基于本体的关系数据集成的查询处理

王进鹏 张亚非 苗 壮
(解放军理工大学指挥自动化学院 南京 210007)

摘 要 为实现异构关系数据库的语义集成,针对传统集成技术存在的问题,在对语义网等相关技术进行分析的基础上,研究基于本体的关系数据集成系统中的查询处理问题,提出了一种基于本体的关系数据库集成框架。设计了基于本体的关系数据的描述方法,使用本体作为集成的全局模式来描述关系模式的语义。设计了查询重写算法,该算法可以将基于全局模式的SPARQL查询重写为针对具体关系数据库的查询,从而实现对异构关系数据库的集成。实验表明,该算法具有良好的可扩展性。

关键词 数据集成,关系数据库,查询重写,本体
中图法分类号 TP391 **文献标识码** A

Query Processing for Ontology-based Relational Data Integration

WANG Jin-peng ZHANG Ya-fei MIAO Zhuang
(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

Abstract *In order to resolve the problem of semantic integration of heterogeneous relational databases, based on semantic Web technology, this paper focused on the query processing problem in ontology-based relational data integration. An ontology-based integration infrastructure for heterogeneous relational databases was presented. To model different databases' schemas, we proposed an ontology-based data describing method. Ontology was used as the global schema to describe the semantic of relational schemas. A query rewriting algorithm was provided, which can rewrite SPARQL query over global schema into local queries that can be executed on heterogeneous relational databases. The experiment showed that the approach we proposed has nice scalability.*

Keywords Data integration, Relational database, Query rewriting, Ontology

1 引言

数据集成的目标是从多个分布、异构和自治的数据源中集成数据,同时还保持数据在不同系统上的完整性和一致性,提供给用户一个统一和透明的数据访问接口,使用户可以有效地访问和查询各个分布的数据源[1]。传统数据集成系统基于关系模型,包括多数据库系统(Multi-database System)和联邦数据库系统(Federated Database System),主要解决的是数据在语法和结构层次上的异构问题。伴随着信息技术的飞速发展,人们对异构数据库的访问、查询和集成的需求越来越强烈,各种数据库之间的结构和语义冲突问题也变得日益明显。然而关系模型由于在语义描述能力上存在不足,并不能很好地解决数据的语义异构问题。如何支持异构数据源之间的互操作性(interoperability)这一问题依然存在[2]。

随着语义网技术的兴起,使用本体对异构关系数据库进行集成逐渐成为近年来数据集成领域研究的热点。本体作为"共享概念模型的明确的形式化规范说明"[3],能够有效地表达特定领域内的通用知识。许多不同领域(如医药、生物、金融等)都构建了领域本体投入科学研究和实际应用中。在数据集成领域,本体正受到越来越多的关注,并被应用在许多研究项目中。将本体用于数据集成系统中描述数据源的语义,可以有效解决由术语定义、概念结构和相互关系的差异所造成的异构,为异构数据的互操作提供语义层的支持。文献[4]是一篇综述,较为全面地介绍了本体在数据集成方面的应用。文献[5,6]提出了不同的基于本体的数据集成方案,使用本体描述异构数据间的语义映射,但都没有讨论具体的实现。我们之前的工作提出了一个基于本体的异构数据集成框架,并对数据源描述和查询重写做了初步的讨论[7]。在数据源描述方面,文献[8]分析了如何从关系模式中识别实体、联系、继承关系、聚类关系及基数约束等语义,给出了一个比较系统的基于关系数据库的OWL本体构建方法;Laborda等人扩展了现有的本体描述语言,提出了一种基于本体的关系数据描述语言Relational.OWL,将关系模式描述为本体[9],使用数据仓库的思想对异构的关系数据库进行集成,类似的工作还有文献[10,11]。这些系统对异构数据模式的描述采用自动翻译的方法,由于没有统一的语义模型,异构的数据源模式经翻译得到的本体在语义层仍然是异构的。查询重写方面,文献[12]给出了几个典型的针对关系模型的查询重写算法,包括:

到稿日期:2010-01-21 返修日期:2010-04-07 本文受总装备部预先研究基金(9140A06050409JB8102)和解放军理工大学预先研究基金(2009JSJ11)资助。

王进鹏(1982—),男,博士生,主要研究方向为数据集成等,E-mail:wangjinpeng1982@gmail.com;张亚非(1955—),教授,博士生导师,主要研究方向为自然语言处理;苗 壮(1978—),博士,讲师,主要研究方向为语义网、数据集成等。

