集成对象代理数据库的科学工作流服务框架中的数据跟踪

王黎维[1)] 黄泽谦[2)] 罗 敏[2)] 彭智勇[2),3)]

[1)](武汉大学国际软件学院 武汉 430072)

[2)](武汉大学软件工程国家重点实验室 武汉 430072)

[3)](武汉大学计算机学院 武汉 430072)

摘 要 文中提出了一个集成数据库的科学工作流服务框架,它采用对象代理模型描述一系列科学任务的执行过程,使得工作流管理操作以类似于传统数据库管理操作的方式来完成.同时,基于对象代理数据库中的双向指针机制,文中提出了一种新的数据跟踪方法,该方法能提供比注释或反向方法更高的性能,不仅节省了大量的存储空间,而且减少了额外的计算代价,为了进一步提高数据跟踪的高效性,文中也提出了一种部分物化中间数据模式,实验显示它具有较好的系统性能.

关键词 科学工作流;Web 服务;对象代理模型;数据跟踪

中图法分类号 TP311

Data Provenance in a Scientific Workflow Service Framework Integrated with Object Deputy Database

WANG Li-Wei[1)] HUANG Ze-Qian[2)] LUO Min[2)] PENG Zhi-Yong[2),3)]

[1)](*International School of Software*, *Wuhan University*, *Wuhan* 430072)

[2)](*State Key Laboratory of Software Engineering*, *Wuhan University*, *Wuhan* 430072)

[3)](*Computer School*, *Wuhan University*, *Wuhan* 430072)

Abstract This paper proposes a DB-integrated scientific workflow service framework which adopts object deputy model to describe the execution of a series of scientific tasks, thus allows workflow management operations to be performed in a way analogous to traditional database management operations. Furthermore, based on bi-directional pointer mechanism in object deputy database, this paper introduces a new data provenance method. This approach is much more efficient than annotation or inversion, which not only saves a lot of storage space, but also reduces the cost of extra computing. A partial materializing data schema is also presented to improve the efficiency of data tracking, and the experiment results show that it can provide a preferable system performance.

Keywords scientific workflow; Web service; object deputy model; data provenance

---

收稿日期:2006-05-31;最终修改稿收到日期:2008-01-07.本课题得到国家自然科学基金(60573095)、教育部新世纪优秀人才计划项目基金(NCET-04-0675)、国家"八六三"高技术研究发展计划项目基金(2006AA12Z210)、教育部博士点基金项目(20050486024)、教育部科学技术研究重点项目基金(107072)、国家"九七三"重点基础研究发展规划项目基金(2007CB310806)资助.**王黎维**,女,1980 年生,博士,讲师,主要研究方向为工作流管理和事务处理.E-mail: wangliwei0815@hotmail.com.**黄泽谦**,男,1983 年生,博士研究生,主要研究方向为工作流管理和事务处理.**罗 敏**,男,1981 年生,博士研究生,主要研究方向为工作流管理.**彭智勇**,男,1963 年生,教授,博士生导师,主要研究领域为 Web 数据管理、复杂数据管理和可信数据管理.

1 引 言

目前,生物、医学、生态等科学领域已经面临信息爆炸,使得这些领域的科学家每天不得不使用各种专业程序处理来自实验室信息管理系统控制的科学实验、大规模计算为中心的模拟或远程传感器实时观测的科学数据,这对自动管理科学实验提出了迫切的需求.为了支持科学家进行高效的数据管理和实验分析,将工作流技术应用于科学领域的科学工作流(Scientific Workflow,SWF)如今已成为在网络基础设施上进行大规模科学计算和协同研究的有效方法.目前在合作环境中进行跨组织的科学实验已经越来越重要.由于该环境中数据和程序资源的分布和异构,使得广泛用于工业界的 Web 服务技术引入到科学工作流中,用于支持来自不同平台的计算资源的整合,如文献[1-2]等.它将程序封装到服务中,并通过服务调用而不是程序调用组成工作流,因此能够跨越技术和组织的边界对流程进行建模.

与以任务为中心的商业工作流相比,科学工作流虽然共享了商业工作流的一些特性,但其主要是以数据为中心[3],常常涉及大规模科学计算,具有描述科学数据的元数据或者注释.因此,科学工作流实际上是数据管理和流程管理的结合[4].探索数据库技术用于科学工作流管理已成为当前的研究热点.通常,工作流管理系统(Workflow Management System,WFMS)使用数据库管理系统(Database Management System,DBMS)来存储任务描述以及工作流数据,并在 DBMS 之上实现各个工作流功能模块.然而目前一些研究证明[5-6],大多数 WFMS 的功能可由 DBMS 提供,许多成熟的数据库技术,如计划、查询、数据跟踪以及并发控制等均可用于工作流管理系统中.采取 DBMS 的方式实现科学工作流管理,可以减少实现开销,增加优化的可能性,使其以一种类似于传统数据库操作的方式进行工作流管理.因此,采用以数据库系统为核心的科学工作流管理体系架构,变得非常有意义.

科学工作流最重要的功能是作为识别实验数据的一种方式,称其为数据跟踪(data provenance)[7-8],它用于提供数据的派生历史,解释数据的由来,这对于实验分析、错误诊断以及核实实验结果的有效性很有帮助.另外,当产生实验数据的实验步骤发生改变时,能通过数据的派生历史跟踪实验进程,确定需要重做哪些步骤以便更新实验结果.因此,数据跟踪主要包含两个重要信息:用于产生实验数据的数据源以及所经历的实验步骤.

目前,数据跟踪的重要性已经被多个科学工作流项目所认可,如 GridDB[6],Chimera[9],myGRID[10],CMCS[11].注释(Annotation)和反向(Inversion)是目前两种主要的数据跟踪方法.注释将一个数据的派生历史搜集起来作为元数据,与数据一起存放在数据库中,用于解释数据的来源.反向方法主要应用逆向查询或者逆向函数,由结果数据溯源到其源数据.但两者都有不足之处,注释不能为细粒度的实验数据提供良好的存储规模,因为记录完整的派生历史可能超过数据本身所需要的存储空间;反向方法从存储的角度来看似乎更好,因为逆向函数或逆向查询能标识某一类数据的派生信息,但每次需要查询数据源时,它需要产生一个逆向函数或逆向查询去计算派生信息,在数据量大的情况下,执行效率较低.

本文,我们提出一个集成对象代理数据库的科学工作流服务框架,并在此框架下着重研究基于对象代理模型的数据跟踪方法.该方法能借助实验数据间的双向指针实现数据跟踪,能提供比注释或反向方法更高的性能,不仅节省了大量的存储空间,而且减少了额外的计算代价.

本文第 2 节介绍一个集成对象代理数据库的科学工作流服务框架;第 3 节提出基于对象代理模型的数据跟踪方法;第 4 节给出实验结果和分析;第 5 节分析并比较一些相关工作;最后总结全文并提出下一步需要开展的工作.

2 集成对象代理数据库的科学工作流服务框架

集成对象代理[12]数据库 TOTEM 的科学工作流服务框架如图 1 所示,它将数据管理和工作流管理有效地结合,以对象代理数据库来建模科学工作流,并采用类似 SQL 语言的方式来执行,管理和监控科学工作流.

该框架使用对象代理数据库 TOTEM 作为科学工作流的运行环境,提供科学工作流所必需的工作流设计和执行功能,从而取代商业工作流引擎的核心位置.工作流设计主要用于定义工作流模式,模式定义通过对象代理数据库中的类和代理类定义工作流中不同科学任务的输入和输出,通过任务输入输出间的对象代理关系实现工作流的自动执行.工

作流执行是系统的核心模块，它主要采取数据驱动的方式，使得科学任务调用相应的科学程序对提供的输入数据进行计算，将执行转化为对数据库的插入和更新操作从而实现工作流的自动执行.

