物联网环境中数据存储与查询机制研究

何炎祥[1] 喻 涛[1] 陈彦钊[2] 李清安[1] 范通让[3]
(武汉大学计算机学院 武汉 430072)[1]
(高效洁净机械制造教育部重点实验室(山东大学) 济南 250061)[2]
(石家庄铁道大学信息科学与技术学院 石家庄 050043)[3]

摘 要 物联网中存在射频识别设备、传感器、智能嵌入设备等数目众多的异质设备,运行着用于标识、感知、处理和传送信息的各类服务,因而存在大量、非连续且时间敏感的数据。这类数据对保证物联网产业链的正常运转非常重要,但现有的数据管理方法不能有效解决其存储与查询问题。基于此,针对物联网数据多维、多态的特征,在面向服务的物联网数据管理框架上,以都柏林元数据标准为基础,制定了支持学习的物联网数据多级元数据标准。设计了从XML元数据描述到关系数据库存储的数据处理流程,提出了优化的数据存储与查询方案。最后对提出的存储与查询方案进行了验证。

关键词 物联网,元数据,存储,查询
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.3.038

Data Storage and Query in Internet of Things

HE Yan-xiang[1] YU Tao[1] CHEN Yan-zhao[2] LI Qing-an[1] FAN Tong-rang[3]
(School of Computers, Wuhan University, Wuhan 430072, China)[1]
(Key Laboratory of High Efficiency and Clean Mechanical Manufacture (Shandong University), Ministry of Education, Jinan 250061, China)[2]
(School of Information Science and Technology, Shijiazhuang Tiedao University, Shijiazhuang 050043, China)[3]

Abstract Internet of Things includes radio frequency identification devices, sensors, smart embedded devices and other heterogeneous devices, in which the services used to mark, perceive, treat, and transmit information are running. So there are a large amount of non-continuous and time-sensitive data. These kinds of data are very important to ensure the normal operation of the industrial chain. However, the existing data management technologies can not fit the need of storage and query of these data. For multidimensional and polymorphic data and data with other features in the Internet of Things, we proposed a multi-level metadata standard with learning support based on service-oriented data management framework of Internet of Things and Dublin metadata standard. Then we designed a process from XML metadata description to data stored in relational databases. Optimized data storage and query programs were proposed. Finally we verified the proposed storage and query schemes.

Keywords Internet of things, Metadata, Storage, Query

1 引言

物联网[1]的出现为智能家居、智能医疗、智能城市、智能交通等众多应用领域的迅猛发展奠定了基础。由于物联网中拥有射频识别设备、传感器、智能嵌入设备等数目众多的异质设备,运行着用于标识、感知、处理和传送信息的各类服务,因此物联网中各节点所采用的通信标准和实现技术存在巨大的差异。这将导致分布在不同地点的各节点不断产生大量、非连续且时间敏感的数据。这类具有多维多态属性数据对保证物联网产业链的正常运转非常重要,但现有的数据管理方法不能有效解决其存储与快速查询问题[2]。因而设备与技术异质性和数据多维多态属性的处理,成为物联网数据管理领域研究的重要内容[3]。

本文提出了基于物联网数据多级元数据标准的优化数据存储与查询方案。其通过改进的XML树匹配算法提高了数据查询效率与查询精度,实现了对多维多态数据压缩存储与快速查询的优化设计,有效改善了物联网中多维多态数据存储量大且查询速度慢的问题。

到稿日期:2014-05-11 返修日期:2014-08-11 本文受国家自然科学基金"可信软件基础研究"重大研究计划重点项目(91118003),国家自然科学基金面上项目(60873208,61170022,61373039,61373160),河北省自然基金(F2009000927),河北省科技攻关重点项目(10213517D)资助。
何炎祥(1952-),男,博士,教授,主要研究方向为软件工程、可信软件,E-mail:yxhe@whu.edu.cn;喻 涛(1982-),男,博士生,主要研究方向为物联网、高性能网络;陈彦钊(1986-),男,博士生,主要研究方向为分布式数据处理、机电系统与制造过程虚拟仿真,E-mail:376229402@qq.com;李清安(1986-),男,博士,讲师,主要研究方向为编译技术、嵌入式系统;范通让(1965-),女,博士,教授,主要研究方向为网络技术与信息处理,E-mail:fantr2009@126.com。