桶算法、逆向规则算法和 MiniCon 算法，都是针对关系模型的查询重写；Abiteboul 等人对基于视图的查询问答的复杂度问题做了全面而深入的探讨[13]；Beeri 等人对面向描述逻辑的查询重写进行了研究[14]，Baader 等人对描述逻辑的概念重写也进行了研究[15]，但这些研究都未专门针对本体进行算法设计，无法直接应用到基于本体的数据集成中，其处理查询的方法也比较简单，没有考虑数据源与当前查询之间变量映射较为复杂的情况以及数据源的查询能力等问题。

针对传统集成技术存在的问题，在对语义网等相关技术进行分析的基础上，本文研究基于本体的关系数据集成系统中的查询处理问题。首先，提出了一种基于本体的关系数据库集成框架，使用本体作为集成的全局模式来描述关系模式的语义。在此基础上，设计了基于本体的关系数据的描述方法和查询重写算法，算法可以将基于全局模式的 SPARQL 查询重写为针对具体关系数据库的查询，从而实现了对异构关系数据库的集成。

2 基于本体的关系数据集成框架

数据的格式、数据结构和数据模式的不同，给分布式异构数据的集成造成了很大困难。其主要原因在于系统的设计者在处理数据集成的过程中只考虑到信息的语法描述，而忽略了很重要的一个方面，语义描述。异构数据集成面临的主要问题是语义异构和缺乏一定的语义描述能力。我们认为，集成系统中没有语义模型是造成语义异构的根本原因。为了解决这一问题，我们将本体引入数据集成系统，利用本体可以实现异构数据源语义描述，使数据语义的描述更加规范和清晰，使用本体代替全局模式进行集成，用户建立的查询、本体和数据源模式之间的映射以及查询分解都是基于语义模型的，整个过程更清楚，含义也更加明确。

针对目前异构数据集成系统缺乏语义描述能力的现状，本文提出了一个基于本体的异构数据集成框架。该框架借鉴了 Mediator/Wrapper[16] 集成的思想，使用本体论原理、基于语义的数据源描述技术和查询重写技术，对分布、自治、异构的关系数据库进行集成。

整个系统架构如图 1 所示，包括如下 5 个层次：

(1)应用层

应用层提供统一的应用程序接口(API，Application Programming Interface)，上层应用直接通过该接口来与集成系统进行交互。应用层屏蔽关于底层数据源的具体细节，为终端用户提供访问接口，它能够显示用户可以查询的集成信息，而底层集成的数据源对用户是透明的。应用层建立在语义层上，不需要同数据源相关联。

(2)语义层

语义层可以说是基于本体的集成系统区别于传统集成系统的最显著特点。它使用领域本体作为语义模型，并支持推理。语义模型用来支持查询表达和数据源描述，为异构数据源提供了语义参照，有利于语义的集成。推理就是在已有知识的基础上发掘蕴含知识，即由显性知识得到隐性知识。语义层提供的概念定义及推理机制是语义集成的基础。

(3)中介层

中介层屏蔽了数据源的分布性和异构性，用户认为所有的数据都是本地的，处于同一服务域中，而具体的查询请求的处理、结果的返回都由中介层负责。中介层由中介器和包装器两部分组成，其中，中介器包括语法分析器和查询重写引擎两个功能组件。包装器包含了访问数据源的各种细节，主要描述了每个数据源能够产生的数据和能够接受的查询条件等，通过把数据源模式描述为基于全局模式的视图，在最大程度上保持了数据源的语义。包装器包括查询转换引擎和结果处理器两个功能组件。