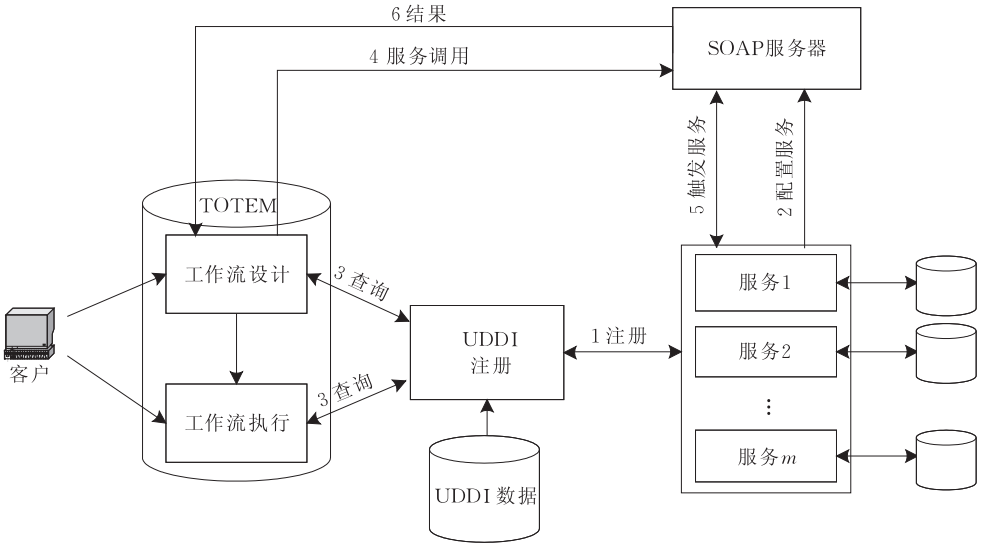


图 1　集成对象代理数据库的科学工作流服务框架

其中，科学程序通过使用 Web 服务描述语言(如 WSDL)封装，我们称其为科学服务程序. 它仅仅提供程序的接口而不必关心程序的具体实现，隐藏了它们的异构性，使它们能被任何类型的客户端应用程序存取. 从图 1 中我们可以看出，服务提供者使用公共的 UDDI 注册器注册服务描述，客户端查询 UDDI 得到服务描述细节，找到所感兴趣的服务，并构造和执行工作流. 一旦工作流中的某一任务被触发，它通过 UDDI 接受服务的输入参数，并向 SOAP 服务器发送合适的 SOAP 消息，接受 SOAP 消息的 SOAP 服务器触发服务的实际实现，结果被返回到 SOAP 服务器，再返回给客户端. 一个任务的输出可能被直接传递或修改，并作为下一个任务的输入.

2.1　工作流建模

科学工作流由科学数据和操作它们的科学任务组成. 科学任务具有输入数据与输出数据，我们建模输入、输出数据作为对象代理数据库 TOTEM 中的类和代理类，其中，科学任务的输入数据是其直接前驱科学任务的输出数据. 工作流流程初始科学任务的输入数据用类来定义，构成工作流流程的所有科学任务的输出数据定义为其输入数据的代理类，代理类的类型根据对应工作流模式的不同而有所不同. 被科学任务调用的科学服务程序定义为输入数据对象属性上的读方法.

2.1.1　数据声明

定义 1.　工作流流程初始科学任务的输入数据被定义成类 $C=\langle O,A\rangle$.

(1) O 是 C 的数据对象集. $o\in O$ 是 C 的一个数据对象.

(2) A 是 C 的属性集合. $(T_a:a)\in A$ 分别代表一个属性的类型和属性名. 数据对象 o 的属性 a 的值表示为 $o.a$. 对每个属性 $T_a:a$，都有两个基本方法：

$$\mathrm{read}(o,a)\Rightarrow\uparrow o.a,\ \mathrm{write}(o,a,v)\Rightarrow o.a:=v.$$

这里，$\Rightarrow$，$\uparrow$ 分别代表激活、结果返回.

定义 2.　工作流流程除初始科学任务外的其它科学任务的输出数据基于其输入数据来定义，前者称为后者的代理类，后者称为前者的源类. 源类中的数据对象与其代理类中的数据对象通过双向指针链接. 代理数据对象拥有自己的标识符，并通过切换操作完全或部分共享源数据对象的属性值，同时还可以增加属性. 令 $C^s=\langle O^s,A^s\rangle$ 是源类，其代理类 C^d 可以定义为 $C^d=\langle O^d,A^d\cup A^d_+\rangle$.

(1) 代理数据对象 $O^d=\{o^d_i\mid o^d_i\rightarrow o^s_i\mid\cdots\times o^s_i\times\cdots\mid\{o^s_i\},\ sp(o^s_i)\mid jp(\cdots\times o^s_i\times\cdots)\mid gp(\{o^s_i\})==\mathrm{true}\}$，其中 $o^d_i\rightarrow o^s_i\mid\cdots\times o^s_i\times\cdots\mid\{o^s_i\}$ 表示 o^d_i 是 o^s_i，$\cdots\times o^s_i\times\cdots$，或 $\{o^s_i\}$ 的代理数据对象，sp，jp 和 gp 分别表示选择、组合和分组谓词.

根据对象代理模型[12]，代理数据对象具有 4 种类型，分别是 Select，Join，Group 和 Union，用于支持特化、聚集、分组和组合.

(2) $A^d\cup A^d_+$ 是 C^d 的属性集.

(i) $(Ta^d:a^d)\in A^d$ 是从 $(Ta^s:a^s)\in A^s$ 继承而来

的属性.对代理数据对象 o^d 的属性值 a^d 的计算通过切换操作最终作用到源数据对象的属性 a^s 上.由于更新继承的属性值不符合科学工作流的要求,所以属性 a^d 的写方法不存在,a^d 读方法的切换操作定义如下:

$$read(o^d,a^d)\Rightarrow\uparrow fTa^s\rightarrow Ta^d(read(o^s,a^s)).$$

(ii)$(Ta^d_+:a^d_+)\in A^d_+$ 是增加的属性,其基本方法定义如下:

$$read(o^d,a^d_+)\Rightarrow\uparrow o^d.a^d_+,$$

$$write(o^d,a^d_+,v^d_+)\Rightarrow o^d.a^d_+:=v^d_+.$$

根据该定义,每次查询代理数据对象继承的属性值仍然需要通过访问源数据对象的切换操作来计算.由于科学计算的结果数据可能由运行时间较长的科学程序产生,所以最好物化继承的属性值以避免每次查询时重新计算它们.物化的继承属性的读方法定义被修改如下:

$$read(o^d,a^d)\Rightarrow\uparrow o^d.a^d.$$

即继承的属性值能直接从代理数据对象中读取.

源数据对象上的更新将触发更新迁移过程[12]对代理数据对象进行更新,也就是,如果增加、删除或修改源数据对象,相应的代理数据对象也会被增加、删除或修改.由于源数据对象与其代理数据对象之间存在各种依赖关系和语义约束,其一致性可通过更新迁移进行维护.

2.1.2 流程声明

工作流流程的执行可能需要连接多个科学任务,任何复杂的工作流流程能被简化成在给定数据上按某种顺序执行的科学任务序列.本框架的特点是将工作流流程看成一系列数据的相互连接,将工作流任务间的相互关系转换为任务关联数据间的关系,同时采用对象代理数据库根据不同的工作流模式来建立数据间的关系,利用更新迁移机制触发工作流的自动执行,因此能够支持文献[13]提出的几种常见的工作流模式,包括:顺序模式、并行分支模式、同步模式、单选模式、简单合并模式、多路合并模式、多路选择模式等.