目前多维多态数据的研究在地理信息系统中比较普遍[4,5]，但地理数据大多用以描述地理空间信息，数据维数十分有限，一般不超过四维，而物联网数据复杂得多，不仅包含空间维，还包含多种状态维，特别是对时间敏感的数据的处理问题。为此有学者提出了为数据添加时间标识等方法，如文献[6]利用4种存活时间来管理RFID数据，并开发了RFID数据流的时间性管理系统；文献[7]提出了传感器网络中实时数据流的有效处理方法。类似研究取得了一定的成果，然而所涉及的技术大多只是应用于某个特定领域，没有解决物联网中跨领域应用异质设备的通信及数据处理问题，适用范围和通用性有限。

随着网络信息管理的发展，XML已经演变为异构网络、异构系统间数据交换的事实标准[8]，也为物联网中复杂数据的管理提供了可能。XML作为异质数据通讯标准的研究，一直受到很多学者关注。早在2000年，文献[9]就提出了一种异质数据源的XML查询语言。近年来，XML成为传感器领域学者关注的重点，文献[10]研究了在传感器和执行器网络中，基于XML的数据交换方法；文献[11]研究了在无线传感器网络中XML的数据传输和压缩，以降低数据传输代价；文献[12]针对无线传感网络的异质数据，提出了一种XML数据绑定技术。

用关系数据库存储XML数据成为目前普遍的做法[13,14]；XML树匹配技术成为XML数据查询的重要途径[15,16]。这些都标志着XML的研究在解决设备与数据异质性方面取得了一定进展，但其研究成果还不能直接应用于物联网中多维多态数据的处理。因此，对这类数据的描述、存储、查询的语义扩展以及冗余数据排除[17]的研究，成为物联网中复杂数据管理的重要领域。

支持数据语义描述的[18]元数据技术为物联网多维多态数据的描述提供了便利，也正逐渐成为数据描述的关键技术[19]。很多国家和国际性组织都发布了实施元数据内容标准，如都柏林元数据标准[20]，并开发了多种元数据操作工具，大量基于XML描述的元数据被使用。

本文的贡献有以下几个方面：

(1)本文在课题组前期研究成果[21]所提出的面向服务的物联网管理框架的基础上，借助XML与元数据的相关技术，设计了支持学习的物联网多级元数据标准。

(2)使用XML语言进行数据的编码与描述，并用关系数据库存储数据。通过剔除无用结点、合并低效结点，极大地压缩了存储空间。

(3)提出了XML树匹配过程中加式和减式的概念，用以提高数据查询效率与查询精度，实现了数据存储与查询的优化设计。

本文第2节进行物联网多级元数据设计；第3节详细介绍数据存储的策略与方法；第4节重点介绍数据查询的策略与方法；第5节对文中设计的数据存储与查询方案进行验证；最后总结全文。

2 物联网多级元数据设计

2.1 预备知识

元数据标准[20]是描述某类资源的具体对象时所有规则的集合。它一般包括完整描述一个具体对象时所需要的数据项集合、各数据项语义定义、著录规则和计算机应用时的语法规则。

都柏林核心元数据集是为各个领域提供的一个最精简的标准集，各领域可以在其基础上进行扩展，以形成适合于当前领域的元数据集。

2.2 物联网多级元数据标准

传统数据结构很难描述具有多维多态特征的物联网中数据。在元数据结构中使用支持语义的元数据，成为一种物联网中数据描述的可行方案。用元数据对物联网中数据进行描述，需要选择或制定一种适用于物联网数据的元数据标准。物联网的外延十分广泛，各种应用领域的数据具有不同特征，归纳起来如表1所列。

表1 物联网数据种类

数据类型	类型描述	类型举例
地理数据	对象地理属性描述	大小、高度、宽度
媒体数据	对象媒体特性描述	声音、图像、视频
状态数据	对象物理状态描述	温度、湿度、电流
时间数据	对象时间状态描述	发生、有效时间

