基于数据服务的数据组合视图的优化更新

张 鹏[1),2)] 王桂玲[2)] 季 光[1),2)] 刘 晨[2)]

[1)](中国科学院计算技术研究所软件集成与服务计算研究分中心 北京 100190)

[2)](中国科学院研究生院 北京 100049)

摘 要 数据服务为实现跨域数据集成提供了统一的数据模型,并且可通过组合的方式支持用户定义数据视图,当底层的数据服务发生数据更新时,数据更新需要自下而上经过多个中间复合数据服务传播到顶层的数据视图.一类挑战性问题是,如何在这个传播的过程中保障数据视图的更新效率.为此,文中规范化数据服务的操作及其性质,并且根据用户的组合结果,通过复合数据服务的等价变换,生成运行效果相同的多种数据服务组合方案.同时,建立复合数据服务的更新代价模型,该模型能够利用底层数据服务的更新频率和请求频率,衡量复合数据服务的更新代价.基于该模型,文中提出最小化更新代价的数据视图的更新优化算法,利用该算法可以为用户推荐优化的数据服务组合方案.实验表明,该方法能够提高数据视图的更新效率.

关键词 数据服务;数据视图;数据服务组合;优化更新

中图法分类号 TP309 **DOI 号**: 10.3724/SP.J.1016.2011.02344

Optimization Update for Data Composition View Based on Data Service

ZHANG Peng[1),2)] WANG Gui-Ling[2)] JI Guang[1),2)] LIU Chen[2)]

[1)](*Research Laboratory of Software Integration and Service Computing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2)](*Graduate University of Chinese Academy of Sciences, Beijing* 100049)

Abstract Data service provides unified data model for cross-domain data integration, and generates data view through data services composition. A challenging problem is once a primitive data service is updated, how to propagate the update bottom-up to the top level via several intermediate composite data services, and to ensure the update efficiency of the data view. In this paper, the data operations and their properties are analyzed to facilitate transformation of composite data services, and generate some equivalent composite data services from a user defined composite data service. In addition, the time cost model for the incremental update of a composite data service is built. The model measures the time cost of updating a composite data service within a time unit in terms of the update frequency and data volume of the primitive data services. Based on the model, a data view update optimization algorithm is proposed to minimize the time cost. Experiments show that our approach can effectively improve the update efficiency of the data view.

Keywords data service; data view; data service composition; optimization update

1 引 言

随着互联网和服务计算技术的快速发展,为了解决跨域数据集成问题,越来越多的信息系统以服务的形式对外发布异构数据[1],形成了大量的数据服务.通过组合这些服务,可以实现跨域异构数据的集成,满足业务信息管理、业务流程管理等应用的需

收稿日期:2011-06-26;最终修改稿收到日期:2011-10-29.本课题得到国家自然科学基金(60903048,61033006)和北京市自然科学基金(4092046)资助.**张 鹏**,男,1984 年生,博士研究生,主要研究方向为服务组合、工作流管理、数据管理. E-mail: zhangpeng@software.ict.ac.cn. **王桂玲**,女,1978 年生,博士,助理研究员,主要研究方向为分布式系统、软件集成、服务组合和 Mashup 技术等. **季 光**,男,1984 年生,博士研究生,主要研究方向为 Internet 服务组合、服务计算. **刘 晨**,男,1980 年生,博士,助理研究员,主要研究方向为语义服务、工作流管理.

求[2-3]. 然而，现有的服务组合方法及工具通常以业务逻辑为中心，服务所输出的数据仅被视为业务流程的中间结果，无法直接支持以用户为中心的数据视图的构造（这里的数据视图是指一种通过数据服务组合得到的物化视图[4]），同时，用户也很难对数据视图进行调整. 近年来出现了一些针对数据服务组合的方法与工具，如 Aqualogic[5] 和 Damia[6]，这类工具通常为用户提供了简便的方式来定义数据服务及组合视图，有较好的易用性和灵活性. 但是，它们在数据视图的更新优化方面存在挑战. 当用户在某时间点上进行数据服务组合生成数据视图后，该视图伴随底层数据的变化不断重复运行，以实现数据视图的动态更新. 在这个过程中，由于使用数据服务组合工具的用户往往缺乏专业编程能力，无法在进行数据服务组合的过程中给出更新效率优化的组合方案，导致在更新数据视图时有较长延迟. 这在某些时间关键的应用中尤为突出，以社会应急管理为例，在突发事件的应急响应系统中，随着事态的进展，数据是不断动态变化的[7]，数据视图应及时体现底层数据变化从而为决策人员提供正确信息. 针对数据组合视图的更新优化问题，现有的数据服务组合方法及工具还没有给出令人满意的问题求解，因而也得到一些研究工作的重视，如文献[8]等.

基于上述研究背景，本文拟提出一种基于数据服务的数据组合视图的优化更新方法. 该方法借鉴数据缓存的思想，把数据缓存到数据视图中间产生的复合数据服务来减少数据更新时间. 同时，利用复合数据服务的等价变换法则，通过对数据操作的次序及组合方式进行变换，得到输出结果相同的若干候选数据服务组合方案，并且把数据服务组合方案的更新优化问题建模为 0-1 规划问题，通过遗传算法得到更新代价近似最优的数据服务组合方案，推荐给用户，使得当底层数据服务发生更新时，这个更新可以自下而上经过多个中间复合数据服务高效地传播到顶层的数据视图. 这里，数据传播是指从定义数据视图的底层数据服务发生数据更新开始到顶层数据视图发生数据更新结束的一系列数据服务的更新过程.

本文第 2 节描述和分析问题；第 3 节介绍数据服务及其组合操作；第 4 节给出复合数据服务的等价变换法则，并且基于该法则给出数据视图的更新代价模型以及使代价最小化的优化算法；第 5 节是实验和评价；第 6 节介绍相关工作；最后总结全文.

2　问题分析

本节通过分析一个贯穿全文的实例来说明本文研究的问题及其求解思路. 图 1 左边示出了一个数据服务组合实例，它为某跨国公司在进行供求信息统计时生成 A、B 两供货商供应零件情况的数据视图. 其中，DS_1 表示封装 A 供货商信息的数据服务，DS_2 表示封装 B 供货商信息的数据服务，通过对这两个服务的调用可获得供货商 A 和供货商 B 的标识 ID、名字、地址等基本信息. DS_3 表示封装零件信息的数据服务，它以供货商的标识 ID 为输入参数，可获取零件的库存数量、价格等详细信息. 将 DS_1 和 DS_2 先进行集合并操作，再以供货商标识 ID 属性（图中记为 R_i）的值为参数值逐次调用 DS_3 并连接 $DS_1 \cup DS_2$ 和 DS_3 的输出元组，即可得到所需要的数据视图，并记为复合数据服务的形式：$(DS_1 \cup DS_2) \triangleright_{R_i} DS_3$. 事实上，要得到同样的数据视图，往往有多个不同的数据服务组合方案. 图 1 右边示出了其中的一种，它以 DS_1 和 DS_2 供货商标识 ID 属性的值为参数值分别逐次调用 DS_3 并分别进行连接操作，最后再将结果进行集合并操作，记为 $(DS_1 \triangleright_{R_i} DS_3) \cup (DS_2 \triangleright_{R_i} DS_3)$. 这两种数据服务组合方案的输出完全相同，但通过下面分析可知，其数据视图的运行代价并不相同.

