作图形数据的选择、插入、更新、删除，适合于开发者与在数据库上做点对点模式查询的专业操作人员。Cypher语言是基于英语单词和灵巧的图解，它通过一系列方法和建立于确定的实践为表达查询而产生。许多关键字如like、order by是受SQL语言的启发。其模式匹配的表达式则来自于SPARQL。除此之外，正则表达式匹配实现实用Scala programming 1anguage语言。Cypher作为一种声明式的语言，对比命令式语言（如Java）和脚本语言（JRuby和Gremlin），它的关注点不是怎么去做，而是在于从图中如何查询。

Cyhper句法组成有几个部分，如下表：

|  |  |  |
| --- | --- | --- |
| 句法 | 举例 | 备注 |
| START:查找图形中的起始节点。 | start n = (1)  match (n)-[:CONNECT]->(X)  return x; | 找到id为1的节点，使用n作为标识，匹配与节点n具有CONNECT关系的节点x，返回所有的x。 |
| MATCH:匹配图形模式，可以定位感兴趣数据的子图形。 | Match (n: people)  where n.age = 20  return n; | 找到people中年龄等于20的节点，返回该节点。 |
| RETURN: 返回感兴趣的结果。 | Start n = node(1)  return n; | 找到id为1的节点，返回该节点。 |
| WHERE:基于某些标准过滤数据。 | start n = (2,1)  where (n.name=”Tom” and n.age<20) or not (n.name=”Tom”)  return n; | 找到id为1和2的节点，匹配出名字是“Tom”且年龄小于20的节点或者名字不是“Tom”的节点。 |
| CREATE:创建节点或者关系。 | create (n : people { name:’Tom’, age:’20’, sex:’male’})  return n; | 创建一个名字为“Tom”、年龄为“20”、性别为“男”的节点。 |
| DELETE:删除节点、关系或者属性。 | start john = node(1)  delete john; | 找到id为1的节点，并删除该节点。 |
| SET:设置属性的值。 | start tom = node:users (name=”Tom”)  set tom.birth=1993; | 找到名字为“Tom”的节点，设置其出生的年份为1993。 |
| WITH:将查询切分一个成多个不同的部分。 | match (n)  where n.name starts with ‘t’  return n; | 匹配名字以t开头的节点，返回该节点。 |
| 聚合函数 | start n=(1,2,3)  return avg(n.weigh); | 找出id为1,2,3的节点，返回它们的权值的平均值。 |
| 排序 | start n=(1,2,3)  return n order by n.weigh DESC; | 找出id为1,2,3的节点，按照它们的权值降序排列 |
| 分页 | start n=(1,2,3,4,5)  return n order by n.weigh skip 1 limit 2; | 找出id为1,2,3,4,5的节点，跳过第一个节点，限制返回的节点数为2，所以返回节点2,3。 |

3.2 使用Cyhper处理图形数据库

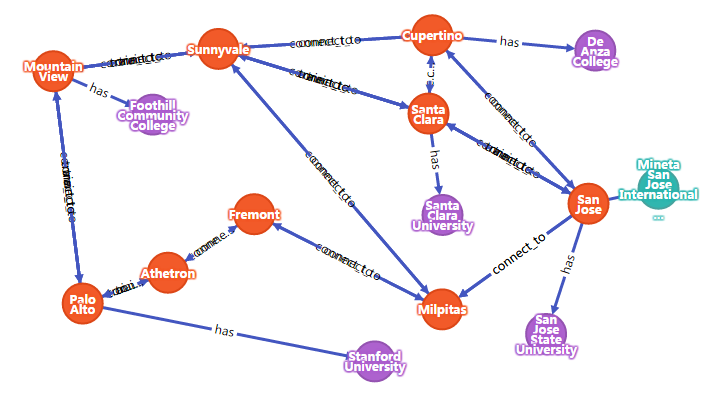
3.2.1 数据模型

在互联网时代，许多人每天都在使用地图的服务，查询从A点到B点的最佳路径，哪种出行方式是最快或者最便宜的。因此本小节通过建立简单的地图数据模型，使用cypher来解决实际问题。该地图的数据模型如下图：