由表1可以看出，物联网中数据种类差别很大，用一种元数据标准不能对其进行通用描述。因此，本文以都柏林核心元数据为基本元数据，根据不同领域的具体应用进行扩展(见图1)。扩展元数据分为一级扩展和多级扩展。基本元数据对数据进行基本的宏观描述，而扩展元数据是在基本元数据描述的基础上，再对数据进行详细描述，这样就为异质数据的同质描述提供了可能，为物联网数据的互操作和资源的共享奠定了基础。

(1)基本元数据：在都柏林核心元数据标准基础上建立，用于描述数据的基本特征。本文对都柏林核心元数据标准中的15个要素进行剪裁，只保留9个要素：提名、主题、描述、来源、创建者、标识符、日期、类型、格式。

(2)扩展元数据：根据各应用领域的特征扩展而成，是对数据的详细描述，如地理数据、媒体数据、状态数据、时间数据等。

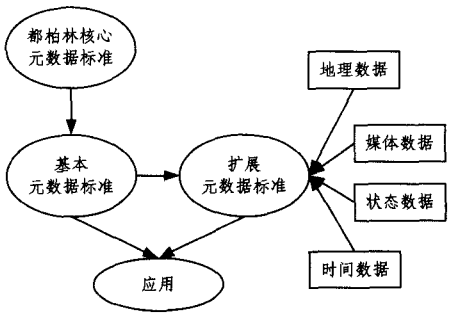

图1 物联网元数据框架

随着物联网规模、数据种类和数据量的不断膨胀，元数据需要不断地学习和扩展以适应该情况，图2描述了元数据学习的流程。当新数据被系统获取后，系统将从元数据库中调取元数据标准来描述该数据，如能完整描述，则直接对其进行描述；如不能够完整描述，则在原有元数据标准的基础上进行元数据扩展，生成目标元数据，然后使用新生成的目标元数据标准对该类数据进行描述，同时将新生成的元数据标准添加到元数据库中，完成元数据的学习过程。

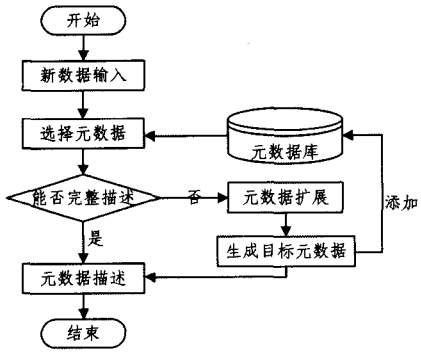

图 2　元数据学习流程

本文采用物联网多级元数据标准描述数据。其形式化描述如下：

$\sum(\{XS(element)\}\rightarrow\{XE(content)\})$

其中，XS 代表元数据的 XML Schema 元素集，XE 是一类 XML 实体。

图 3 所示为典型代码片段。

```
〈?Version="1.0" encoding="UTF-8"〉
〈xsd:complexType name="基本数据"〉
 〈xsd:sequence〉
  〈xsd:element name="标识符" type="int"/〉
  〈xsd:element name="来源" type="String"/〉
  〈xsd:element name="日期" type="date"/〉
  ……
〈/xsd:sequence〉
〈/xsd:complexType〉
……

〈基本数据〉
〈标识符〉1〈/标识符〉
〈来源〉温度传感器〈/来源〉
〈日期〉2010-10-1〈/日期〉
……
〈/基本数据〉
〈状态信息数据〉
〈温度〉18℃〈/温度〉
〈湿度〉80%〈/湿度〉
〈光强〉180cd〈/光强〉
……
```

图 3　XML Schema 描述元数据

3　数据存储

数据存储分为缓存存储和持久化存储，本文重点探讨持久化存储。持久化存储 XML 格式数据的方法有多种，关系型数据库经过多年的发展已经相对成熟，应用广泛，故选用关系数据库来存储经物联网元数据描述的 XML 数据。将关系数据库按照数据维来组织，每一个数据维映射成一个关系表，XML 文档中的元素、属性与表中的字段相对应。为每个表增加一个标识字段作为外键，以便与其它表建立关联。