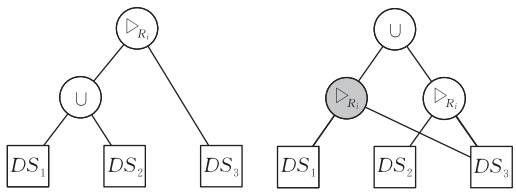


图 1　数据服务组合示例

首先，我们对在不同的应用场景下的数据视图的运行方式区分为被动方式和主动方式两种. 在被动方式下，系统为了响应请求，从底层的数据服务获取最新数据并通过层层计算，把计算结果呈现在数据视图上. 这种方式的优点是只有在需要数据的时候才进行计算，适用于数据不是频繁变化的场景. 在主动方式下，数据视图缓存所有的中间计算结果并监控基本数据服务的数据变化，当数据发生变化时，对受影响的中间结果重新计算和更新. 这种方式的优点是数据视图可以随时得到最新的数据，适用于处理不断更新的快速变化的数据及具有时间限制的场景. 用户可以根据自己的需要构造多个数据视图，

并为这些数据视图选择不同的运行方式.因此,复合数据服务具有两方面的运行代价:一方面是被动方式下应对请求的响应代价,另一方面是主动方式下应对数据变化的更新代价.其中,被动运行的服务可以复用主动运行的服务的缓存结果.例如,假设数据服务 $DS_1 \triangleright_{R_i} DS_3$ 为主动运行,那么,图 1 右边所示的被动运行的数据服务 $(DS_1 \triangleright_{R_i} DS_3) \cup (DS_2 \triangleright_{R_i} DS_3)$ 可以复用 $DS_1 \triangleright_{R_i} DS_3$ 的缓存结果而省略一步计算,从而提高更新效率.然而,由于图 1 左边数据服务 $(DS_1 \cup DS_2) \triangleright_{R_i} DS_3$ 和 $DS_1 \triangleright_{R_i} DS_3$ 没有公共的计算部分,无法复用 $DS_1 \triangleright_{R_i} DS_3$ 的缓存结果.因此,即使两种数据服务组合方案输出结果相同,$(DS_1 \cup DS_2) \triangleright_{R_i} DS_3$ 的更新效率也不如 $(DS_1 \triangleright_{R_i} DS_3) \cup (DS_2 \triangleright_{R_i} DS_3)$ 的更新效率.

在这个例子中,由于使用数据服务组合工具进行视图构造的用户在数据服务组合的过程中没有能力准确估计数据视图的更新效率,因此即使如图 1 右边所示的方案更优,有的用户仍可能选择如图 1 左边所示的方案.在上述例子中,我们主要考虑了缓存对更新效率的影响,事实上,各中间结果的计算代价、数据服务的请求频率和更新频率等因素都会影响数据服务组合方案的更新代价.因此,本文的研究工作就是在给定数据视图构建需求及若干复合数据服务运行方式、数据服务的请求频率和更新频率设置的条件下,求出一组等价的生成该数据视图的复合数据服务,并从中选择一组复合数据服务推荐给用户,使得数据视图的更新代价之和最小,简称视图选择问题.为求解此问题,本文首先定义了数据服务的基本元素及基本操作,在此基础上证明了复合数据服务的等价变换法则,该法则可以生成若干输出结果相同的候选复合数据服务.其次,本文研究了从候选复合数据服务中进行优选的方法.根据文献[9]的讨论,视图选择问题是一个 NP 问题,目前的主要方法是通过建模 0-1 规划的数学模型进行问题求解.因此,本文拟把用户给定的若干生成数据视图的复合数据服务及其运行方式建模为 0-1 规划的数学模型,然后利用遗传算法得到优化的数据服务组合方案向用户进行推荐.

3 数据服务及其组合操作

本文的数据服务是一种以提供数据资源的访问为目的的软件服务,它以统一的数据模型表示数据资源,并提供各种操作实现对该数据资源的访问.

3.1 数据服务的定义及数据模型

数据服务是对信息系统中数据资源的封装,其本身并不具备业务逻辑功能,在调用数据服务的前后并不会对外部世界的状态发生改变.因此,本文不必描述数据服务的前提和效果,而是重点描述数据服务的输入参数和输出数据模式.数据服务的形式化定义如下.

定义 1. 数据服务.数据服务 DS 是信息系统发布数据资源的软件服务,它实现了数据资源在给定数据模型 DM 下的统一表示.数据服务被表示为四元组 $DS=\langle id, params, schema, properties\rangle$,其中:

(1) id 是 DS 的唯一标识.本文以 DS_{id} 表示不同的数据服务.

(2) $params$ 是 DS 的输入参数,$params=\{p_1, p_2, \cdots, p_m\}$,其中 $p_i=\langle pn_i, pv_i\rangle$,$pn_i$ 为参数名,pv_i 为需在调用服务时赋予的参数值.DS 根据不同的输入参数可以返回不同的响应结果.

(3) $schema$ 是 DS 所发布的数据资源的数据模式.对于每次请求,DS 都会返回一个在给定数据模型 DM 下的数据实例,且该实例的元组遵循 $schema$.

(4) $properties$ 是 DS 的元信息属性集合,包括 DS 的服务质量以及 DS 的请求频率、更新频率等.

为了易于用户理解和使用,我们以嵌套关系模型作为数据服务的统一数据模型 DM,并且建立了类似电子表格的可视化数据服务组合环境"嵌套电子表格"[10].嵌套关系模型借鉴了非第一范式的嵌套关系,并对其中的属性和元组进行了有序化,该数据模型的形式化定义建立在嵌套关系模式和嵌套关系的基础上,相关定义参考文献[10].

本文使用 $Schema(DS)$,$Tuple(DS)$ 表示 DS 的嵌套关系模式、调用 DS 返回的嵌套元组,显然有 $\forall t \in Tuple(DS)$,t 遵循 $Schema(DS)$.$Tuple(DS, \{\langle pn_i, pv_i\rangle\})$ 表示以 pv_i 作为参数 pn_i 的值调用 DS 得到的嵌套元组,在不关心 pn_i 的具体取值的情况下,可简写为 $Tuple(DS, \{pv_i\})$.$Table(DS)=\langle Schema(DS), Tuple(DS)\rangle$ 表示调用 DS 返回的嵌套关系,也称嵌套表.

3.2 数据服务组合的基本操作

关系代数的操作对象为关系表和视图,具体操作包括选择、投影、并、连接等.相对而言,数据服务组合的操作对象为数据服务,我们前期工作通过定义嵌套表上的增加列、更新列、过滤元组等十余种操

作，实现了嵌套关系代数的表达能力[11]，并将不同的数据服务组合为复合数据服务，实现了异构数据集成. 本文不再重复这些操作的具体定义，而是关注利用这些操作如何生成复合数据服务. 从这种视角出发，本文把数据服务操作划分为集合操作，无数据依赖和有数据依赖的连接操作以及单数据源操作.

定义 2.　集合操作. 对于数据模式相同的数据服务DS_1和DS_2，DS_1和DS_2上的集合操作实现了两者所输出元组上的集合运算，生成新的数据服务$DS_3=DS_1 \bullet DS_2$. 设$Schema(DS_1)=Schema(DS_2)=R$，那么$Schema(DS_3)=R$，$Tuple(DS_3)=Tuple(DS_1) \bullet Tuple(DS_2)$（$\bullet$可以是并、交、差中的一种，分别用符号$\cup$、$\cap$、$-$表示）.