如图 2(a)所示,任务 A 的输出 $AOutput$ 定义为任务 B 的输入,任务 B 的输出定义为类 $AOutput$ 的 Select 型代理类 $BOutput$.任务 B 对应的科学服务程序作为 $AOutput$ 中数据对象属性上的读方法,一旦 A 完成对输入数据的计算,往 $AOutput$ 中插入结果数据对象时,会触发 B 的执行,并在 $BOutput$ 中派生对应代理数据对象.

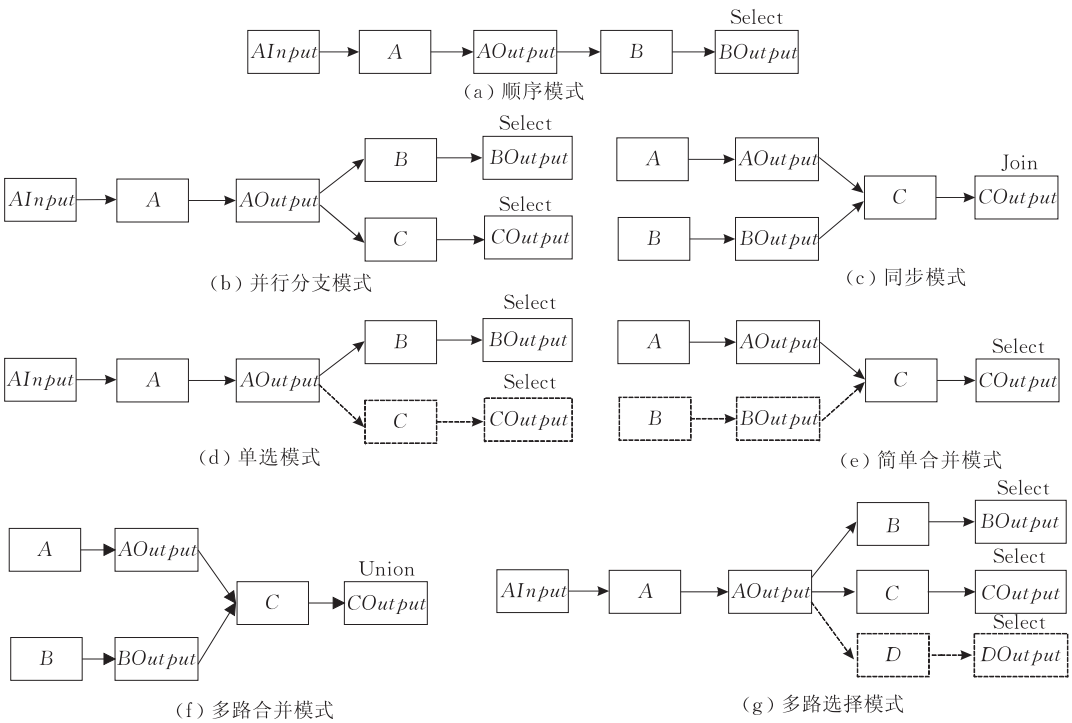


图 2 对象代理模型建模工作流

如图 2(b)所示,任务 B 和 C 的输出 $BOutput$ 和 $COutput$ 分别定义为任务 A 的输出类 $AOutput$ 的 Select 型代理类.对于并行分支模式,一旦 A 完成对数据的计算,往 $AOutput$ 中插入一个或多个结果数据对象时,会同时触发 B 和 C 的执行,并在 $BOutput$ 和 $COutput$ 中派生对应的代理数据对象.

如图 2(c)所示，任务 A，B 的输出分别定义为类 $AOutput$，$BOutput$；任务 C 的输出定义为 $AOutput$ 和 $BOutput$ 的 Join 型代理类 $COutput$. 对于同步模式，只有当 A 和 B 都完成时，在 $AOutput$ 和 $BOutput$ 中分别产生数据对象，才会触发 C 的执行，并在 $COutput$ 中派生一个代理数据对象. 同步模式中的任务同步要求转换为 Join 型代理类的更新迁移.

如图 2(d)所示，任务 B 和 C 的输出分别定义为任务 A 的输出 $AOutput$ 的 Select 型代理类 $BOutput$ 和 $COutput$. 对于单选模式，只有一个分支会被执行. 当 A 往 $AOutput$ 插入数据对象时，数据库会根据 $BOutput$ 和 $COutput$ 的代理规则，触发相应分支的执行，并派生相应的代理数据对象到 $BOutput$ 或 $COutput$ 中.

如图 2(e)所示，任务 A 与 B 的输出分别定义为类 $AOutput$ 和 $BOutput$；任务 C 的输出 $COutput$ 定义为 $AOutput$ 或 $BOutput$ 的 Select 型代理类. 对于简单合并模式，只有一个分支 A 或 B 被触发执行，一旦任务执行完成，往 $AOutput$ 或 $BOutput$ 插入数据，触发 C 的执行，并会在 $COutput$ 中派生相应的代理数据对象.

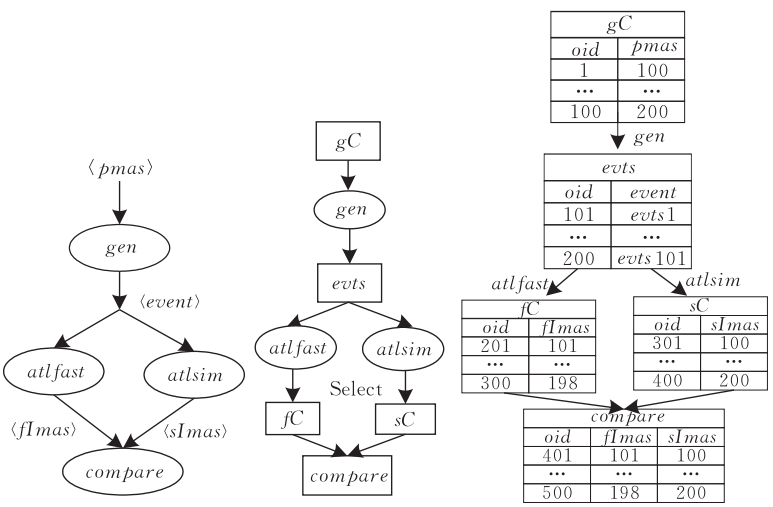
如图 2(f)所示，任务 A 与 B 的输出分别定义为类 $AOutput$ 和 $BOutput$；任务 C 的输出 $COutput$ 定义为 $AOutput$ 和 $BOutput$ 的 Union 型代理类. 对于多路合并模式，A 和 B 中任何一个分支执行完，分别往 $AOutput$ 或 $BOutput$ 插入数据对象时，都会触发 C 的执行，并在 $COutput$ 中派生相应的代理数据对象.

如图 2(g)所示，任务 B，C 和 D 的输出分别定义为任务 A 的输出 $AOutput$ 的 Select 型代理类 $BOutput$，$COutput$ 和 $DOutput$. 对于多选模式，至少 B，C 和 D 中有一个分支会被执行，当 A 往 $AOutput$ 插入数据对象时，数据库会根据 $BOutput$，$COutput$ 和 $DOutput$ 的代理规则，触发任务执行，并派生相应的代理数据对象到 $BOutput$，$COutput$ 和 $DOutput$ 中，从而触发相应分支的执行.

采用对象代理数据库建模工作流，使得与工作流模式相关的控制逻辑由数据库来实现. 由于除初始任务外，所有任务的输出都定义为该任务所有直接前驱任务的输出类的代理类，所以一旦前驱任务往其输出数据类中插入数据，更新迁移将根据各个输出类的代理规则，调用后继任务进行计算，派生对应的代理对象到后继任务的输出类中.