3.1　数据存储策略

物联网中存在的结构化数据、非结构化数据和半结构化数据持久化存储策略各不相同。

(1)结构化数据

结构化数据具有规范的结构，数据的组织比较有规则，可直接存储到关系型数据库的表和字段中。

(2)非结构化数据

非结构化数据一般没有固定的结构，如视频、声音、图像等。这些数据不易在关系型数据库中存储，因此可将其存储在磁盘的文件系统中。

(3)半结构化数据

半结构化数据是介于结构化数据与非结构化数据之间的数据。物联网中数据多数情况下是半结构化数据，选用关系数据库和文件系统相结合的方式对此类数据进行存储。其中用关系数据库存储数据的结构化部分，用文件系统来存储数据的非结构化部分，并将非结构化数据的索引信息存储在相应的关系数据库中。表 2 给出了交通监控中某传感器的部分数据结构，其中 1－5 号字段用来直接存储结构化数据，6 号字段存储了非结构化数据，即一张图片的索引，而图片存储在磁盘文件系统中。

表 2　半结构化数据的存储结构

编号	字段名	含义
1	id	编号
2	name	传感器名称
3	location	位置
4	time	时间
5	date	日期
6	capture	图像索引信息

3.2　数据存储优化

为了适应物联网中数据的多样性与应用领域的高耦合性，制定的元数据标准尽可能保证通用。这样导致在某具体应用领域，元数据标准中有些元素无用，而有些数据项可用同一元数据元素描述的现象。若将所有元数据元素一一映射到关系数据库中，必将产生一些无效字段和利用率低的字段。随着存储数据量的增长，这些无用的字段成为沉重的冗余数据，浪费存储空间。所以需要对关系数据库模型进行优化，即剔除无用字段，合并低效字段。

合并原则如下：

(1)$(\sum Use(Node_1))/total<30\%$(视具体情况而定)

(2)$Use(Node_1)\cap Use(Node_2)=\varnothing$

(3)$\exists(Node_1\cup Node_2=Node_2)=>Node_1\backsim Node_2$

图 4 描述了数据存储优化的流程。

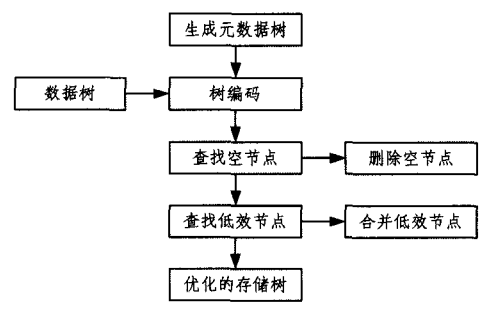

图 4　存储优化流程

(1)根据元数据描述文档，通过递归深度遍历 XML Schema 数据来创建元数据树和元数据描述的数据树。

(2)对元数据树进行编码。

对文档树中每个结点进行编码，并按照层次顺序进行遍历。每个树结点存储数据信息和编码。编码规则：将根结点编码为 1，子结点的编码＝父结点的编码＋“.”＋“i”。

(3)查找空结点,将其剔除,合并相似结点。

(4)数据存储优化算法的实现。

设元数据树为 M,数据树为 D,num 为元数据树中的结点数,初始为 0;$numTree$ 为总的数据树的数量,初始为 0;$code$[]存储元数据树编码信息,初始为 0;数组 m[]用来记录各结点被使用的次数,初始为 0;对象集合为 S。其算法实现如下:

```
L1. Fuction StorageOptimization(M,D)
L2. Input(M,D)
L3. Output S
L4. num=M. getTreeNum();//获取树结点数
L5. code[]=M. getCode();//获取元数据编码
L6. store(S,M);
L7. while(this. hasTree()){//如有数据输入,则搜索所有数据树
L8. for(int i=0;i<num;i++){//遍历 M 获得各结点被使用次数
L9. if(match(D,M. code(i))){ //在 D 中查找 M 的第 i 号结点
L10. m[i]++;
L11. }
L12. M. nextNode();
L13. }
L14. numTree++;
L15. }
L16. for(i=0;i<=countOf(S);i++){ //遍历集合 S
L17. if(m[i]==0){ //Si 未使用,删除它
L18. deleteNode(S,Si);
L19. }else if(m[i]/numTree<30%){ //对结点出现的比率小于 30%
     的低效结点进行合并
L20. seek(Si);
L21. combin(Si);
L22. deleteNode(S,Si);
L23. }
L24. }
L25 return S;
```