定义 3.　无数据依赖连接操作. 对于数据服务DS_1和DS_2，DS_1和DS_2上的无数据依赖连接操作通过建立嵌套元组连接两者的输出元组，生成新的数据服务$DS_3=DS_1 \bowtie DS_2$，设$Schema(DS_1)=R_1$，$Schema(DS_2)=R_2$，则$Schema(DS_3)=\{R_1,R_2\}$，$Tuple(DS_3)=\{\langle R_1,Tuple(DS_1)\rangle,\langle R_2,Tuple(DS_2)\rangle\}$.

定义 4.　有数据依赖连接操作. 对于数据服务DS_1和DS_2，设$R_i \in Schema(DS_1)$，DS_2依赖于DS_1中R_i的连接操作，是指通过以R_i的属性值为参数值逐次调用DS_2并连接DS_1和DS_2的输出元组，生成新的数据服务$DS_3=DS_2 \triangleright_{R_i} DS_1$. 设$Schema(DS_1)=\{R_1,\cdots,R_m\}$，$m \geqslant i$；$Schema(DS_2)=R'$，则$Schema(DS_3)=\langle R_1,\cdots,R_m,R'\rangle$，并且$Tuple(DS_3)=\{\{\langle R_1,r_1\rangle,\cdots,\langle R_m,r_m\rangle,\langle R',Tuple(DS_2,\{r_i\})\rangle\}\}$.

定义 5.　单数据服务操作：对于数据服务DS_1，其上的单数据服务操作f包括增加列、更新列、过滤元组等等. 该函数分别对DS_1的数据模式和元组集合进行转换（分别用f_R和f_r表示），转换后的数据服务为$DS_2=f(DS_1)$，其中$Schema(DS_2)=f_R(Schema(DS_1))$，$Tuple(DS_2)=f_r(Tuple(DS_1))$.

4　数据视图的更新优化

数据视图的更新优化的基本思想如下：首先根据用户定义数据视图的数据服务组合方案生成输出结果相同的数据服务组合方案，即对于每个复合数据服务生成若干输出结果相同的候选复合数据服务；然后从这些候选复合数据服务中进行优选，使得总体的响应代价和更新代价之和最小. 本节首先探讨复合数据服务的等价变换法则，然后把复合数据服务的选择问题建模为 0-1 规划问题，最后利用遗传算法对该问题进行求解.

4.1　复合数据服务的等价变换法则

复合数据服务的等价变换，就是在保证复合数据服务输出的嵌套表不变的前提下，对数据操作的次序及组合方式进行变换. 为了简化表示，这里用“=”连接输出相同的数据服务. 复合数据服务的等价变换法则主要包括交换律、结合律、分配律等等.

性质 1.　并操作的交换律. 参与并操作两个数据服务的次序交换后，输出的嵌套表不变，即$DS_1 \cup DS_2=DS_2 \cup DS_1$.

证明.　根据集合操作的定义，DS_1和DS_2以及操作生成的复合数据服务的数据模式相同，不妨设$Schema(DS_1)=Schema(DS_2)=R$，则$Schema(DS_1 \cup DS_2)=Schema(DS_2 \cup DS_1)=R$. 另外，由于$Tuple(DS_1) \cup Tuple(DS_2)=Tuple(DS_2) \cup Tuple(DS_1)$，$Tuple(DS_1 \cup DS_2)=Tuple(DS_2 \cup DS_1)$. 因此原式得证.　　证毕.

性质 2.　并操作的结合律. 3 个数据服务参与并操作，如果并操作次序变化，则输出的嵌套表不变，即$(DS_1 \cup DS_2) \cup DS_3=DS_1 \cup (DS_2 \cup DS_3)$.

证明.　同上.

同理，交操作和无数据依赖的连接操作也具有交换律和结合律.

性质 3.　有数据依赖的连接操作对并操作的分配律：两个数据服务进行并操作之后与一个数据服务进行有数据依赖连接，与这两个数据服务分别与该数据服务进行有数据依赖连接再进行并操作，输出的嵌套表相同，即$(DS_1 \cup DS_2) \triangleright_{R_i} DS_3=(DS_1 \triangleright_{R_i} DS_3) \cup (DS_2 \triangleright_{R_i} DS_3)$.

证明.　根据集合操作的定义，DS_1和DS_2的数据模式相同，设$Schema(DS_1)=Schema(DS_2)=\{R_1,\cdots,R_m\}$，$Schema(DS_3)=R'$，则等号两端的数据模式均为$\{R_1,\cdots,R_m,R'\}$. 另外，根据有数据依赖的连接操作的定义及集合运算法则，$Tuple((DS_1 \cup DS_2) \triangleright_{R_i} DS_3)=\{\{r,\langle R',Tuple(DS_3,r)\rangle\} \mid r \in Tuple(DS_1) \cup Tuple(DS_2)\}$，$Tuple(DS_1 \triangleright_{R_i} DS_3) \cup Tuple(DS_2 \triangleright_{R_i} DS_3)=\{\{r,\langle R',Tuple(DS_3,r)\rangle\} \mid r \in Tuple(DS_1)\} \cup \{\{r,\langle R',Tuple(DS_3,r)\rangle\} \mid r \in Tuple(DS_2)\}=\{\{r,\langle R',Tuple(DS_3,r)\rangle\} \mid r \in Tuple(DS_1) \cup Tuple(DS_2)\}$，因此原式得证. 证毕.

同理，$(DS_1 \cap DS_2) \triangleright_{R_i} DS_3=(DS_1 \triangleright_{R_i} DS_3) \cap$

$(DS_2 \triangleright_{R_i} DS_3)$.

性质 4. 数据服务先后与两个数据服务进行有数据依赖连接操作,若这两个数据服务之间无数据依赖关系,则连接操作的次序对结果无影响,即 $(DS_1 \triangleright_{R_i} DS_2) \triangleright_{Rj} DS_3 = (DS_1 \triangleright_{Rj} DS_3) \triangleright_{R_i} DS_2$,其中 $R_j, R_j \in Schema(DS_1)$.

证明. 证明方法参考性质 3.

性质 5. 数据服务 DS_1 先后与两个数据服务进行有数据依赖连接操作,若连接所依赖的属性相同,则操作结果等价于两个数据服务先进行无数据依赖连接,DS_1 再与该结果进行有数据依赖连接,最后进行单数据源操作解嵌套,也即是 $(DS_1 \triangleright_{R_i} DS_2) \triangleright_{R_i} DS_3 = f(DS_1 \triangleright_{R_i} (DS_2 \bowtie DS_3))$,其中 $R_i \in Schema(DS_1)$,f 表示对 $\{R_2, R_3\}$ 的解嵌套变换.

证明. 证明方法参考性质 3.

性质 6. 单数据服务操作 f,数据服务 DS_1 和 DS_2,如果 $f_r(Tuple(DS_1) \cdot Tuple(DS_2)) = f_r(Tuple(DS_1)) \cdot f_r(Tuple(DS_2))$,那么 $f(DS_1 \cdot DS_2) = fDS_1 \cdot fDS_2$.

证明. 由于 $Schema(DS_1 \cdot DS_2) = Schema(DS_1) = Schema(DS_2)$,$f$ 对相同数据模式的转换结果相同,因此等号两端的数据模式相同. 另外,$f(DS_1 \cdot DS_2)$ 的元组集合为 $f_r(Tuple(DS_1)) \cdot f_r(Tuple(DS_2))$,$fDS_1 \cdot fDS_2$ 的元组集合为 $f_r(Tuple(DS_1)) \cdot f_r(Tuple(DS_2))$,根据 f_r 满足的条件,两端相同. 因此原式得证. 证毕.