2.2　Atlas 高能物理工作流的建模与执行

我们引用 Atlas 高能物理工作流[5]解释构造和执行工作流的过程. 如图 3(a)所示，Atlas 工作流包括 3 个科学任务，分别执行事件发生器程序(gen)、快速模拟探测器程序($atlfast$)和慢速模拟探测器程序($atlsim$). gen 接收一种粒子的质量，产生一个 event 文件，该文件描述了粒子的衰变. 之后 event 文件被分别输入到程序 $atlfast$ 和 $atlsim$，模拟两种探测器对粒子衰变事件的反应，并且产生包含数值 $fImas$ 和 $sImas$ 的文件，两个结果数值实际上是粒子衰变而产生的子粒子的质量之和. 最后，比较这两个结果.



(a) Atlas工作流　　(b) Atlas工作流模型　　(c) Atlas数据库模式

图 3　Atlas 工作流建模

图 3(b)和图 3(c)分别描述了 Atlas 工作流模型及对象代理数据库中对应的数据库模式. 对象代理数据库语言[14]建立 Atlas 工作流如下，其中 gen，$atlfast$ 和 $atlsim$ 被封装成 3 个科学服务程序，分别定义为 $event$，$fImas$ 和 $sImas$ 的读方法.

第 1 个科学任务的输入和输出类定义如下：

```
CREATE CLASS gC (pmas int);
CREATE SELECTDEPUTYCLASS evts AS (SELECT
    gen(pmas) AS (event int) FROM gC);
```

类 gC 的 $pmas$ 属性表示输入的粒子质量，由类 gC 派生的代理类 $evts$ 存储由科学服务程序 gen 产生的 event 文件，其中 $event$ 属性表示文件标识符.

执行科学服务程序 $atlfast$ 和 $atlsim$ 的科学任务的输出类分别定义为类 $evts$ 的 Select 型代理类 fC 及 sC：

```
CREATE SELECTDEPUTYCLASS fC AS
  (SELECT atlfast(event) AS (fImas int) FROM evts);
CREATE SELECTDEPUTYCLASS sC AS
  (SELECT atlsim(event) AS (sImas int) FROM evts);
```

fC 及 sC 分别存放科学服务程序 $altfast$ 和 $atlsim$ 产生的结果数据 $fImas$ 及 $sImas$，它们表示粒子衰变得到的子粒子的质量总和.

类 fC 和 sC 的 Join 型代理类 $compare$ 比较 $atlfast$ 和 $atlsim$ 所产生的结果 $fImas$ 和 $sImas$.

```
CREATE JOINDEPUTYCLASS compare AS
(SELECT fC.fImas,sC.sImas FROM fC, sC WHERE
    fC→evts.event=sC→evts.event);
```

因此，Atlas 工作流的建立通过不断地派生代理类完成. 向类 gC 中插入一个数据对象将启动 Atlas 工作流的自动执行，如：

```
INSERT INTO gC VALUES (100);
```

一旦一个 $pmas$ 值为 100 的数据对象被插入到 gC，根据 $evts$ 的代理规则，自动触发第一个科学任务，也即科学服务程序 gen 执行，并在 $evts$ 类中产生它的代理数据对象 $evts1$，同样，$evts1$ 触发科学服务程序 $atlfast$ 和 $atlsim$ 的自动执行，并将执行完后的结果分别添加到类 fC 和 sC 中，最终导致产生对象 $compare1$. 另外，如果 $gC1$ 的 $pmas$ 值被修改，所有数据对象包括 $evt1$，$fC1$，$sC1$ 以及 $compare1$ 将自动的重新计算.

集成对象代理数据库 TOTEM 的科学工作流服务框架意味着与数据库的紧密结合. 工作流的声明和定义使用了类似 SQL 的对象代理数据库语言，触发、查询和执行使用 SQL 来完成，使得工作流的管理操作以类似于传统数据管理操作的方式来完成.

3 基于对象代理模型的数据跟踪

本节在该框架的基础上提出了一种基于对象代理模型的数据跟踪方法. 区别于一般的注释和反向等数据跟踪方法，利用对象代理数据库内部对象间的双向指针可以实现高效的数据跟踪.

3.1 数据跟踪

该框架使用 TOTEM 数据库的类、代理类为科学任务的输入数据和输出数据建模，其中，每一个科学任务实现由输入数据到输出数据的映射、输入数据和输出数据通过输入数据类和输出数据类中的数据对象或代理数据对象来表达. 每一个数据对象或代理数据对象在数据库中具有唯一的对象标识 OID. 所以该框架可以基于对象之间的 OID 双向指针来实现数据跟踪.

对象 OID 间的双向指针通过 TOTEM 自身维系的双向指针系统表 $tt_bipointer$ 来实现，记录了所有对象与代理对象之间的联系. 科学任务的输入等于该任务所有直接前驱任务的输出数据类，科学任务的输出定义为该任务的输入数据类的代理类，也就是说，任务的输出数据对象是其直接前驱任务的输出数据对象的代理对象，通过系统维护的数据对象和代理数据对象间的双向指针，可以查询到一个任务的输出数据对象是由其前驱任务的哪些数据对象所生成的. 如图 4 中的 $tt_bipointer$ 结构所示，$SourceClass_oid$ 表示源类的 OID，$SourceObj_oid$ 表示源对象的 OID，$DeputyClass_oid$ 表示代理类的OID，$DeputyObj_oid$ 表示代理对象的 OID，可以通过 $DeputyObj_oid$ 直接查找到 $SourceObj_oid$.

由于科学任务的输出是其直接前驱科学任务的输出数据类的代理类，因此，一旦前驱任务的输出数据类中产生结果数据，对象代理模型的更新迁移将会自动触发后继任务的执行，并派生代理数据对象到其后继任务的输出数据类中. 同时，在系统表 $tt_bipointer$ 中增加 OID 映射关系. 对于 Select 和 Union 代理类中的代理对象，它与源对象在 $tt_bipointer$ 系统表中具有一个一对一的映射关系；对于 Join 代理类中的代理对象，由于一个代理对象是由多个源对象产生的，因此，它与源对象具有多条映射关系，其中每一条映射关系表示它和其中一个源对象的对应.

另外，数据跟踪不仅需要找到产生数据的数据源，还需要提供产生这些数据所经历的实验过程[15]. TOTEM 数据库将科学服务程序作为对象属性的读方法存放在 $tt_switching$ 系统表中. 如图 4 中的 $tt_switching$ 结构所示，$Class_oid$ 表示类的 OID，$Attr$ 表示对象属性，$Proc$ 表示科学服务程序，

借助 *tt_bipointer* 系统表中的 *DeputyClass_oid* 域，可以在 *tt_switching* 系统表中查找到 *DeputyObj_oid* 经过了哪些科学服务程序的计算.