4 数据查询

物联网多维、多态数据的查询处理,不能直接使用关系数据库提供的查询功能,因为表和字段的名字都按照维属性来建立,可能不是语义良好的。而应用层的查询多数情况下具有语义,是基于事件或面向对象的。查询需求频繁变化,直接从关系数据库中查询,会产生大量视图,索引的维护变得复杂。所以借助具有语义描述的元数据来实现。

4.1 数据查询原理

元数据提供具有语义的描述。当处理复杂查询时,一方面可对元数据进行查询,得到具体的数据集,缩小数据查询的范围;另一方面根据元数据查询结果,转到关系数据库中进行进一步查询,以得到相应的数据,最后将查询结果进行处理和封装,返回给查询请求者。

图 5 所显示的复杂查询的流程图中,关键是进行两次树匹配。定义了加式和减式两个概念:

定义 1(加式) 用"+"表示。结点集合 $A=\{Node_1, Node_2 \cdots Node_i \cdots Node_n\}$;$A \in TreeA$;$Node_m \notin TreeA$;$Node_m$ 与 $Node_i$ 语义上具有相关性,将 $Node_m$ 并入集合 A。

定义 2(减式) 用"-"表示。结点集合 $A=\{Node_{a1}, Node_{a2} \cdots Node_{ai} Node_{aj} \cdots Node_{am}\}$,$B=\{Node_{b1}, Node_{b2} \cdots Node_{bk} \cdots Node_{bm}\}$。$A \in TreeA$,$B \in TreeB$。$TreeA$ 与 $TreeB$ 做匹配,若 $Node_{ai}$,$Node_{aj}$ 都可与 $Node_{bk}$ 匹配,则对 $Node_{ai}$,$Node_{aj}$ 进行合并或删除,使匹配结果集 C 中结点得到精简。

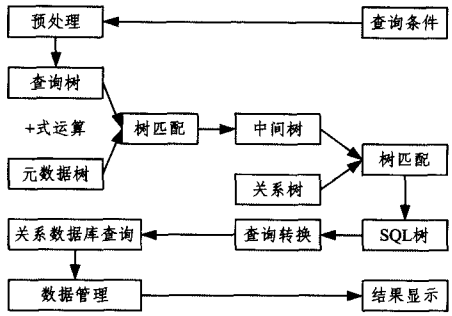

图 5 查询流程

4.2 数据查询算法

(1)对查询条件进行预处理,针对带有语义的用户查询,提取查询要素。

(2)根据查询要素生成 XML 查询树和元数据树。

(3)将查询树与元数据树进行匹配,得到一个中间树。

按照 3.2 节中的方法对文档树编码后,进行查询树与元数据树的匹配,以生成中间树。匹配时,需要进行两种运算,即加式和减式。如图 6 所示,实线箭头代表完全匹配;虚线箭头代表近似匹配;加粗结点表示完全不匹配。在树 Q 与树 M 进行匹配时,Q 中结点 q_1. q_{11},q_1. q_{12} 与 M 中结点 m_1. m_{11},m_1. m_{12} 可以完全匹配,将其完整析出,算法如下。

①减式处理。

所有数据对象都采用同一个元数据标准,各数据对象的元数据树结点都是同一元数据树的一部分,但是元数据值不同。例如描述某地区天气状况的元数据树,它可能只有地理维、状态维和时间维的信息,在查询时就没有必要对媒体维进行查询,需要去除该维。通过对查询树与元数据树匹配,进行减式运算,可剔除未匹配到的元数据树结点(如媒体维中的结点),从而降低查询冗余,节省查询空间,提高查询效率。在图 6 中树 M 中的结点 m_3. m_{31},m_3. m_{32} 与树 Q 中的任何结点都完全不匹配,因此应将其舍弃。