4.2 复合数据服务的更新代价模型

为建立复合数据服务的更新代价模型,下面以图 1 右边包含的 $(DS_1 \cup DS_2) \triangleright_{R_i} DS_3$ 和 $DS_1 \triangleright_{R_i} DS_3$ 两个复合数据服务组成的集合为例,给出具体的建模过程. 为简便起见,例子中省略了有数据依赖连接的依赖属性. 我们约定用大写字母表示集合和函数名,以带下标的小写字母表示集合元素,以带大写字母上标的小写字母表示向量,该向量的长度与大写字母所表示的集合元素个数相同. 在本例中,$S = \{(DS_1 \cup DS_2) \triangleright_{R_i} DS_3, DS_1 \triangleright_{R_i} DS_3\} = \{s_1, s_2\}$,表示定义数据视图的复合数据服务集合,$S$ 中的元素简称为原始数据服务. 式(1)给出了复合数据服务的更新代价模型:

$$c_S = c_P + c_R \tag{1}$$

其中 c_S 表示原始数据服务的更新代价,c_P 表示原始数据服务中以主动方式运行的数据服务的更新代价,c_R 表示原始数据服务中以被动方式运行的数据服务的响应代价. 为了计算它们,首先需要得到如下 3 个矩阵.

$E(s_i)$ 为数据服务 s_i 的等价数据服务集合,该集合可以通过人工方式应用等价变换法则得到. E 为 S 中所有元素通过等价变换得到的数据服务集合,E 中元素简称等价数据服务. 本例中,$E(s_1) = \{(DS_1 \cup DS_2) \triangleright_{R_i} DS_3, (DS_2 \cup DS_1) \triangleright_{R_i} DS_3, (DS_1 \triangleright_{R_i} DS_3) \cup (DS_2 \triangleright_{R_i} DS_3), (DS_2 \triangleright_{R_i} DS_3) \cup (DS_1 \triangleright_{R_i} DS_3)\} = \{e_1, e_2, e_3, e_4\}$,$E(s_2) = \{DS_1 \triangleright_{R_i} DS_3\} = \{e_5\}$,$E = \{e_1, e_2, e_3, e_4, e_5\}$. $\boldsymbol{ES}$ 是 $|S| \times |E|$ 的 0-1 布尔矩阵,当且仅当 $E(s_i) = e_j$ 时 $\boldsymbol{ES}(i,j) = 1$,否则 $\boldsymbol{ES}(i,j) = 0$.

$B(e_i)$ 为数据服务 e_i 的中间数据服务集合. 所谓 e_i 的中间数据服务是指包括 e_i 以及构成 e_i 的所有中间产生的数据服务. B 为 E 中所有元素的中间数据服务的集合,其中的元素简称中间数据服务. 本例 $B = \{DS_1 \cup DS_2, DS_2 \cup DS_1, DS_1 \triangleright_{R_i} DS_3, DS_2 \triangleright_{R_i} DS_3, (DS_1 \cup DS_2) \triangleright_{R_i} DS_3, (DS_2 \cup DS_1) \triangleright_{R_i} DS_3, (DS_1 \triangleright_{R_i} DS_3) \cup (DS_2 \triangleright_{R_i} DS_3), (DS_2 \triangleright_{R_i} DS_3) \cup (DS_1 \triangleright_{R_i} DS_3)\}$. $\boldsymbol{BE}$ 是 $|E| \times |B|$ 的 0-1 布尔矩阵,当且仅当 $B(e_j) = b_i$ 时 $\boldsymbol{BE}(i,j) = 1$,否则 $\boldsymbol{BE}(i,j) = 0$.

$A(b_i)$ 为 b_i 所包含的原子数据服务集合,A 为 B 中所有元素所包含的原子数据服务集合. A 中元素简称原子数据服务. 在本例中,$A = \{DS_1, DS_2, DS_3\}$. $\boldsymbol{AB}$ 是 $|B| \times |A|$ 的 0-1 布尔矩阵,当且仅当 $A(b_j) = a_i$ 时 $\boldsymbol{AB}(i,j) = 1$,否则 $\boldsymbol{AB}(i,j) = 0$.

$\boldsymbol{p}^S$ 是 0-1 布尔向量,1 表示 S 中的原始数据服务以主动方式运行,0 表示 S 中的原始数据服务以被动方式运行. 为了便于说明,在本例中假设 $\boldsymbol{p}^S = [0,1]$. $\boldsymbol{r}^S$ 和 $\boldsymbol{u}^A$ 分别表示各原始数据服务的请求频率和各原子数据服务的更新频率,这些频率可以通过一段时间内的统计得到,为了便于说明,本例中假设 $\boldsymbol{r}^S = (1,1)$,$\boldsymbol{u}^A = (1,1,1)$. $\boldsymbol{c}^B$ 表示各中间数据服务的单一运行代价. 中间服务的单一运行代价是指该表达式完成最后一步操作的代价. 例如,$(DS_1 \cup DS_2) \triangleright_{R_i} DS_3$ 的单一运行代价是指完成 DS_3 与 $DS_1 \cup DS_2$ 进行有数据依赖连接的代价,而不包含 $DS_1 \cup DS_2$ 的代价. $\boldsymbol{c}^B$ 可以通过实际测试得到. 为了便于说明,本例中假设 $\boldsymbol{c}^B = (1,1,2,2,3,3,1,1)$. 在现实中,有数据依赖连接 $DS_1 \triangleright_{R_i} DS_2$ 的运行代价一般高于集合操作,并且随着 DS_1 输出的元组个数的增加而增长. 为体现这一性质,本文把 $DS_1 \triangleright_{R_i} DS_3$ 和 $DS_2 \triangleright_{R_i} DS_3$ 的代价均设为 2,高于集合操作的代

价 1；而($DS_1 \cup DS_2$)输出的元组数为 DS_1 和 DS_2 元组数之和，考虑到有数据依赖连接本身的常量开销，本文将($DS_1 \cup DS_2$) $\triangleright_{R_i} DS_3$ 的代价设为 3，即取值在 $DS_1 \triangleright_{R_i} DS_3$ 和 $DS_2 \triangleright_{R_i} DS_3$ 的代价和与其中一者的代价之间.

以上信息构成复合数据服务的更新代价模型的全部输入，下一步给出复合数据服务更新代价的计算过程，其中包括计算以主动方式运行的中间数据服务的更新代价和计算以被动方式运行的中间数据服务的响应代价. 这里使用符号“¬”、“∧”和“⊗”表示长度相同的向量之间对应元素的逻辑非、逻辑与和点乘运算，其结果仍为向量；符号“×”表示矩阵和向量之间的乘法，其运算法则与结果与线性代数中的定义一致；符号“′”表示矩阵或向量的转置；函数 L 表示把矩阵中非 0 元素变成 1，从而将整数矩阵转化为 0-1 矩阵.