(4)描述层

描述层利用语义模型对数据源的表示模型、数据模式、访问接口等各方面加以描述，将数据源的语义显示地表示出来，使之能够为机器所理解、处理。

(5)数据源层

数据源层是由分布、自治、异构的关系数据库组成，每一个数据库都可以在物理上分布存在，采用本地的方式对数据进行管理。

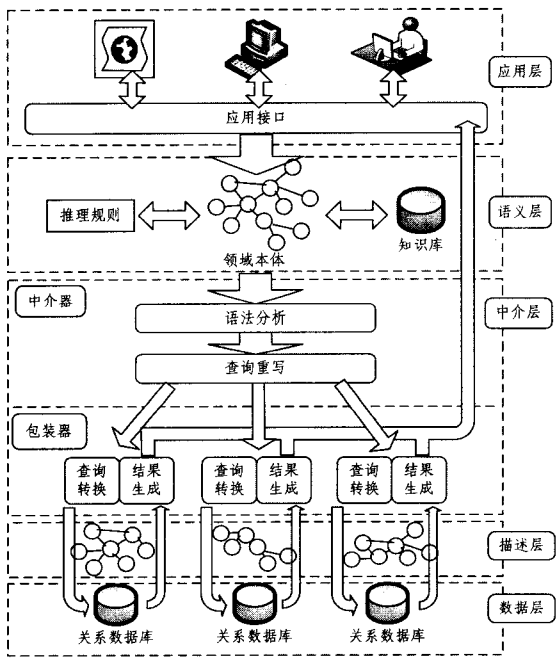

图 1 基于语义的异构数据集成框架

3 基于本体的关系模式描述

语义集成的最终目标是实现异构数据在语义层次上的知识共享与查询，因此其关键是实现各个数据源模式在语义层上的统一表示。当前异构数据集成系统的局限性源自它们使用的“模式翻译”方法，即集成系统依据预定义的规则对数据源模式进行自动或者半自动的翻译，得到对应的全局模式，然后将数据源数据转换成符合全局模式的数据实例。这种做法虽然实现了数据模式的统一，但异构的数据源模式经翻译得到的全局模式在语义层仍然是异构的，由此转换得到的数据同样也是语义异构的。要实现基于语义的异构数据源集成，必须首先将异构的数据源模式映射到统一的语义模型上。为此，本文提出基于本体的数据源描述方法，将数据源描述与语义模型结合起来，利用语义模型强大的描述能力，在语义层对数据源描述，赋予数据规范的、可共享的语义，从而协调数据库模式之间语义的异构性，实现数据源内容的机器可理解，为实现异构数据的语义集成打下基础。

为更好地描述异构数据源，提出了数据源描述本体(Data Source Description Ontology，DSDO)，用它来描述异构数据源

的元数据。数据源描述本体重点抽象出了语义描述中涉及的概念及其相关的属性。在数据源描述本体中，描述数据源的基本信息主要有3类，MetaData_Descriptor，Schema_Descriptor和Access_Descriptor，分别代表数据源的描述元数据、内容元数据和管理元数据，它们之间的相互关系和代码片段如图2和图3所示。MetaData_Descriptor提供异构数据源的基本信息，Schema_Descriptor和Access_Descriptor一起为异构数据源的内容和访问提供足够的语义信息。

(1)MetaData_Descriptor：它说明“什么样的数据源”，主要包含关于数据源的基本信息，包括数据源的规模、创建时间、拥有者、物理位置等。

```
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
      <rdf:RDF xmlns:rdf="&rdf;"
             xmlns:rdf_="&rdf_;"
             xmlns:rdfs="&rdfs;">
      <rdfs:Class rdf:about="&rdf_;Access_Descriptor"
             rdfs:label="Access_Descriptor">
            <rdfs:subClassOf
rdf:resource="&rdf_;DataSource_Descriptor"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;Concept_Mapping"
             rdfs:label="Concept_Mapping">
            <rdfs:subClassOf
rdf:resource="&rdf_;Semantic_Mapping"/>
      </rdfs:Class>
      <rdfs:Class
rdf:about="&rdf_;DataSource_Descriptor"
             rdfs:label="DataSource_Descriptor">
            <rdfs:subClassOf
rdf:resource="&rdfs;Resource"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;DataSource_Term"
             rdfs:label="DataSource_Term">
            <rdfs:subClassOf
rdf:resource="&rdfs;Resource"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;MetaData_Descriptor"
             rdfs:label="MetaData_Descriptor">
            <rdfs:subClassOf
rdf:resource="&rdf_;DataSource_Descriptor"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;Property_Mapping"
             rdfs:label="Property_Mapping">
            <rdfs:subClassOf
rdf:resource="&rdf_;Semantic_Mapping"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;Relation_Mapping"
             rdfs:label="Relation_Mapping">
            <rdfs:subClassOf
rdf:resource="&rdf_;Semantic_Mapping"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;Schema_Descriptor"
             rdfs:label="Schema_Descriptor">
            <rdfs:subClassOf
rdf:resource="&rdf_;DataSource_Descriptor"/>
      </rdfs:Class>
      <rdfs:Class rdf:about="&rdf_;Semantic_Mapping"
             rdfs:label="Semantic_Mapping">
            <rdfs:subClassOf
rdf:resource="&rdfs;Resource"/>
      </rdfs:Class>
      ……
</rdf:RDF>
```