②加式处理。

在图 6 中,树 Q 中结点 q_1. q_{11},q_1. q_{12} 和树 M 中结点 m_2. m_{21},m_2. m_{22} 虽然不能完全匹配,但有一定的语义相关性,可近似匹配,因此对其做加式运算,将其析出,以提高查全率。例如在采集电机运行参数时,环境温度和室温两个概念在一定情况下是相同的,在查询树与元数据树中,表示这两个概念的结点就可以看做是匹配的,可做加式运算,将其析出。

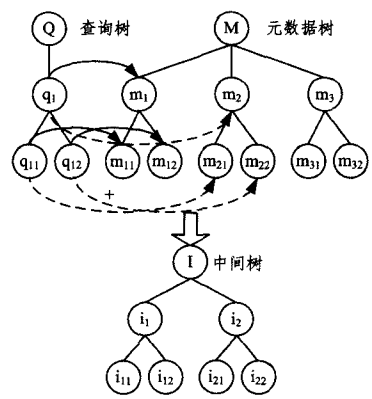

图 6 查询树与元数据树匹配

(4)中间树与关系树匹配,生成 SQL 查询树。

元数据定义时除了保证通用性和精简性,还应尽量能够覆盖物联网的各领域。而为了减少存储冗余,在映射生成关系数据库模式时,将语义相同的字段进行了合并,因此元数据的规模要比关系数据库模型大,在到关系数据库的查询转化中,需要进行处理。即,根据关系数据库生成关系树,再对(3)中生成的树进行编码,然后进行减式匹配得到最终的 SQL 查询树。如图 7 所示,树 I 中结点 $i_2.i_{21}$ 和 $i_2.i_{22}$ 存在相关性,可以与树 R 中结点 $r_2.r_{21}$ 匹配。例如,在电动机性能参数采集时,容差值和保证值存在相关性,在数据库中只存储保证值,对容差值的查询也应该被映射到对保证值的查询,因此将两种查询进行合并,即进行减式运算。这样可减少查询冗余,节省查询空间,提高查准率。

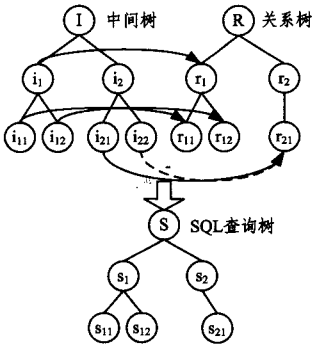

图 7　中间树与关系树匹配

(5)进行查询转化,生成 SQL 查询。

经过加式和减式匹配后得到的 SQL 查询树,构成查询数据集对应的元素和元素属性。根据该树进行查询转化,产生 SQL 查询语句,生成对关系数据库的查询。执行关系数据库查询,并返回一个结果集。

(6)将结果数据进行整理聚合。

对查询得到的数据进行分类、排序等操作,使其语义良好,并将结果返回给请求者。

(7)主要算法实现。

下面是查询树与元数据树查询匹配算法的实现。设查询树为 Q,元数据树为 M,临时空间为 $tempInfo$,计数器 $i=0$,计数器 $j=0$,数组 $m[]=$false,对象集合为 S。

```
L1. Fuction QueryOptimization(Q,M)
L2. Input (Q,M)
L3. Output S
L4. store(S,M);
L5. while(Q. subNode!=leafNode){
L6. if(match(M,Q. subNode)||approxMatch(M,Q. subNode)){//若
    完全匹配或近似匹配,则记录到 tempInfo 中,实现加式运算
L7. record(tempInfo,Q. subNode)
L8. }
L9. Q. nextNode();
L10. }
L11. if(tempInfo!=null){
L12. for(i=0;i<=countOf(S);i++){ //遍历 S
L13. m[i]==false;
L14. while(Si!=null){
L15. if(search(tempInfo,Si)){m[i]=true;//记录结点个数
L16. }
L17. Si. nextNode();
L18. }
L19. }
L20. for(i=0;i<=countOf(S);i++){//遍历 S
L21. if(!m[i]){ //m[i]=false 则 Si 中没有能与 Q 中匹配的结点,做
    减式运算,将其删除
L22. deleteNode(S,Si);
L23. }
L24. }
L25. }
L26. return S;
```