注：has指城市中有学校或者机场或者体育馆；train\_to指城市之间坐火车的距离；connect\_to指城市之间有连接。

创建的城市地图如下：

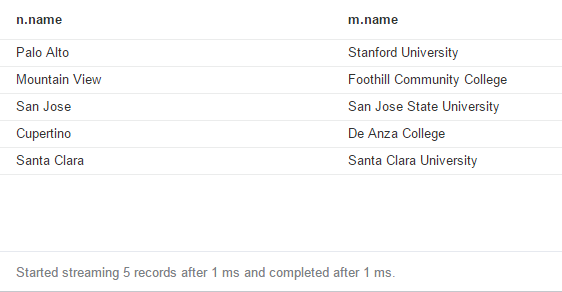


3.2.2 找出所有有学校的城市

执行的语句为：

MATCH (n:City)-[:has]->(m:School) RETURN n.name, m.name

执行的结果为：



由上图可知，有五个城市有学校，所消耗的时间为2ms。

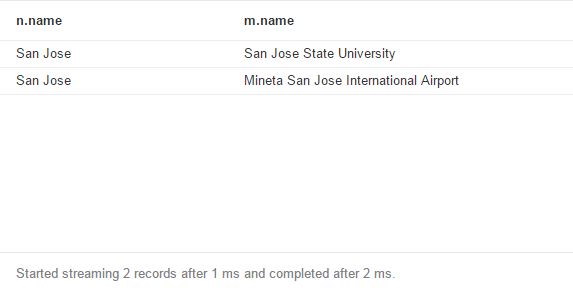
3.2.3 找出一个城市拥有的公共设施

执行的语句为：

MATCH (n {name: 'San Jose'})-[r:has]->(m)

RETURN n.name, m.name

执行的结果为：



由上图可知，San Jose这个城市有学校和机场，所消耗的时间为3ms。

3.2.4 找到从A城到B城的最短路径

MATCH p = allShortestPaths((s {name: 'Palo Alto'})-[r:connect\_to\*..5]->(d {name: 'Milpitas'}))

RETURN NODES(p)



由上图可知，从Palo Alto到Milpitas有两条最短路径，所消耗的时间为160ms。

3.2.5 找出从城市A到城市B的最短路线

执行的语句为：

MATCH p=(a {name: 'Palo Alto'})-[r\*2..5]->(b {name: 'Milpitas'})

WHERE NOT((a)-->(b))

WITH p, relationships(p) AS rcoll

RETURN p, reduce(totalDistance=0, x IN rcoll| totalDistance + x.distance) AS totalDistance

ORDER BY totalDistance

LIMIT 1

执行的结果为：



由上图可知，从Palo Alto到Milpitas的最短路线为18.3公里，所消耗的时间为226ms。

3.3 本章小结

本章主要介绍了Cypher查询语句的句法，举例介绍了句法的实际使用，并设计一个地图示例，使用Neo4j服务器进行图操作。

# 第四章 基于Neo4j的图算法性能研究

4.1 图的基本概念

4.2 图形遍历的算法

## 概述

图的遍历是指从图中的任一顶点出发，对图中的所有顶点访问一次且只访问一次。图的遍历操作和树的遍历操作功能相似。图的遍历是图的一种基本操作，图的其它算法如求解图的连通性问题，拓扑排序，求关键路径等都是建立在遍历算法的基础之上。

由于图结构本身的复杂性，所以图的遍历操作也较复杂，主要表现在以下四个方面：  
① 在图结构中，没有一个“自然”的首结点，图中任意一个顶点都可作为第一个被访问的结点。  
② 在非连通图中，从一个顶点出发，只能够访问它所在的连通分量上的所有顶点，因此，还需考虑如何选取下一个出发点以访问图中其余的连通分量。  
③ 在图结构中，如果有回路存在，那么一个顶点被访问之后，有可能沿回路又回到该顶点。

④ 在图结构中，一个顶点可以和其它多个顶点相连，当这样的顶点访问过后，存在如何选取下一个要访问的顶点的问题。

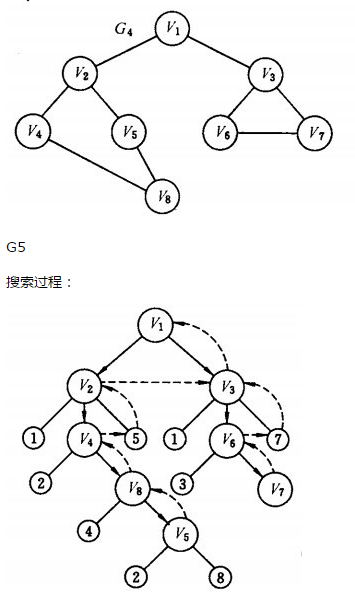
图的遍历通常有深度优先搜索和广度优先搜索两种方式，他们对无向图和有向图都适用。

深度优先搜索

深度优先搜索（Depth\_Fisrst Search）遍历类似于树的先根遍历，是树的先根遍历的推广。

假设初始状态是图中所有顶点未曾被访问，则深度优先搜索可从图中某个顶点发v 出发，访问此顶点，然后依次从v 的未被访问的邻接点出发深度优先遍历图，直至图中所有和v 有路径相通的顶点都被访问到；若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作起始点，重复上述过程，直至图中所有顶点都被访问到为止。