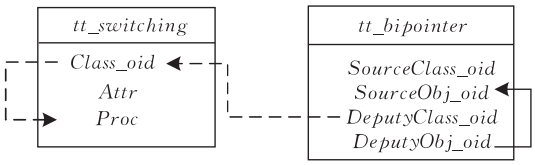
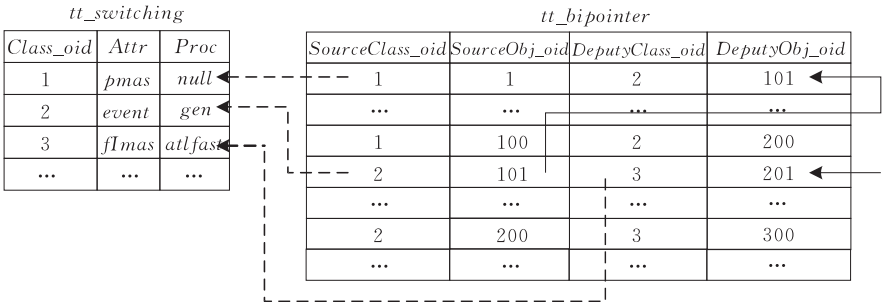


图 4　*tt_bipointer* 与 *tt_switching*

图 5 是 Atlas 工作流的 *tt_bipointer* 与 *tt_switching* 内部结构，详细记录了类 *gC* 中的数据对象与 *evts* 中的代理数据对象及 *evts* 中的代理数据对象与 *fC* 中代理数据对象的映射关系，其中 *Class_oid* 中的 1，2 和 3 分别表示类 *gC*，*evts* 和 *fC*. 从图中可以看到，类 *gC* 中的 *OID* 为 1 的数据对象经过科学服务程序 *gen* 的计算，得到 *evts* 类中 *OID* 为 101 的数据对象. 同时，*evts* 类中 *OID* 为 101 的数据对象经过科学服务程序 *atlfast* 和 *atlsim* 后在 *fC* 类中产生 *OID* 为 201 的代理数据对象.



tt_switching

Class_oid	Attr	Proc
1	*pmas*	*null*
2	*event*	*gen*
3	*fImas*	*atlfast*
…	…	…

tt_bipointer

SourceClass_oid	SourceObj_oid	DeputyClass_oid	DeputyObj_oid
1	1	2	101
…	…	…	…
1	100	2	200
2	101	3	201
…	…	…	…
2	200	3	300
…	…	…	…

图 5　Atlas 的 *tt_bipointer* 与 *tt_switching* 内部结构

3.2　跟踪查询

基于对象代理模型的数据跟踪方法可以根据系统表 *tt_bipointer* 与 *tt_switching* 查找到数据对象的来源和所经历的科学服务程序，即找出一个数据对象完整的派生信息，实现数据跟踪.

定义 3.　op_{dep}表示代理操作，其中 op_{dep}分成以下 3 种：

(1) Select

$C^d = select(proc(C^s))$是应用操作 *select* 到源类 C^s 中而形成的类，其中对选择出的数据对象应用了科学服务程序 *proc*. 假定一个对象 $o^d \in C^d$，根据操作 *select*，我们定义 o^d 在类 C^s 中的派生信息是 $P_{(C^s)}(o^d) = select^{-1}_{(C^s)}(o^d) = \{\langle o^s, C^s, proc\rangle\}$，其中 o^s 为 o^d 在 C^s 中的源数据对象.

(2) Union

$C^d = union(proc_1(C_1^s), proc_2(C_2^s), \cdots, proc_m(C_m^s))$ 是应用操作 *union* 到源类 $C_1^s, C_2^s, \cdots, C_m^s$ 中而形成的类. 假定一个数据对象 $o^d \in C^d$，根据操作 *union*，我们定义 o^d 在类 $C_1^s, C_2^s, \cdots, C_m^s$ 中的派生信息是 $P_{(C_1^s, C_2^s, \cdots, C_m^s)}(o^d) = union^{-1}_{(C_1^s, C_2^s, \cdots, C_m^s)}(o^d) = \{\langle o_i^s, C_i^s, proc_i\rangle \mid 1 \le i \le m\}$，其中，$o_i^s$ 为 o^d 在 C_i^s 中的源数据对象.

(3) Join

$C^d = join(proc_1(C_1^s), proc_2(C_2^s), \cdots, proc_m(C_m^s))$ 是应用操作 *join* 到源类 $C_1^s, C_2^s, \cdots, C_m^s$ 中而形成的类. 假定一个数据对象 $o^d \in C^d$，根据操作 *join*，我们定义 o^d 在类 $C_1^s, C_2^s, \cdots, C_m^s$ 中的派生信息是 $P_{(C_1^s, C_2^s, \cdots, C_m^s)}(o^d) = join^{-1}_{(C_1^s, C_2^s, \cdots, C_m^s)}(o^d) = \{\langle o_1^s, C_1^s, proc_1\rangle, \langle o_2^s, C_2^s, proc_2\rangle, \cdots, \langle o_m^s, C_m^s, proc_m\rangle\}$，其中 $o_1^s, o_2^s, \cdots, o_m^s$ 分别为 o^d 在 $C_1^s, C_2^s, \cdots, C_m^s$ 中的源数据对象.

如果 *join* 操作仅仅用于比较实验数据的结果，那么 *proc* 可以允许为空.

定义 4.　$C_1^b, C_2^b, \cdots, C_n^b$ 是基本科学类，C^d 是 $C_1^b, C_2^b, \cdots, C_n^b$ 经过多次 op_{dep} 操作生成的代理类. 考虑数据对象 $o^d \in C^d$. $C^d = op_{\mathrm{dep}}(proc_1(Deputy_1), proc_2(Deputy_2), \cdots, proc_k(Deputy_k))$，其中 $Deputy_j$ 是基本科学类 $C_1^b, C_2^b, \cdots, C_n^b$ 经派生得到的中间代理类，$k \ge j \ge 1$. 假设 $o^{d\prime} \in Deputy_j$，$o^{d\prime}$ 应用操作 op_{dep} 派生数据对象 o^d（定义 3），并且 $o^b \in C_i^b$，$n \ge i \ge 1$，o^b 派生 $o^{d\prime}$ 到 $Deputy_j$ 中（本定义递归），那么 o^b 派生 o^d 到 C^d 中. 因此，o^d 在 $C_1^b, C_2^b, \cdots, C_n^b$ 中的派生信息是 $P_{(C_1^b, C_2^b, \cdots, C_n^b)}(o^d) = \{C_1^{b*}, C_2^{b*}, \cdots, C_n^{b*}\}$. 其中 $C_1^{b*}, C_2^{b*}, \cdots, C_n^{b*}$ 分别是 $\{\langle o_1^b, C_1^b, proc_1\rangle\}, \{\langle o_2^b, C_2^b, proc_2\rangle\}, \cdots, \{\langle o_n^b, C_n^b, proc_n\rangle\}$ 的子集.

根据定义 4，考虑数据对象 $o^d \in C^d$，对 o^d 在类 $C_1^b, C_2^b, \cdots, C_n^b$ 中的跟踪查询 PQ_{o^d, C^d} 表示为 $PQ(o^d, C^d, (C_1^b, C_2^b, \cdots, C_n^b))$，它主要用于查找 o^d 在类 C_1^b，

$C_2^b,\cdots,C_n^b$中的源数据对象以及经历的科学服务程序.其算法表示如下：

Begin

$C^d = op_{\text{dep}}(proc_1(Deputy_1),\cdots,\ proc_k(Deputy_k))$;

$PQ(o^d, C^d, (Deputy_1, Deputy_2, \cdots, Deputy_k)) =$

$\{Deputy_1^*, Deputy_2^*, \cdots, Deputy_k^*\}$;

//其中 $Deputy_1^*, Deputy_2^*, \cdots, Deputy_k^*$ 分别是

//$\{\langle o^d_{Deputy_1}, Deputy_1, proc_1\rangle\}, \{\langle o^d_{Deputy_2}, Deputy_2,$

//$proc_2\rangle\}, \cdots, \{\langle o^d_{Deputy_k}, Deputy_k, proc_k\rangle\}$的子集;

For $i=1$ to n do

$C_i^{b*} = \text{Null}$;

For $j=1$ to k do

If $Deputy_j^* \neq \text{Null}$

Then $\{C_1^{b*}, C_2^{b*}, \cdots, C_n^{b*}\} = \{C_1^{b*}, C_2^{b*}, \cdots, C_n^{b*}\} \cup$

$PQ\ (o^d_{Deputy_j}, Deputy_j, (C_1^b, C_2^b, \cdots, C_n^b))$;

//$o^d_{Deputy_j}$表示 o^d在 $Deputy_j$中的源对象;

Return $(C_1^{b*}, C_2^{b*}, \cdots, C_n^{b*})$;

End

PQ_{o^d,C^d}计算 o^d在 $C_1^b, C_2^b, \cdots, C_n^b$中的派生信息，程序首先计算($Deputy_1, Deputy_2, \cdots, Deputy_k$)经过一层代理操作得到的派生信息$\{Deputy_1^*, Deputy_2^*, \cdots, Deputy_k^*\}$，然后不断递归计算每一个 $Deputy_j$中 $o^d_{Deputy_j}$的派生信息.