5　实验与结论

我们通过实验对本文提出的基于加式、减式运算的数据存储与查询方案进行验证,以检验优化的数据存储及其查询的效果。

(1)实验环境

实验采用一台 PC 进行测试,其主要设置如下:Interl Celeron 1.6GHz CPU,1MB 二级缓存,1GB 内存容量,Windows server 2003 操作系统,测试使用的是 SQL Server2000 数据库,采用 Tomcat5.5+Jdk1.6 搭建测试程序运行环境,采用 JSP 开发测试程序。

(2)实验设计

选取一定的数据,通过前面章节制定的元数据标准进行描述,并通过本文制定的存储和查询方案对其进行存储和查询。本节所使用案例的数据库含有 5 个数据表,这 5 个表通过元数据标准直接映射而来,其中存在大量数据冗余。数据容量分别为:basic 表 551178 条记录,geography 表 655 条记录,itime 表 550523 条记录,media 表 0 条记录,status 表 134455 条记录。

①存储设计

按照 3.2 节的数据存储优化算法,对以上 5 个表中的数据进行处理,将使用率低于 30%的字段合并,将合并后的空白字段删除,完成存储优化。比较优化前后的数据容量,以考察数据存储优化效果。

②查询设计

为了检测查询性能,分别选取规模为 500000、5000 和 500 的数据进行实验。每组实验设置不同的查询关键词数,查询关键词越多,说明查询的复杂程度越高。每组进行若干次实验,统计查询响应时间,分别计算出平均值。

(3)结论与分析

①存储结果分析

根据 3.2 节的算法对数据存储进行优化。优化效果反映为数据库存储情况。表 3 显示了各数据表中记录总量以及优化前后各表中字段数量。可以看到,经优化后,在所选实验的领域内,因为存在相似性的结点被大量合并,数据表中的字段数量明显减少。

表 3　数据存储参数

结果项	basic	geography	itime	media	status
记录数	551178	655	550523	0	134455
优化前字段数	14	8	6	6	10
优化后字段数	7	3	4	0	3
节省字段数	7	5	2	6	7

表 4 列出了优化前后数据存储空间的对比，优化前为 72.9MB，优化后为 54.7MB。由此可见，优化后存储空间较优化前节省约 25.97%，极大地降低了冗余。

表 4 数据存储空间

结果项	数据容量/Mb
优化前	72.9
优化后	54.7
节省空间	18.2

②查询性能分析

实验结果如图 8 所示。随着查询复杂程度的增加，平均查询响应时间有所增加，但在相同查询复杂程度的前提下，不同数据规模的测试结果差别不大，说明该优化算法在处理大数据量时是稳定、可行的。

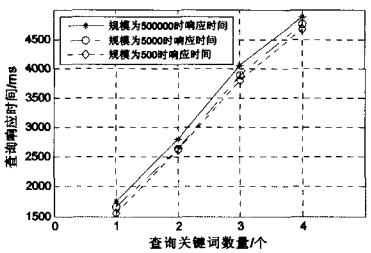

图 8 数据查询分析

结束语 本文在物联网环境下，在面向服务的数据管理框架的基础上，针对物联网数据多维、多态等特点，以都柏林元数据标准为基础，设计了适用于物联网的元数据标准，并使用 XML 技术实现了对数据的元数据描述，之后将经 XML 元数据描述的数据映射到关系数据库中。通过删除无用节点、合并相似节点对数据存储进行优化。然后，设计了数据查询方案，主要通过两次树匹配，进行加式和减式运算实现数据的查询。最后重点对存储与查询方案进行了实例验证，证明了其可行性。

参 考 文 献

[1] ITU. Internet Reports 2005: The Internet of Things-Excutive-Summary[R]. Geneve: ITU, 2005

[2] Jeffery K G. The Internet of Things: The Death of Traditional Database? [J]. IETE Technical Review, 2009(26): 311-312