图2　数据源描述本体代码片段

(2)Schema_Descriptor：它说明“数据源包含什么数据”，描述了内容元数据，这是数据源描述的主体部分，使用全局模式中定义的共享词汇集对数据模式进行描述，从而将数据源的语义显示地表示出来。Schema_Descriptor由两部分组成：View_Definitin和Mapping_Definition，其中，View_Definitin将数据源看成是定义在全局模式上的视图，使用SPARQL查询来定义数据源模式，一方面，SPARQL查询的返回变量指明了数据源的查询能力；另一方面，因为SPARQL是基于图模式的，查询的图模式很好地描述了数据源的内容，这两方面的信息都为后续的查询重写提供支持。Mapping_Definition定义了关系模式中的元素(表、列、主键、外键等)与全局模式中的元素(概念、关系、属性等)之间的对应关系，这些对应关系指明了语义模型与关系模型之间的具体映射关系，为后面的查询转换提供支持。

(3)Access_Descriptor：它说明“如何访问数据”，指明数据源提供的访问接口类型和查询能力。

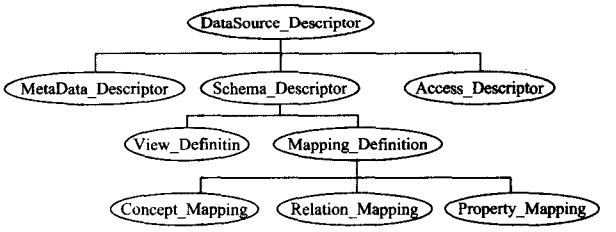

图3　数据源描述本体结构示意图

4　查询重写算法

传统的查询重写算法，如Bucket算法、Inverse-rules算法和MiniCon算法[17]等都基于关系模型，由于关系模型和本体在数学模型上的不同，现有的查询重写算法无法直接应用到基于本体的数据集成中。本文对查询重写问题进行了深入研究，提出了RDF图模型上的查询重写算法，解决了基于本体的数据集成系统中的查询重写问题，算法参考了MiniCon算法的思想。

4.1　基本定义

定义1(RDF三元组，RDF图)　令I,B,L为两两不相交的有限集，分别表示IRIs、空节点、文字和文字的集合。RDF三元组t是一个三元组：

$$tuple(s,p,o)\in(I\cup B)\times I\times(I\cup B\cup L)$$

式中，s，p和o分别表示三元组的主体、谓词和客体。

RDF图G是三元组的集合。

定义2(三元组模式，图模式)　令V表示变量的有限集，三元组模式tp是一个三元组：

$$tp\in(I\cup V)\times(I\cup V)\times(I\cup L\cup V)$$

图模式gp是三元组模式的集合。

定义3(SPARQL查询)　SPARQL查询Q=SELECT $vars$ WHERE gp。其中，$vars=\{x_1,x_2,\cdots,x_p\}$，列出了$Q$的所有返回变量。

在传统的数据库领域中，视图是命名的查询(Named Query)，根据上一节提出的数据源描述方法，数据源可以看作是RDF图模型上的视图，下面给出数据源视图的定义。

定义4(数据源视图)　数据源视图V=SELECT $vars$ WHERE gpv。其中，$vars=\{y_1,y_2,\cdots,y_q\}$，列出了该数据源能够返回的所有变量，反映了数据源的查询能力；gpv为数据源覆盖的图模式，反映了数据源数据的语义。

对于定义在全局模式上的数据源视图集$V=\{V_1,V_2,\cdots,V_n\}$和给定查询Q，查询重写的任务就是从V中选择可能的子集，使之能够满足查询。这些子集就是查询重写的结果。

定义5(最小可连接单元，MCU)　查询Q相对于数据源视图V的一个最小可连接单元是一个形如$\langle\varphi,V,gpm\rangle$的二元组，其中$\varphi$是一个从查询变量到数据源视图变量的映射，将$Q$中的一部分变量映射到$V$中；$gpm$是一个基本图模式，表示$Q$和$V$的图模式相交部分中的一个不可分割的可连接单元。

4.2　算法描述

查询重写算法主要可分为三步：第一步分析数据源视图与查询的具体关系，从中提取最小可连接单元(Minimum Connectable Unit，MCU)；第二步针对上一步得到的MCU组合连接，找出所有可能的查询重写。具体的MCU构造算法和MCU组合算法如下所示。