对于以主动方式运行的数据服务，它运行中的所有中间数据服务都将缓存，并随着原子数据服务的数据更新而更新. 若某个原始数据服务需以主动方式运行，则它所有的等价数据服务都可能被缓存，因此所有可能被缓存的等价数据服务为 $\boldsymbol{p}^E = L(\boldsymbol{p}^S \times \boldsymbol{ES})$. 其中 $\boldsymbol{ES}$ 表示等价数据服务和原始数据服务的对应关系，向量 $\boldsymbol{p}^S$ 表示哪些原始数据服务以主动方式运行，两者相乘并进行 0-1 化即可得到哪些等价数据服务被缓存. 本例中，该步骤的结果为 $(0,0,0,0,1)$，时间复杂度是 $O(n^2)$. 若某个等价数据服务被选中且被缓存，那么它所对应的中间数据服务都将被选中且被缓存，因此所有被缓存且被选中的中间数据服务为 $\boldsymbol{x}_{BP} = L((\boldsymbol{p}^E \wedge \boldsymbol{x}^E) \times \boldsymbol{BE})$. $\boldsymbol{p}^E$ 为上一步所得，$\boldsymbol{x}^E$ 表示哪些等价数据服务被选中，这里表示为 $(x_1, x_2, x_3, x_4, x_5)$，两者通过逻辑与运算可以得到哪些等价数据服务被选中且被缓存；$\boldsymbol{BE}$ 表示中间数据服务与等价数据服务的对应关系，通过和 $\boldsymbol{p}^E \wedge \boldsymbol{x}^E$ 相乘并进行 0-1 化可以得到哪些中间数据服务被选中且被缓存. 该步骤的结果为 $(0,0,x_5,0,0,0,0,0)$，时间复杂度是 $O(n+n^2)$.

当某原子数据服务发生数据更新时，该更新将传播到所有依赖它的中间数据服务，因此总更新代价为 $c_P = ((\boldsymbol{x}_{BP} \otimes \boldsymbol{c}^B) \times \boldsymbol{AB}) \times (\boldsymbol{u}^A)'$. $\boldsymbol{x}_{BP}$ 为上一步所得，$\boldsymbol{c}^B$ 表示中间数据服务的单一运行代价，两者相乘可以得到被选中且被缓存的中间数据服务的单一运行代价；$\boldsymbol{AB}$ 表示原子数据服务和中间数据服务的对应关系，通过与 $\boldsymbol{x}_{BP} \otimes \boldsymbol{c}^B$ 相乘可以得到各原子数据服务的更新代价，经进一步与各原子服务的更新频率 $\boldsymbol{u}^A$ 转置相乘得到一个标量，即所有原子数据服务更新的总代价. 该步骤的结果为 $4x_5$，时间复杂度是 $O(n+n^2+n)$.

对于以被动方式运行的数据服务，它们由原始数据服务的服务请求触发. 由于某些中间数据服务的数据被缓存，在响应服务请求时，这些中间数据服务可以直接返回缓存数据，从而不产生响应代价. 因此，计算它们的响应代价只需要考虑被选中且未被缓存的中间数据服务 $\boldsymbol{x}_{BR} = L(\boldsymbol{x}^E \times \boldsymbol{BE}) \wedge \neg \boldsymbol{x}_{BP}$，它可以通过逻辑与运算得到. $L(\boldsymbol{x}^E \times \boldsymbol{BE})$ 表示被选中的中间数据服务，本例为 $L(x_1, x_2, x_3 + x_4 + x_5, x_3 + x_4, x_1, x_2, x_3, x_4)$，$\neg \boldsymbol{x}_{BP}$ 表示未被缓存的中间数据服务. 在本例中，$\neg \boldsymbol{x}_{BP} = (1,1,\neg x_5,1,1,1,1,1)$，两者进行逻辑与得到被选中且未被缓存的中间数据服务. 该步骤的时间复杂度是 $O(n^2+n)$. 每个等价数据服务都对应一个原始数据服务，因此在不考虑哪些等价数据服务被选中的情况下，各等价数据服务的请求频率为 $\boldsymbol{r}^E = \boldsymbol{r}^S \times \boldsymbol{ES}$，在本例中，该步骤结果为 $\boldsymbol{r}^E = (1,1,1,1,1)$，时间复杂度是 $O(n^2)$.

当某个数据服务被请求时，响应代价为该数据服务所包含的未被缓存的中间数据服务的运行代价之和，因此总响应代价为 $c_R = (((\boldsymbol{r}^E \otimes \boldsymbol{x}^E) \times \boldsymbol{BE}) \otimes \boldsymbol{c}^B) \times \boldsymbol{x}'_{BR}$. 其中，$\boldsymbol{r}^E \otimes \boldsymbol{x}^E$ 表示被选中的等价数据服务的请求频率；$(\boldsymbol{r}^E \otimes \boldsymbol{x}^E) \times \boldsymbol{BE}$ 表示被选中的中间数据服务的请求频率. 通过与各中间数据服务的单步运行代价 $\boldsymbol{c}^B$ 逐一相乘，得到的 $((\boldsymbol{r}^E \otimes \boldsymbol{x}^E) \times \boldsymbol{BE}) \otimes \boldsymbol{c}^B$ 表示在不考虑哪些中间数据服务被缓存的情况下，各中间数据服务的响应代价，本例为 $(x_1, x_2, 2(x_3 + x_4 + x_5), 2(x_3 + x_4), 3x_1, 3x_2, x_3, x_4)$；然后与前面得到的 $\boldsymbol{x}_{BR}$ 的转置相乘，在求和过程中消除了被缓存的中间数据服务代价，从而得到总响应代价. 该步骤的时间复杂度为 $O(n+n^2+n+n)$.

4.3　数据视图的更新优化算法

为了得到优化的数据服务组合方案，这里把本文的视图选择问题建模为 0-1 规划问题，其中包含两个好处：首先，本文的输入能自动转换为 0-1 规划的描述；其次，0-1 规划问题已经被广泛研究，目前已有很多快速求解该问题的工具. 本文的 0-1 规划的描述如式(2)所示.

$$\min \quad c_S = c_P + c_R \quad \text{s.t.} \quad \boldsymbol{x}^E \times \boldsymbol{ES}' = \boldsymbol{I}^S \tag{2}$$

其中 $c_P + c_R$ 是目标函数，$\boldsymbol{x}^E$ 表示从等价数据服务中选取的组合方案，其中被选中的表示为 1，否则为 0.

约束条件表示数据服务组合方案中一个原始数据服务只能选择一个对应的等价数据服务，$\boldsymbol{I}^S$ 为长度为 $|S|$ 并且所有元素均为 1 的列向量. 由于目标函数不满足叠加原理，因此上述的 0-1 规划模型是非线性的[12]. 由于非线性 0-1 规划问题是 NP 问题，因此本文使用遗传算法求解该问题的近似最优解，算法 1 给出了具体描述.

算法 1. 数据视图的优化更新算法(DVOU).