[3] James A, Cooper J, Jeffery K. Research Directions in Database Architectures for the Internet of Things: A Communication of the First International Workshop on Database Architectures for the Internet of Things (DAIT 2009)[C]// Lecture Notes in Computer Science. Birmingham. UK, 2009: 225-233

[4] Luo Xiao-hua, Zheng Kou-gen, Pan Yun-he. Generalization Technology in Three Dimensi on Scaleless GIS Based on SR-Tree[J]. Chinese Journal of Computers, 2005, 28(6): 979-984

[5] Lv Zhi-han, Ma Rui-na, Fang Jing-bao, et al. Index structure for multi-scale representation of multi-dimensional spatial data in WebGIS[J]. Application Research of Computers, 2010, 27(9): 3395-3402

[6] Li Xue, Liu Jing, Sheng Quan Z, et al. TMS-RFID: Temporal management of large- scale RFID applications[J]. Springer: Information Systems Frontiers, 2009(7): 481-500

[7] Hu Kan, Liu Yun-sheng. A Submission Mechanism for Synergic Real-Time Database Transactions in Sensor Networks[J]. Chinese Journal of Computers, 2007, 30(6): 916-923

[8] Kong Ling-bo, Tang Shi-wei, Yang Dong-qing, et al. Querying Techniques for XML Data[J]. Journal of Software, 2007, 18(6): 1400-1418

[9] Chamberlin D, Robie J, Florescu D. Quilt: An XML Query Language for Heterogeneous Data Sources[C]// The Third International Workshop WebDB 2000 on The World Wide Web and Databases. 2001: 1-25

[10] Kabisch S, Peintner D, Heuer J, et al. Efficient and Flexible XML-Based Data-Exchange in Microcontroller-Based Sensor Actor Networks[C]// 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA). April 2010, Perth, WA: 508-513

[11] Hoeller N, Reinke C, Neumann J, et al. XML Data Management and XPath Evaluation in Wireless Sensor Networks[C]// 7th International Conference on Advances in Mobile Computing and Multimedia. Johannes Kepler University, 2009: 218-230

[12] Hoeller N, Reinke C, Neumann J, et al. Efficient XML Usage within Wireless Sensor Networks [C] // Proceedings of the Fourth International Wireless Internet Conference (WICON 2008). Maui, Hawaii, USA, ACM, November 2008: 17-19

[13] Amer-Yahia S, Srivastava D. A mapping schema and interface for XML stores[C]// 4th International Workshop on Web Information and Data Management. Workshop on Web Information and Data Management, 2002: 23-30

[14] Kappel G, Kapsammer E, Retschitzegger W. Integrating XML and Relational Database Systems[J]. Internet and Web Information Systems, 2004(7): 343-384

[15] Xu Ru-zhi, Qian Le-qiu, Cheng Jian-ping, et al. Research on Matching Algorithm for XML-Based Software Component Query[J]. Journal of Software, 2003, 14(7): 1195-1202

[16] Chen Yang-jun, Che Dun-ren. Efficient Processing of XML Tree Pattern Queries[J]. Journal of Advanced Computational Intelligence and Intelligent Informatics, 2006, 10(5): 1-5

[17] Cooper J, James A. Challenges for Database Management in the Internet of Things[J]. IETE Tech Rev, 2009(26): 320-329

[18] Huang Ying, Luo Xian-gang, Guo Ming-qiang. Semantic-Oriented Metadata Management Model in Semantic Grid[C]// International Conference on Environmental Science and Information Application Technology, 2009(ESIAT). 2009: 740-743

[19] Tang Liu, Fang Feng-cai. Mobile agent based metadata framework for heterogeneous wireless sensor network[C]// 2010 International Conference on Educational and Information Technology (ICEIT). 2010: 446

[20] Xiao Long, Chen Ling, Feng Xiang-yun, et al. Chinese Metadata Framework and Application [J]. Journal of Academic Libraries, 2001, 19(5): 29-35

[21] Fan Tong-rang, Chen Yan-zhao. A Scheme of Data Management in the Internet if things[C]// 2010 2nd IEEE International Conference on Network Infrastructure and Digital Content. Beijing, China, 2010: 110-114