以如下图的无向图G5为例，进行图的深度优先搜索：



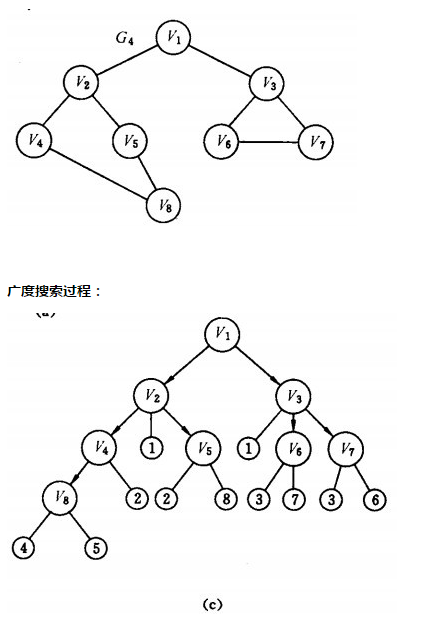
假设从顶点v1 出发进行搜索，在访问了顶点v1 之后，选择邻接点v2。因为v2 未曾访问，则从v2 出发进行搜索。依次类推，接着从v4 、v8 、v5 出发进行搜索。在访问了v5 之后，由于v5 的邻接点都已被访问，则搜索回到v8。由于同样的理由，搜索继续回到v4，v2 直至v1，此时由于v1 的另一个邻接点未被访问，则搜索又从v1 到v3，再继续进行下去由此，得到的顶点访问序列为：

http://my.csdn.net/uploads/201207/03/1341309848_6601.jpg

广度优先搜索（Breadth\_First Search） 遍历类似于树的按层次遍历的过程。

假设从图中某顶点v 出发，在访问了v 之后依次访问v 的各个未曾访问过和邻接点，然后分别从这些邻接点出发依次访问它们的邻接点，并使“先被访问的顶点的邻接点”先于“后被访问的顶点的邻接点”被访问，直至图中所有已被访问的顶点的邻接点都被访问到。若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作起始点，重复上述过程，直至图中所有顶点都被访问到为止。换句话说，广度优先搜索遍历图的过程中以v 为起始点，由近至远，依次访问和v 有路径相通且路径长度为1,2,…的顶点。

对图如下图所示无向图G5 进行广度优先搜索遍历：



首先访问v1 和v1 的邻接点v2 和v3，然后依次访问v2 的邻接点v4 和v5 及v3 的邻接点v6 和v7，最后访问v4 的邻接点v8。由于这些顶点的邻接点均已被访问，并且图中所有顶点都被访问，由些完成了图的遍历。得到的顶点访问序列为：

v1→v2 →v3 →v4→ v5→ v6→ v7 →v8

和深度优先搜索类似，在遍历的过程中也需要一个访问标志数组。并且，为了顺次访问路径长度为2、3、…的顶点，需附设队列以存储已被访问的路径长度为1、2、… 的顶点。

4.3 Neo4j图算法的研究

4.4 实验环境

4.5 实验结果

4.6 实验分析

# 第五章 结论

# 致谢词

时光如梭，不知不觉，整个毕业设计已经走到了今天，好像，毕业也不远了。首先，在这里我要感谢我的毕设老师，风里雨里，一路有你。在毕业设计刚开始的时候，我就是一只懵懵懂懂的小菜鸟，从选题，老师就给了我很多专业的指导和介绍工作，是老师的一路领航，我才能一路跌跌撞撞中顺顺利利走到今天。老师的严格要求，我才能从一开始的一无所知，扎扎实实地一步一个脚印地完成我的毕业设计。再次，毕业设计的结束，也意味着大学生涯已经走到了尾声，在这里，故往昔，感慨千万，铭感五内，从刚进入校园的青涩懵懂，到现在能够独当一面走向社会，离不开4年的同窗，热心的学长学姐，老师的辛勤教诲，更离不来深大的润物无声。是你们，给了我一个平台，让我有梦，敢去飞。充实、快乐的大学四年即将结束，感谢一路的陪伴，祝老师身体康健，祝同学前程似锦。风里雨里，我心永远在深大这片土地里。