如图 5，如果我们要计算 fC 类中对象 OID 为 201 的数据对象 o^d 的派生信息，容易发现这个数据对象是 $evts$ 类中 OID 为 101 的数据对象经过科学服务程序 $atlfast$ 的计算而得到的代理数据对象，因此，它在 $evts$ 类中的派生信息是 $P_{evts}(o^d) = \{\langle 101, evts, atlfast\rangle\}$. 同样，$evts$ 类中 OID 为 101 的数据对象 $o^{d'}$ 是 gC 中 OID 为 1 的数据对象经过科学服务程序 gen 的计算而生成的，所以 OID 为 101 的数据对象在 gC 类中的派生信息为 $P_{gC}(o^{d'}) = \{\langle 1, gC, gen\rangle\}$. 因此，$OID$ 为 201 的数据对象 o^d 在 gC 类中的派生信息为 $P_{gC}(o^d) = \{\langle 1, gC, gen\rangle, \langle 101, evts, atlfast\rangle\}$. 由此看出，数据对象的派生信息能通过双向指针直接构造，节约了记录派生信息的存储空间，也不必使用逆向查询或逆向函数进行数据跟踪，提高了查询效率.

3.3 数据物化

在跟踪查询过程中，通常需要查询中间数据的属性值，如追踪 fC 类中 OID 为 201 的数据对象在 gC 中的派生信息时，需要查找 $evts$ 类中 OID 为 101 的数据对象属性值. 由于该数据对象是 gC 类中 OID 为 1 的数据对象的代理对象，为了减少额外的存储空间，该数据对象没有存储实际的数值，而是通过切换操作与它的源数据对象共享，因此查询它的属性值时，需要通过查找它的源数据对象的属性值再计算得到. 由于该数据对象的属性值是源数据对象的属性值经过复杂的科学程序计算得到，因此，会导致很低的追踪性能.

物化数据可以提高追踪的性能. 如果物化 $evts$ 类中 OID 为 101 的数据对象属性值，那么一旦用户提出查询这个数据对象的属性值，该属性值可以立即得到而不必切换到类 gC 中 OID 为 1 的数据对象，再计算出它的值，因此物化中间数据能帮助直接查询中间结果而不需要重新计算它们，从而增强了追踪性能. 然而，如果物化整个中间数据，那么可能造成很大的存储开销和维持代价，同时许多中间数据也许与用户感兴趣的最终数据毫不相关，因此没有必要完全物化它们.

为了减少存储和维持代价，我们提出一种部分物化中间数据的模式 PM，PM 只物化具有代理数据对象的中间数据. 例如，我们仅仅物化 $evts$ 类中需要被输入到科学服务程序 $atlfast$ 和 $atlsim$ 的对象，与完全物化 $evts$ 类中数据对象相比，大大地节省了存储空间.

相对于完全物化或不物化中间数据而言，PM 模式也能减少对象更新维持的代价. 例如，更新 gC 中的一个数据对象 o_{gC}时，PM 首先需要计算 $evts$ 类中它的代理数据对象 o^d_{evts}的值并判断是否引起 fC 类中 o^d_{evts}的代理数据对象 o^d_{fC}的更新，如果不会导致更新，根据 PM 的说明，o^d_{evts}一定未被物化，因此系统不需要消耗额外时间将被更新的 o^d_{evts}数据对象写入磁盘. 这种情况能保证与不物化这种方式相同的维持代价. 此时，如果采用完全物化这种方式，那么系统需要消耗额外时间将被更新的 o^d_{evts}对象写入磁盘.

如果会导致更新，那么 o^d_{evts}一定被物化，因此可根据被更新的 o^d_{evts}的值直接更新数据对象 o^d_{fC}的属性值，这种情况能保证与完全物化这种方式相同的维持代价. 此时如果采用不物化这种方式，即 o^d_{evts}未被物化，那么就需要根据基本数据对象 o_{gC}的属性值重新计算 o^d_{fC}的属性值，该过程经历了 gen 和 $atlfast$ 两个科学服务程序的计算，虽然此时 PM 需要消耗额外时间将被更新的 o^d_{evts}数据对象写入磁盘，但这个时间显然远远少于重新计算 o^d_{fC}的时间. 因此 PM 也能有效地减少对象更新维持的代价.

另外，由于最终数据的跟踪查询只与被物化的中间数据相关，因此 PM 模式并不影响追踪的性能.

4　性能测试

这个部分测试了 PM 模式的跟踪性能和维持代价，并与未物化中间数据模式 NM 进行了比较.

4.1　实验模型

实验模型按照图 3(b)的结构设计. 简单起见，我们假设只有两层代理类，其中第 1 层代理类 *evts* 存储程序 *gen* 的结果，并且完全地继承 *gC* 中的数据对象；第 2 层代理类 *fC* 存储科学服务程序 *atlfast* 的结果，并且继承 *evts* 中大约 40% 的数据对象. 为了模拟真实的环境，我们假设 *gen* 和 *atlfast* 分别运行 3s 和 2s. 模式 NM 和 PM 分别描述不物化和部份物化类 *evts*.

我们首先比较随着基本类大小的增加其两种模式在平均追踪查询、对象更新维持代价和存储开销上的性能，基本类的大小从 100000 变化到 500000 个对象. 其次分析基本类数据对象为 500000 时，随着查询率的增加，500000 个对象分别所需的追踪查询时间、更新维持时间以及系统负载. 更新维持代价和存储开销分别描述更新维持 *evts* 类和 *fC* 类的总时间以及它们总共占用的存储空间. 追踪查询或更新维持时，我们假设每次操作只追踪或更新一个数据对象，并计算半小时内，成功完成的操作数量. 实验环境采用赛扬 2.0GHz CPU，256MB 内存，Linux 操作系统.

4.2　实验结果

图 6(a)、图 6(b)和图 6(c)分别显示 PM 和 NM 模式的对象平均追踪时间，对象平均维持代价和存储开销. 图 6(d)、图 6(e)和图 6(f)分别显示在基本类具有 500000 个对象时，随查询率的增加，PM 和 NM 模式所需的追踪查询时间、更新维持时间以及系统负载.