算法 1 MCU 构造算法

输入：查询图模式 gpq 和数据源集 $V=\{v_1,v_2,\cdots,v_n\}$

输出：MCU 集 $M=\{m_1,m_2,\cdots,m_r\}$

(1) 初始化返回结果集 $M=\emptyset$；

(2) 对于 V 每个相关数据源 v，令 gpv 表示数据源图模式，gpc 为查询图模式中和数据源图模式闭包相交的部分，映射 φ 表示从 gpq 中的查询变量到 gpv 中变量的映射；

(3) 若 gpc 不为空，构造一个新的图模式 $gpm=\emptyset$，从 gpc 中取一条三元组放入 gpm 中。若 gpc 为空则直接返回；

(4) 检查 gpm 中的每个变量，若变量在查询图中是一个中间变量，则扩展 gpm，使其包含所有与该中间变量相邻的节点。重复步骤(4)，直到 gpm 的大小不再变化；

(5) 将三元组$\langle\varphi,v,gpm\rangle$生成的 MCU 加入到结果集；

(6) 重复步骤(3)；

(7) 返回最终求得的 MCU 集 M。

算法 2 MCU 组合算法

输入：查询图模式 gpq 和 MCU 集 M

输出：查询重写集 R

(1) 初始化返回结果集 $R=\emptyset$；

(2) 对 M 的每个子集$\{m_1,m_2,\cdots,m_k\}$，若满足下面条件：

(a) $gpm_1\cup gpm_2\cup\cdots\cup gpm_k=gpq$

(b) $\forall i\neq j, gpm_i\cup gpm_j=\emptyset$

则对于每个 $m_i=\langle\varphi_i,v_i,gpm_i\rangle$，定义如下的变量映射 ψ_i：若存 gpq 中存在变量 x 使得 $\varphi_i(x)=y$，则令 $\psi_i(y)=x$，否则 $\psi_i(y)=y$

(3) 构造查询重写 $Q'=\psi_1(v_1),\psi_2(v_2),\cdots,\psi_k(v_k)$；

(4) 将 Q'加入结果集 R 中；

(5) 重复步骤(2)；

(6) 返回结果集 R。

综合算法 1 和算法 2，我们得到基于本体的查询处理算法如下。

算法 3 基于语义的查询重写算法

输入：查询 Q 和相关数据源集 $V=\{v_1,v_2,\cdots,v_n\}$

输出：查询重写集 R

(1) 初始化返回结果集 $R=\emptyset$；

(2) 初始化 MCU 集 $M=\emptyset$；

(3) 对于每个 V 中的视图 v，如果 v 与查询 Q 相关，则调用算法 1 构造 MCU，将结果加入 M；

(4) 根据上一步得到的 MCU 集 M，调用算法 2，将结果集 R 返回。

4.3 算法分析

给定一个数据源集合 $V=\{v_1,v_2,\cdots,v_n\}$和一个查询 $Q=$ SELECT $vars$ WHERE gp，令$|gp|=m$，算法 1 和算法 3 都可以在多项式时间内完成，而算法 2 需要从 MCU 集中寻找满足条件的子集，这需要 $O(n^m)$的时间复杂度。但由于算法 2 在构造 MCU 时已经将不相关的数据源视图排除，大大裁剪了不必要的搜索分支，而且一般查询的长度 m 都不会很大，不会消耗太多时间为这些查询构造查询重写，因此可以预计算法的实际运行效率会比较理想。

5 实验与分析

本节提出的算法已全部在 Java 1.6 版本上实现，实验在一台 Pentium Dual 1.79GHz，1G 计算机上进行。为了实验数据尽可能符合实际应用，我们收集了 100 个异构的关系表并应用第 3 节中提出的基于语义的数据源描述方法对它们做了描述，然后构造了长度从 1 到 10 的查询各 10 个。理论上，查询重写算法的时间复杂度为 $O(n^m)$，其中 n 为数据源数量，m 为查询的长度。为了分析查询长度和数据源数量对查询重写算法执行效率产生的影响，设计了如下实验，实验结果中的数据都是在相同条件下重复 10 次实验结果的平均值。