输入：$S, \boldsymbol{p}^S, \boldsymbol{r}^S, \boldsymbol{u}^A, \boldsymbol{c}^B$

输出：$\boldsymbol{x}^E$

1. $E=ApplyEquivalentFormula(S)$; //应用等价变换

2. $B=GenerateB(E)$; //得到中间数据服务

3. $A=GenerateA(B)$; //得到原子数据服务

4. $\boldsymbol{ES}=getES(E,S)$, $\boldsymbol{BS}=getBS(E,S)$, $\boldsymbol{AB}=getAB(B,S)$;

5. 随机对 $\boldsymbol{x}^E$ 进行编码，这里 $\boldsymbol{x}^E$ 本身是 0-1 字符串；给出一个有 N 个染色体的初始群体，这里 N 等于 $2|\boldsymbol{x}^E|$. $POP(1), t:=1$;

6. 对每个编码检查它是否满足 $\boldsymbol{x}^E \times \boldsymbol{ES}' = \boldsymbol{i}^S$，如果不满足，则返回步 5，直到得到 N 个染色体的初始群体；

7. 对群体 $POP_i(t)$ 计算它的适应函数，这里等于目标函数 $c_P + c_R$；

8. 若停止规则满足，则算法停止，这里停止规则当最优个体的适应度不再上升或者迭代次数达到 $|\boldsymbol{x}^E| \times 100$ 次时；否则，计算概率 $P_i = f_i \Big/ \sum_{j=1}(f_j)$，$i=1,2,\cdots,N$ 并且以上述概率分布从 $POP(t)$ 中随机选取一些染色体构成一个种群 $NewPOP(t+1)=\{pop_j(t)|j=1,2,\cdots,N\}$；

9. 通过交配，得到一个有 N 个染色体的 $CrossPOP(t+1)$，前两个染色体交换第 $|\boldsymbol{x}^E|/2$ 个位置以后基因，后两个染色体交换第 $|\boldsymbol{x}^E|/2+1$ 个位置以后的基因；

10. 以概率 p，使染色体的一个基因发生变异，形成 $MutPOP(t+1)$；$t: t+1$，这里 $p=0.02$. 一个新的群体 $POP(t)=MutPOP(t)$；返回步 6；

11. 返回适应函数最小的 $\boldsymbol{x}^E$.

根据上面的分析，计算目标函数的时间复杂度是 $O(n^2)$，因此算法 1 的时间复杂度至多为 $O(|\boldsymbol{x}^E| \times 100 \times O(n^3))$. 在本例中，经过该算法得到的优化方案是 $\boldsymbol{x}^E=(0,0,1,0,1)$，也即选择 $(DS_1 \triangleright_{R_i} DS_3) \bigcup (DS_2 \triangleright_{R_i} DS_3)$ 和 $DS_1 \triangleright_{R_i} DS_3$. 这种优化方案的更新代价是 7.

5 实验与评价

我们从原型系统和实验两个方面来验证本文提出方法的合理性和有效性. 我们开发了相应的软件工具(当前主要包括数据服务建模工具及以用户为中心的数据服务组合工具)用以支持数据服务的建模以及数据组合视图的构建，并集成了本文提出的上述数据组合视图的更新优化算法. 对于进行数据组合视图构造的用户，由于他们往往缺乏专业编程能力，无法在进行数据服务组合的过程中给出更新效率优化的组合方案. 因此，当用户完成一个数据组合视图的构造后，本软件工具中的数据视图推荐模块负责利用本文提出的算法向用户推荐优化的组合方案. 该模块首先遍历数据视图组合脚本，把数据服务组合操作转换为规范操作(集合操作、无数据依赖和有数据依赖的连接操作以及单数据源操作)，输出规范操作下的数据服务组合描述，然后运行算法 DVOU，得到等价的优化组合方案推荐给用户. 用户可参照推荐的优化组合方案对已有方案进行调整或者直接选择系统推荐的可执行方案来替代原有方案.

为进一步验证本文提出的方法对数据组合视图运行效率的优化效果，我们通过把 TPC-H① 基准测试中的数据集进行服务化封装得到测试的实验数据，基于由此得到的一组数据服务进行了实验验证. 选择 TPC-H 主要基于以下几个事实：首先，TPC-H 所选用的数据及查询广泛参考了业界的实际应用，是对业务需求的抽象和概括；其次，TPC-H 中的查询结果均用于回答业务中的关键问题，用于辅助用户进行决策，也被称为“决策支持基准测试”. 不过，TPC-H 规定的最小数据规模为 1 G. 由于本实验主要测试数据服务的请求频率、更新频率以及运行方式对数据服务组合方案的更新时间的影响，所以这里只采用 10 兆字节的数据规模. 在未来的工作中，我们会研究数据量对数据服务组合方案的更新时间的影响.

TPC-H 规范描述了一个多国零件生产与销售的应用场景，其中涉及 8 个数据表，关系如图 2 所示，其中箭头表示外码之间的依赖关系. 每个表被封装为表 1 所示的数据服务.

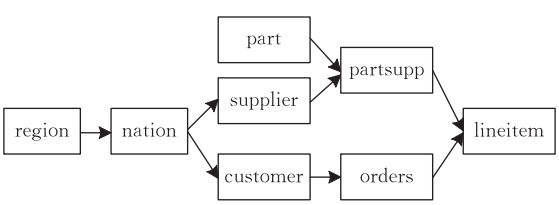


图 2 数据表

① http://www.tpc.org/tpch/

表 1　原子数据服务

原子数据服务	编号	原子数据服务的结构及业务含义
region	r	⟨id:001,Params:{regionkey}, Schema:{regionkey,name,comment}⋯⟩ 国家的地区信息
nation	n	⟨id:002,Params:{nationkey}, Schema:{nationkey,name,comment}⋯⟩ 国家信息
part	p	⟨id:003,Params:{partkey}, Schema:{partkey,name,mfgr,type, brand⋯}⋯⟩ 零件详细信息
supplier	s	⟨id:004,Params:{suppkey, nationkey},Schema:{regionkey, nationkey,name,address⋯}⋯⟩ 供货商信息,标识供货商所在国家
partsupplier	ps	⟨id:005,Params:{partkey,suppkey}, Schema:{partkey,suppkey,availqty, supplycost⋯}⋯⟩ 供货商供应零件信息
customer	c	⟨id:006,Params:{custkey,nationkey}, Schema:{custkey,nationkey,name address,phone⋯}⋯⟩ 客户信息,标识了客户所在的国家
order	o	⟨id:007,Params:{orderkey,custkey}, Schema:{orderkey,custkey,status, price,date,clerk⋯}⋯⟩ 订单信息,标识订单来自哪个客户
lineitem	l	⟨id:008,Params:{orderkey,partkey, suppkey},Schema:{orderkey, partkey,suppkey,number, quantity,discount,tax⋯}⋯⟩ 标识来自哪个订单,零件与供货商的交易信息

实验所需要的输入条件包括 $\boldsymbol{ES}$,$\boldsymbol{BE}$ 和 $\boldsymbol{AB}$ 矩阵以及 $\boldsymbol{c}^B$,$\boldsymbol{r}^S$,$\boldsymbol{u}^A$ 和 $\boldsymbol{p}^S$ 向量,需要求解的是 $\boldsymbol{x}^E$,即优化的数据服务组合方案,并且得到复合数据服务集合的更新代价 c_S.

下面介绍各输入条件的取值与设置情况.根据复合数据服务等价变换法则,我们从 12 个原始数据服务等价变换得到 41 个等价数据服务,并且对这些等价数据服务进行分解,得到 82 个中间数据服务.这些中间数据服务由 8 个原子数据服务构成.利用这些条件,我们分别得到了 $\boldsymbol{ES}$,$\boldsymbol{BE}$ 和 $\boldsymbol{AB}$ 矩阵,其大小分别为 41×12、82×41 和 8×82.通过对中间数据服务的实际测试,我们得到各中间数据服务的运行代价,从而得到 $\boldsymbol{c}^B$ 的具体取值.

根据 TPC-H 规范,所有查询的请求频率相同.为方便起见,这里把 $\boldsymbol{r}^S$ 设为 12×1 所有项均为 1 的列向量.根据 TPC-H 规范,order 表和 lineitem 表需要通过更新函数进行更新,更新频率与请求频率成一定比例,该比例由比例因子控制.在本实验中,依照 TPC-H 规范给出的比例,我们把请求频率与更新频率之比从 1∶2 递增至 1∶11. TPC-H 对其它表的更新频率没有明确说明,但是为了体现更新频率的变化,我们将其它表的更新频率设为 order 表和 lineitem 表的一半.因此,$\boldsymbol{u}^A$ 为 8×1 的列向量,共有 10 个,用编号为 1～10 表示,其中对应 order 和 lineitem 的数据服务的项分别为 2,3,4,⋯,9,10,11,其余项分别为 1,2,2,⋯,5,5,6.