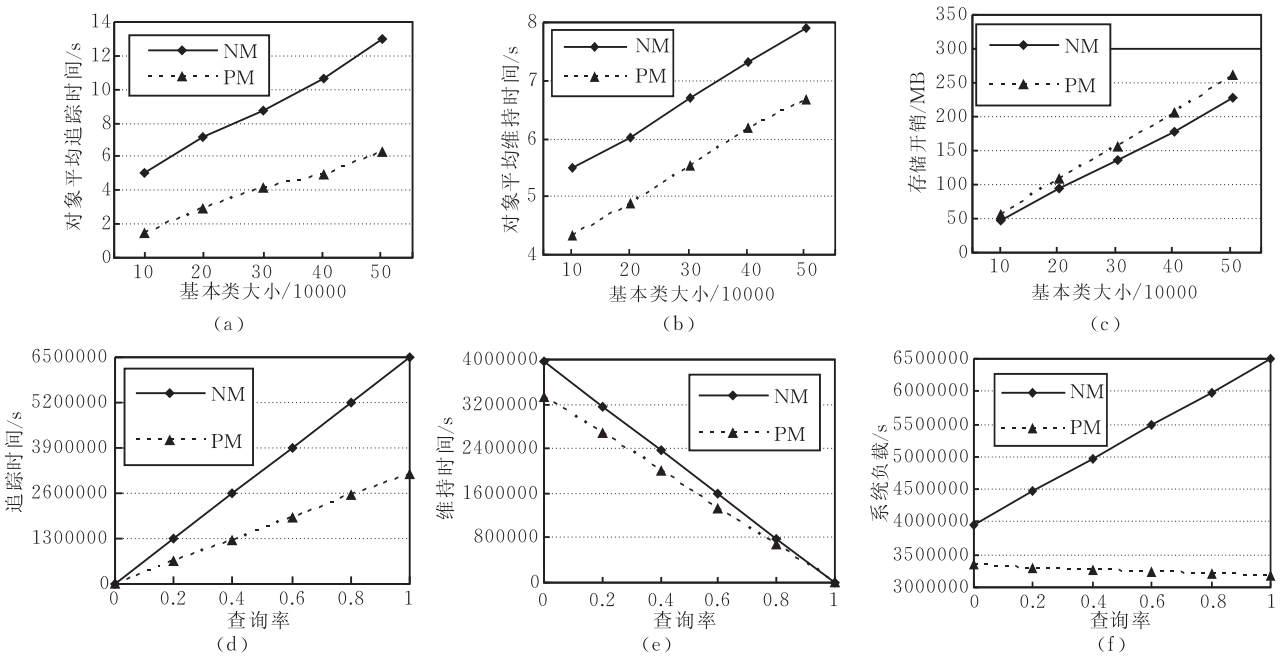


图 6　性能测试

图 6(a)显示了 PM 模式比 NM 模式所需的跟踪时间更短，并随基本类大小的增加，NM 模式与 PM 模式之间的时间差距越来越大. 图 6(b)显示了随着基本类大小的增加，PM 模式和 NM 模式的更新维持时间都增加，但 PM 模式有更好的维持性能. 图 6(c)显示 PM 模式的存储开销比 NM 模式大.

图 6(d)和图 6(e)显示随查询率的增加，PM 和 NM 的追踪查询时间和更新维持时间成比例的增加和减少. 显然，PM 的性能优于 NM 模式. 图 6(f)显示了 PM 模式具有更好的系统性能. 而 NM 模式随查询率的增大，所消耗的时间呈上升趋势.

根据上述分析，由此得出在运行时间较长的科学实验中，PM 模式显示出比 NM 模式更好的性能，尤其 PM 模式在基本类有大量数据对象而最终的数据对象仅有一小部分时显得特别有效. 当然，由于 PM 模式的存储开销比 NM 模式大，如果存在大量的最终数据对象，并且这些数据对象本身是由运行时间非常短的程序产生，那么 NM 模式可能也会显示出较好的性能.

5 相关工作

科学工作流技术研究在近些年得到很好的发展,特别是将 Web 服务技术应用到科学工作流中已引起了广泛关注[1,2,15],它们通常采用基于 XML 的语言如 WSFL,XLANG,BPEL4WS 等表达 Web 服务组合之间的控制流.然而,这些语言主要针对商业流程建模,对于将 Web 服务应用到以数据库为核心的科学工作流系统中的研究几乎没有被探讨.据我们所知,第一个倡导 WFMSs 和 DBMSs 相结合的系统是 ZOO[5].ZOO 是一个桌面实验管理环境,它使用面向对象语言 Moose 建模工作流之间的控制流,将其表示为任务与输入数据、输出数据间的关系.通过在关系上安排相应的规则来触发任务的执行.与 ZOO 类似,GridDB[6]同样基于数据库模式来构建工作流,不同的是,它关注的是网格环境,并使用了更加简单的关系模型来构造工作流.程序的输入和输出被建模为关系表,用户使用功能数据建模语言 FDM 定义程序以及程序的输入与输出间的关系.通过触发器的使用,在输入表中插入元组能触发程序的执行.文献[16]提出了一种紧密集成 WFMS 和 DBMS 的工作流建模语言.初始的输入数据和程序被描述为活动关系表,派生的数据和程序被描述为活动视图,工作流通过不断地定义活动视图而形成,程序的触发和执行通过 SQL 语句来完成.与 ZOO 和 GridDB 一样,集成对象代理数据库的科学工作流服务框架也是基于数据库管理系统为核心来实现科学工作流管理.它使用对象代理模型来建模工作流,表达工作流的控制逻辑,不仅能对同一任务的输入和输出进行建模,还对任务间的数据关系进行建模.最重要的是,利用对象代理数据库内部对象间的双向指针可以实现高效的数据跟踪.

作为一个关键组件,数据跟踪方法已经被科学工作流和数据库领域研究得很多了.主要包括注释和反向两种方法.反向方法主要是应用逆向查询或者逆向函数,由结果数据溯源到其源数据.而注释的方法则是将一个数据的派生历史搜集起来作为元数据,对源数据和处理过程进行描述.当前,大多工作流管理系统更依赖于注释[9-11],Chimera[9]分析虚拟数据目录并提出了表示操作序列的抽象有向无环图.一些数据跟踪系统如 myGird[10],CMCS[11]也使用 RDF 和 OWL 提供语义信息,以便于实现科学工作流在合作环境中的交互.但是,CMCS 对语义信息的支持比较有限,所以真正在数据跟踪信息中提供语义信息的只有 myGrid,它将本体概念注释到数据跟踪日志以提供丰富的语义.由于注释描述数据对象的派生历史,因此,既使数据是粗粒度的,注释信息占用的存储空间也可能比数据本身大.而在基于对象代理模型的数据跟踪方法中,数据对象的派生历史能通过双向指针直接构造,因此节约了大量的存储空间.

反向方法从存储角度看,特别对大量细粒度的数据似乎是更佳的选择,因为逆向函数或查询标识一整类数据的派生信息.许多数据库系统[7,17-19]都采取这个方法为用户提供面向数据的数据跟踪服务.Buneman[7]通过反向方法确定了数据跟踪的两种形式,why 表示数据产生的原因,而 where 提供产生数据的源数据.Trio[17]使用反向方法自动地通过视图查询或用户定义的函数确定元组的源数据.任何使用数据库查询和函数构造工作流的系统均可使用这样的技术.然而,并非所有的函数都有逆函数[20],虽然文献[18]提出一个框架以便根据弱逆向函数计算近似的源,但该文并没有提供产生弱逆向函数的机制.基于对象代理模型的数据跟踪方法与反向方法具有一些相似性,例如,也要求"反向"查询以便找到派生该数据对象的源数据对象,但它不必使用逆向查询或逆向函数计算数据的派生信息.反向方法虽然不会造成存储的开销,但每次一个数据的派生历史被需求时,它需要产生一个逆向函数或逆向查询去计算派生信息,在数据量大的情况下,执行效率较低.文献[21]也表明如果有大量的派生信息需求时,这种计算信息源的方法并不是好的选择.而基于对象代理模型的数据跟踪方法仅仅通过源数据对象与代理数据对象之间的双向指针便可查找到代理数据对象的派生信息,而不需要额外的计算代价.该方法也能有效地避免一些逆向函数不存在或计算不精确的问题.使用反向方法计算的派生信息通常需要存取信息源,而这些信息源可能很遥远或在一段时间内不可取.反向方法通常需要存储额外的辅助信息以便在数据跟踪时减少或避免整个源存取.如通过在数据仓库中存储额外的辅助信息,文献[19]能减少或完全避免存取信息源.而在基于对象代理模型的数据跟踪方法中,代理数据对象能物化一些来自源数据对象的属性值,从而有效地解决了以上问题.同时,从数据跟踪信息语义丰富程度的角度来考虑,显然反向方法所能给出的派生信息语义是非常有限的,通常只表示信息源极其经历的实验

过程. 基于对象代理模型的数据跟踪方法能随应用需要在代理类中增加自己的属性，可以更加丰富地表示数据对象的信息，弥补了反向方法表示数据不丰富的缺陷.