从实验数据看出，在保持数据源数量不变的情况下，查询长度的增加导致查询重写算法执行时间的增长，图 4 是查询重写算法执行时间随查询长度变化的情况，3 条曲线分别表示 3 种不同的数据源数量；而在保持查询长度不变的情况下，数据源的增加也会导致查询重写算法执行时间的增长，图 5 是查询重写算法执行时间随数据源数量变化的情况，3 条曲线分别表示 3 种不同的查询长度。

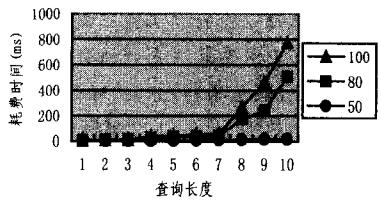

图 4 查询重写算法执行时间随查询长度变化曲线

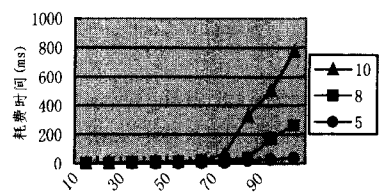

图 5 查询重写算法执行时间随数据源数量变化曲线

上述实验结果与第 4 节的复杂性分析是一致的。根据这些实验数据得出结论：在查询长度不超过 10，而数据源较多时，基于图模式的查询重写能够提供较好的执行时间。这也符合数据集成的应用需求，由于 SPARQL 具有较好的可读性，并能准确、直观地表达查询的语义，其查询长度通常不会超过 10。

结束语 本文分析了目前研究中基于关系模型的数据集成的不足，设计了基于语义的异构数据集成框架，提出使用数据描述本体对数据源描述，设计了一个用来表示数据语义的数据源描述本体，实现了对异构数据的语义的有效描述。提出了基于图模式的查询重写，并通过实验对该算法的性能进行了验证。

参 考 文 献

[1] Halevy A, Rajaraman A, Ordille J. Data integration: the teenage years[C]// Proceedings of the 32nd International Conference on Very Large Data Bases. 2006: 9-16

[2] 孟小峰，周龙骧，王珊. 数据库技术发展趋势[J]. 软件学报，2004，15(12)：1822-1836

[3] Gruber T R. A translation approach to portable ontology specifications[J]. Knowledge Acquisition, 1993, 5(2): 199-220

[4] Wache H, Vögele T, Visser U, et al. 2001, Ontology-Based Integration of Information-A Survey of Existing Approaches[C]// Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing. Seattle, USA, 2001: 108-117

[5] Zhu H, Madnick S. Scalable Interoperability Through the Use of COIN Lightweight Ontology[C]// Proceedings of the 2nd International VLDB Workshop on Ontologies-Based Databases and Information Systems. Seoul, Korea, 2006: 37-50

(下转第 160 页)

guity resolution[D]. Pennsylvania:Pennsylvania,1998
[2] Nigam K,Laferty J,McCallum A. Using Maximum Entropy for Text Categorization[C]// Workshop on Machine Learning for Information Filtering. 1999:61-67
[3] 李荣陆,王建会,陈晓云,等. 使用最大熵模型进行中文文本分类[J]. 计算机研究与发展,2005,42(1):94-102
[4] 李素建. 汉语组块计算的若干研究[D]. 北京:中国科学院计算技术研究所,2002
[5] Berger A L,Pietra S A D,Pietra V J D. A Maximum Entropy Approach to Natural Language Processing[J]. Association for Computational Linguistics,1996,22(1):39-71
[6] Jelinek F, Mercer R L. Interpolated Estimation of Markov Source Parameters from Sparse Data[C]// Proceedings of the Workshop on Pattern Recognition. Practice,1980:381-398
[7] 于浚涛. 基于最大熵的汉语介词短语自动识别[D]. 大连:大连理工大学,2006
[8] Zhai C,Lafferty J. A Study of Smoothing Methods for Language Models Applied to Information Retrieval[J]. ACM Transactions on Information Systems,2004,22(2):179-214
[9] Gao J,Goodman J,Li M,et al. Toward a Unified Approach to Statistical Language Modeling for Chinese[J]. ACM Transactions on Asian Language Information Processing,2002,1(1):3-33
[10] 黄永文,何中市. 基于全局折扣的统计语言模型平滑技术[J]. 重庆大学学报:自然科学版,2005,28(8):51-55
[11] 黄建中,王肖雷. Katz 平滑算法在中文分词系统中的应用[J]. 计算机工程,2004,增刊(1):370-372
[12] 许家金. 兰开斯特汉语语料库介绍[EB/OL]. http://nlp.org/,2006
[13] Yang Xiaojun. Survey and Prospect of China's Corpus-based Researches[C]// The Corpus Linguistics Conference. Lancaster University(UK). 2003