$\boldsymbol{p}^S$ 代表数据服务的运行方式,它的取值不受 TPC-H 规范的约束.这里,$\boldsymbol{p}^S$ 设定为如下几种运行方式:第一种方式是将 S 中的复合数据服务都设为被动运行;最后一种方式是将 S 中的复合数据服务都设为主动运行;其余方式是按复合数据服务的分类逐渐将被动运行的数据服务设为主动运行.由于复合数据服务共分 5 类别,因此共 6 种运行方式.

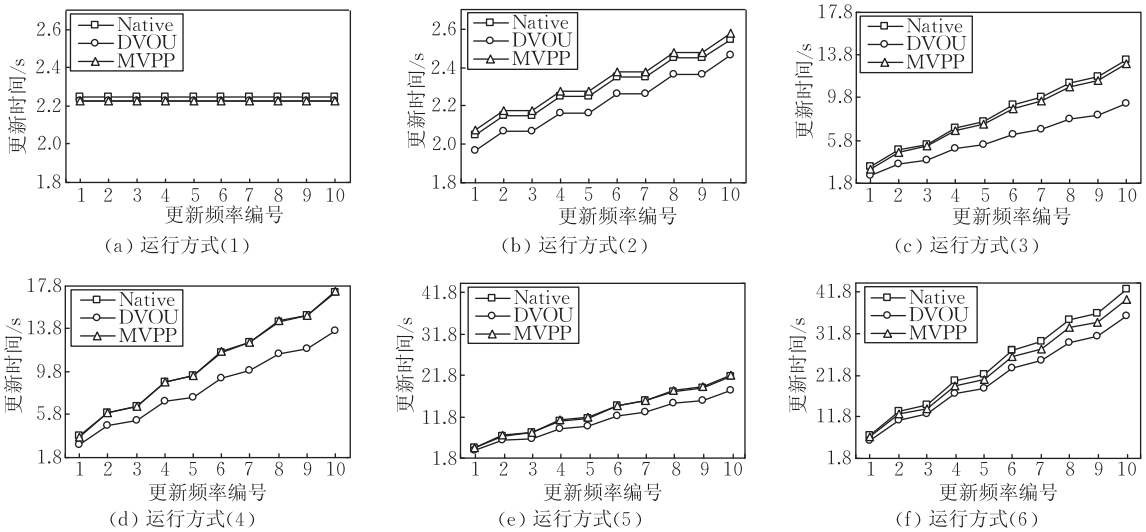
在这 6 种情况下,被动运行的数据服务数量逐渐减少,主动运行的数据服务数量逐渐增加,我们把这些运行方式编号为(1)～(6).至此,我们已经得到了在 TPC-H 测试集下运行优化算法的全部输入条件,接着我们可以求出复合数据服务的优选方案,并得到整个复合数据服务集合的更新代价.我们在 6 种运行方式下使用编号 1～10 种不同的更新频率,生成实验结果.

表 2　复合数据服务

类别	编号	复合数据服务的业务含义	复合数据服务的结构
市场参与者	S_1	各国供货商数量统计	$f(n\triangleright s)$
	S_2	各国客户数量统计	$f(n\triangleright c)$
产品供求情况	S_3	零件产量最多的国家	$f(n\triangleright s\triangleright ps)$
	S_4	零件需求数量最大的国家	$f(n\triangleright c\triangleright o\triangleright l)$
	S_5	零件数量上供过于求的国家	$f(n\triangleright((s\triangleright ps)\triangleright(s\triangleright\!\!\triangleleft(c\triangleright o\triangleright l))))$
市场规模	S_6	供货商与客户总数最大的国家	$f(n\triangleright s)\cup f(n\triangleright c))$
	S_7	生产、销售额最大的国家	$f(f(n\triangleright s\triangleright ps)\cup f(n\triangleright c\triangleright o))$
各国零件销售概况	S_8	各国大型零件的平均售价	$f(n\triangleright s\triangleright ps\triangleright p)$
	S_9	各国各品牌的实际销售量	$f(n\triangleright s\triangleright(ps\triangleright p\triangleright l))$
对外贸易	S_{10}	出口零件数量最多的国家	$f(l\triangleright(n\triangleright s)\triangleright(n\triangleright c\triangleright o))$
	S_{11}	地区内国际贸易最盛行的地区(以零件数量计)	$f(l\triangleright(r\triangleright n\triangleright s)\triangleright(r\triangleright n\triangleright c\triangleright o))$
	S_{12}	国内贸易最盛行的地区(以零件数量计)	$f((r\triangleright n)\triangleright(s\triangleright\!\!\triangleleft(c\triangleright o))\triangleright l)$

该实验除了比较优化前和优化后的复合数据服务集合的总体更新时间，而且还比较了本文提出的DVOU方法和MVPP[17]方法. MVPP是一个关于多个查询的全局最优查询执行计划，是针对数据库视图优化的算法，其本身不支持用户对数据视图缓存的指定. 针对本实验场景，我们对MVPP算法进行了适当修改，使之生成在不考虑用户指定数据服务运行方式的情况下的优化方案，并且允许用户指定运行方式，然后与本文提出的DVOU方法进行对比. 在分析实验结果之前，这里对实验结果做出如下几点说明：

图3(a)～(f)中的纵轴“更新时间”是指完成指定请求频率 $\boldsymbol{r}^S$ 和更新频率 $\boldsymbol{u}^A$ 所规定操作所需要的时间，数值越小表明越好. 由于请求频率是不变的，横轴“更新频率编号”从1递增至10，根据前面更新频率的设定，原子数据服务的更新频率逐步提高，使得更新时间也逐步提高.



(a) 运行方式(1) (b) 运行方式(2) (c) 运行方式(3)

(d) 运行方式(4) (e) 运行方式(5) (f) 运行方式(6)

图3 复合数据服务更新效率优化的实验结果

图4表示了本文的DVOU方法与MVPP方法为用户推荐的方案在图3所示的不同运行方式下更新效率提高的百分比. 这里首先根据图3计算优化后推荐的方案和原始方案在运行方式(1)～(6)对应同一更新频率的更新时间的平均值，然后计算总的更新时间减少的百分比，该数值就是更新效率提高的百分比. 通过实验分析，我们可以得出如下结论：

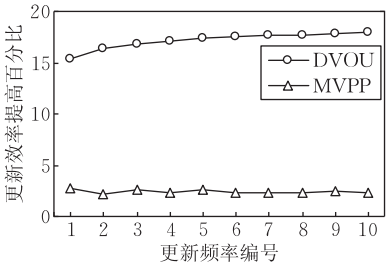


图4 更新效率优化百分比

本文提出的DVOU方法为用户推荐的方案优于MVPP方法. 除了运行方式(1)和(2)，其它运行方式，DVOU均把更新效率提高了15%以上，同时在不同的更新频率下，更新效率也均提高了15%以上. 这表明了DVOU方法在不同的更新频率和运行方式下都具有良好的效果. 相比之下，MVPP在各种运行方式下的优化效果不是很明显.