无论是哪种数据跟踪方法，一旦源发生改变，那么后续的数据需要使用一种类似于“维持表达式”的方式修改视图. 在我们的方法中，如果数据源改变，工作流随后的阶段可能自动地被触发，并自动更新派生的数据对象. 然而，无论注释或“反向”方法都不支持这个功能.

6　结论与展望

本文在分析了目前数据跟踪方法的基础上，研究并提出了一个基于对象代理模型的数据跟踪方法. 该方法基于集成对象代理数据库的科学工作流服务框架对科学工作流建模，借助对象之间的 *OID* 双向指针来实现数据跟踪，使得数据跟踪能提供比注释或反向方法更高的性能. 因为科学工作流的实验方法和实验数据通常需要被共享和重用，如何增强科学工作流的流程重用和数据重用，是我们下一步的工作.

参　考　文　献

[1] Cavalcanti M C, Targino R, Baião F, Rössle S C, Bisch P M, Pires P F, Campos M L M, Mattoso M. Managing structural genomic workflows using Web services. Data & Knowledge Engineering, 2005, 53(1): 45-74

[2] Jaeger E, Altintas I, Zhang J, Ludäscher B, Pennington D, Michener W. A scientific workflow approach to distributed geospatial data processing using Web services//Proceedings of the 17th International Conference on Scientific and Statistical Database Management. Santa Barbara, CA, 2005: 87-90

[3] Ludäscher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, Lee E A, Tao J, Zhao Y. Scientific workflow management and the Kepler system. Concurrency and Computation: Practice & Experience, 2005, 18(10): 1039-1065

[4] Dan M. Dynamic workflows and advanced data management for problem solving environments [M. S. dissertation]. Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2004

[5] Ailamaki A, Ioannidisz Y E, Livny M. Scientific workflow management by database management//Proceedings of the 10th International Conference on Scientific and Statistical Database Management. Los Alamitos, CA, 1998: 190-199

[6] Liu David T, Franklin M J. The design of griddb: A data-centric overlay for the scientific grid//Proceedings of the 30th International Conference on Very Large Data Bases. Toronto, Canada, San, 2004: 600-611

[7] Buneman P, Khanna S, Tan W C. Why and where: A characterization of data provenance//Proceedings of the 8th International Conference on Database Theory. London, UK, 2001: 316-330

[8] Greenwood M, Goble C, Stevens R, Zhao J, Addis M, Marvin D, Moreau L, Oinn T. Provenance of e-science experiments- experience from bioinformatics//Proceedings of the UK OST e-Science second All Hands Meeting. Nottingham, UK, 2003: 223-226

[9] Foster I, Vockler J, Wilde M, Zhao Y. Chimera: A virtual data system for representing, querying, and automating data derivation//Proceedings of the 14th International Conference on Scientific and Statistical Database Management. Edinburgh, Scotland, UK, 2005: 37-46

[10] Zhao J, Goble C A, Stevens R, Bechhofer S. Semantically linking and browsing provenance logs for E-science//Proceedings of the 1st International IFIP Conference on Semantics of a Networked World. Paris, France, 2004: 158-176

[11] Pancerella C, Hewson J, Koegler W, Leahy D et al. Metadata in the collaboratory for multi-scale chemical science//Proceedings of the International Conference on Dublin Core and Metadata Applications. Seattle, Washington, USA, 2003: 1-9

[12] Peng Z Y, Kambayashi Y. Deputy mechanisms for object-oriented databases//Proceedings of the 11th International Conference on Data Engineering. Los Alamitos, CA, 1995: 333-340

[13] Russell N, Hofstede A, Edmond D. Workflow Data Patterns. Brisbane: Queensland University of Technology, QUT Technical Report: FIT-TR-2004-01, 2004

[14] Zhai B X, Shi Y, Peng Z Y. Object deputy database language//Proceedings of the 4th International Conference on Creating, Connecting and Collaborating Through Computing. Los Alamitos, CA, 2006: 88-95

[15] Zhang J T, Pennington D, Michener W. Using web services and scientific workflow for species distribution prediction modeling//Proceedings of the 6th International Conference on Advances on Web-Age Information Management. Hangzhou, China, 2005: 610-617

[16] Shankar S, Kini A, DeWitt D J, Naughton J. Integrating databases and workflow systems. ACM SIGMOD Record, 2005, 34(3): 5-11

[17] Widom J. Trio: A system for integrated management of data, accuracy, and lineage//Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research. Asilomar, CA, USA, 2005: 262-276

[18] Woodruff A, Stonebraker M. Supporting fine-grained data lineage in a database visualization environment//Proceedings of the 13th International Conference on Data Engineering. Birmingham UK, 1997: 91-102

[19] Cui Y, Widom J. Practical lineage tracing in data warehouses//Proceedings of the 16th International Conference on Data Engineering. San Diego, California, USA, 2000: 367-378

[20] Simmhan Y L, Plale B, Gannon D. A survey of data provenance in e-science. ACM SIGMOD Record, 2005, 34(3): 31-36

[21] Bhagwat D, Chiticariu L, Tan W C, Vijayvargiya G. An annotation management system for relational databases//Proceedings of the 30th International Conference on Very Large Data Bases. Toronto, Canada, 2004: 900-911

WANG Li-Wei, born in 1980, Ph.D., lecturer. Her research interests mainly include workflow management and transaction processing.

HUANG Ze-Qian, born in 1984, Ph. D. candidate. His research interests mainly include workflow management and transaction processing.

LUO Min, born in 1981, Ph. D. candidate. His research interests mainly include workflow management and transaction processing.

PENG Zhi-Yong, born in 1963, Ph. D., professor, Ph. D. supervisor. His main research interests include Web data management, complex data management, trusted data management.

Background

This work is supported by the National Natural Science Foundation of China under grant No. 60573095, the New Century Excellent Talents of Education Ministry under grant No. NCET-04-0675, the National High Technology Research and Development Program of China under grant No. 2006AA12Z210, the Doctoral Foundation of Education Ministry under grant No. 20050486024, the Humanities and Social Science research base projects of the Education Ministry in 2005 under grant No. 05JJD870158, the Science and Technology research projects of Education Ministry under grant No. 107072, and the National Base Research and Development Program of China under grant No. 2007CB310806.

The most important function of scientific workflows is the way of recognizing data products, which is called data provenance. Data provenance provides derivation histories and explains the sources for data products. The solutions of determining data provenance in the literature usually involve annotations that comprise of the derivation history of a data product and inversion that generates a "reverse" query to find the origins supplied to derive a data product. Annotations may not scale well for fine-grained data as the complete annotations for the data may outsize the storage space required for the data itself. Inversion seems to be more optimal from a storage perspective since an inverse function or query identifies the provenance for an entire class of data. However, it requires a reverse query or function to be generated and executed to compute provenance every time the provenance of a data product is required.

In order to make up for the shortcomings of the above methods, this paper proposes a new data provenance method based on the bi-directional pointer mechanism of object deputy model. Not only derivation history of a data product can be directly constructed by bi-directional pointers between the data product and its sources, saving a mass of storage space, but also we can directly find source data of derived data by bi-directional pointers without computing provenance using inverse queries or inverse functions, increasing the querying efficiency.