(上接第 137 页)

[6] Dou D, LePendu P. Ontology-based Integration for Relational Databases[C]// Proceedings of the 2006 ACM Symposium on Applied Computing. Dijon,France,2006:461-466
[7] Wang Jin-peng, Lu Jian-jiang, Zhang Ya-fei, et al. Integrating heterogeneous data source using ontology[J]. Journal of Software,2009,4(8):843-850
[8] 吕艳辉,马宗民,王玉喜. 基于关系数据库的 OWL 本体构建方法研究[J]. 计算机科学,2009,36(7):153-156
[9] Laborda C P, Conrad S. Bringing Relational Data into the Semantic Web using SPARQL and Relational. OWL[C]// Proceedings of the 22nd International Conference on Data Engineering Workshops. Atlanta,USA,2006:55
[10] Erling O,Mikhailov I. Integrating Open Sources and Relational Data with SPARQL[C]// The 5th European Semantic Web Symposium and Conference. Tenerife,Spain,2008:838-842
[11] Weiske C, Auer S. Implementing SPARQL Support for Relational Databases and Possible Enhancements[C]// Proceedings of the 1st Conference on Social Semantic Web. Leipzig,Germany,2007:69-80
[12] Halevy A. Answering queries using views:A survey [J]. Very Large Data Bases Journal,2001,10(4),270-294
[13] Abiteboul S,Duschka O M. Complexity of answering queries using materialized views[C]// Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. Seattle,US,1998:254-263
[14] Beeri C,Levy A,Rousset M C. Rewriting queries using views in description logics[C]// Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS). Tuscon,Arizona,1997:99-108
[15] Baader F,Hollander B. Kris:Knowledge representation and inference system[J]. SIGART Bulletin,1991,2(3):8-14
[16] Levy A,Rajaraman A,Ordille J. Querying heterogeneous information sources using source descriptions[C]// Proceedings of the 22th International Conference on Very Large Data Bases. Bombay,India,1996:251-262
[17] Pottinger R,Halevy A. MiniCon:A Scalable Algorithm for Answering Queries Using Views[J]. The VLDB Journal,2001,10(2/3):182-198

(上接第 133 页)

图 2 显示的则是两个关系数据流分布不协调一致的情况,BSJ 算法能显示出其优越性。这是由于两个关系流分布不协调一致,也就是说在 R_1(或者 R_2)中可能存在一些关键字的大量取值,在 R_2 中与此相等的关键字含量过低,因而通过算法能够准确地找到关系流中出现概率较低的元组及其相应分区。

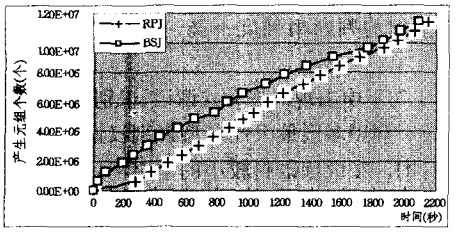

图 2 数据分布不协调时的性能

结束语 提出了一个新的内存刷新策略,再对数据流上的数据频率进行近似的统计分析,将分析结果应用于关系连接的输出流上,很好地反映了输入流中的数据分布情况,提高了刷新策略的准确性。

参 考 文 献

[1] Lawrence R. Early Hash Join:A configurable algorithm for the efficient and early production of join results[C]// VLDB. 2005:841-852
[2] Wilschut A N,Apers E M G. Pipelining in query execution[C]// Proc. of the International Conference on Databases,Parallel Architectures and their Applications. Miami,USA,March 1990
[3] Urhan T,Franklin M J. XJoin:Getting Fast Answers from Slow and Burst Networks[R]. CS-TR-3994. Computer Science Department,University of Maryland,1999
[4] Mokbel M F,Lu Ming,Aref W G. Hash-Merge Join:A Non-blocking Join Algorithm for Producing Fast and Early Join Results[C]// Proceeding of the 20th International Conference on Data Engineering. Washington:IEEE Computer Society,2004:251-263
[5] Tao Yufei,Yiu M L,Papadias D,et al. RPJ:producing fast join results on streams through rate-based optimization[C]// Proceedings of the 24th ACM SIGMOD International Conference on Management of Data. New York:ACM press,2005:371-382