在无数据服务以主动方式运行的情况下，本文提出的DVOU方法和MVPP方法为用户推荐的方案的优化效果相同；在其它情况下，DVOU方法为用户推荐的方案更好. 当所有服务都以被动方式运行时，DVOU方法与MVPP方法为用户推荐的方案是相同的. 但是，当用户选择主动方式运行数据服务后，DVOU方法会进行调整，从图3(a)～(f)可以看出，DVOU方法推荐的方案均不相同，而MVPP则不会做出任何调整，这使得在运行方式(2)～(6)的情况下，DVOU方法为用户推荐的方案好于MVPP方法.

6 相关工作

为了提高数据视图的更新效率，经典的数据集成方法主要围绕着物化的数据库视图展开. 利用物化视图提高视图更新效率的工作包括两类：一类是把待优化的视图直接保存为物化视图，当底层数据

变化时，实现该物化视图受影响的部分数据的更新，即物化视图的增量更新方法[13]；另一类是围绕待优化的中心视图定义一些相关的物化视图作为数据缓存，利用这些访问速度较快的缓存提高中心视图的查询效率，即利用缓存优化查询方法[14]. 其中，增量更新方法在适用的应用场景上具有一定的局限性：只有基本数据库关系的变化能够精确地以元组的增加、删除与修改的形式进行描述时，才能够生成增量更新元组. 在本文探讨的多源异构数据集成场景中，数据服务位于其它管理域中，且其精确变化难以被捕捉到；即使要跟踪数据元组的改变，也需要大量的计算开销，影响数据服务的实际更新效率. 因此在本文的数据视图的更新效率的优化上，这种方法难以直接使用. 当查询内容相对固定时，利用缓存进行查询优化的方法往往具有优势. 在本文中，数据服务组合生成的数据视图可以视为对底层数据服务的联合查询，是相对固定的. 因此，本文为复合数据服务的中间结果建立了缓存，通过更新需要更新的缓存来减少计算量.

现有的服务组合方法及工具通常以业务逻辑为中心，近年来虽然出现了一些数据服务组合工具可用于构造数据视图，如基于可视化数据流的 Yahoo Pipes①、IBM Damia[6] 等以及基于电子表格的 SheetMusiq[15]、SpreadATOR[16] 和 Mashroom[11] 等，但它们普遍忽视了数据视图的更新效率问题. Hassan 等人注意到此问题，并提出了 Mashup 算子合并和重排方法[8]，通过发现 Mashup 公共算子序列避免重复计算，并且给出了算子重排规则，增加发现公共子序列的概率，但是该工作的前提是统一的 RSS 种子的数据模型，因此算子重排规则不适合指导以嵌套关系为数据模型的数据服务进行等价变换，并且该工作也没有考虑更新频率和请求频率. Yang 等人提出的一种 MVPP 方法[17]，它是一个关于多个查询的全局最优查询执行计划，当所有数据服务都以被动方式运行时，MVPP 对方案的选择依据是“在不缓存任何中间结果的情况下使响应代价最小化”，因此本文优化方案与 MVPP 做出的选择相同. 但是，当用户选择主动方式运行数据服务时，MVPP 则不会进行调整.

7　结　论

基于数据服务组合的数据视图的优化更新问题是一个亟待解决的问题，现有的数据服务组合方法和工具还未给出令人满意的求解方法. 本文提出了基于数据服务的数据组合视图的优化更新方法，该方法通过复合数据服务的等价变换，生成运行效果相同的多种数据服务组合方案，并且基于复合数据服务的更新代价模型，提出了最小化更新代价的数据视图的更新优化算法，利用该算法可以为用户推荐优化的数据服务组合方案. 实验表明该方法提高了数据视图的更新效率.

优化数据视图的更新效率既可以在构造时静态进行，也可以在运行时动态进行. 本文所采用的方法为静态分析方法：通过了解基本数据服务的参数（包括更新频率和元组数量等），对影响更新效率的参数进行静态估计，以此选择优化组合方案. 下一步，我们将研究动态和静态结合的分析方法. 值得指出的是，除进行视图的更新优化之外，综合考虑多种指标的数据视图优化也是下一步有待研究的工作. 这是因为，数据视图的质量、访问权限、使用费用、使用策略或协议约束等其它各种非功能特征也可能是进行数据视图选取时需要考虑的重要因素.

参　考　文　献

[1] Gray J. The next database revolution//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France, 2004: 1-4

[2] Fagin R, Haas L, Hernández M et al. Clio: Schema mapping creation and data exchange. Conceptual Modeling: Foundations and Applications, 2009: 198-236

[3] Manolescu I, Florescu D, Kossmann D. Answering XML queries on heterogeneous data sources//Proceedings of the 27th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., 2001: 241-250

[4] Silberschatz A, Henry F K, Sudarshan S. Database System Concepts. New York, USA: McGraw-Hill, 2002

[5] Carey M. Data delivery in a service-oriented world: The BEA aquaLogic data services platform//Proceedings of the ACM SIGMOD International Conference on Management of Data. Chicago, Illinois, USA, 2006: 695-705

[6] Altinel M, Brown P, Cline S et al. Damia: A data mashup fabric for intranet applications//Proceedings of the International Conference on Very Large Data Bases. Vienna, Austria, 2007: 1370-1373

[7] Jiang Hui, Huang Jun. The study on the issues of scenario evolvement in real-time decision making of infrequent fatal emergencies. Journal of Huazhong University of Science and Technology (Social Science Edition), 2009, 23(1): 104-108 (in Chinese)

① Yahoo Pipes, Inc. http://pipes.yahoo.com/

(姜卉，黄钧. 罕见重大突发事件应急实时决策中的情景演变. 华中科技大学学报：社会科学版，2009，23(1)：104-108)

[8] Hassan O A H, Ramaswamy L, Miller J A. Enhancing scalability and performance of mashups through merging and operator reordering//Proceedings of the IEEE International Conference on Web Services. Miami, Florida, USA, 2010: 171-178

[9] Lin Zi-Yu, Yang Dong-Qing, Wang Teng-Jiao, Song Guo-Jie. Research on materialized view selection. Chinese Journal of Software, 2009, 20(2): 93-213(in Chinese)
(林子雨，杨冬青，王腾蛟，宋国杰. 实视图选择研究. 软件学报，2009，20(2)：93-213)

[10] Yang Shao-Hua. User-steered development of Internet-based situational applications [Ph. D. dissertation]. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 2009(in Chinese)
(杨少华. 用户主导的互联网情景应用构造研究[博士学位论文]. 中国科学院计算技术研究所，北京，2009)

[11] Wang G, Yang S, Han Y. Mashroom: End-user mashup programming using nested tables//Proceedings of the 18th International Conference on World Wide Web. Madrid, Spain: ACM, 2009: 861-870

[12] Avriel M. Nonlinear Programming: Analysis and Methods. New Jersey, USA: Prentice-Hall, 1976

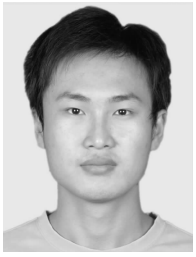
[13] Gupta A, Mumick I S, Subrahmanian V S. Maintaining views incrementally. ACM SIGMOD Record, 1993, 22(2): 157-166

[14] Chaudhuri S, Krishnamurthy R, Potamianos S et al. Optimizing queries with materialized views//Proceedings of the 11th International Conference on Data Engineering. Taipei, Taiwan, China, 1995: 190-200

[15] Liu B, Jagadish H V. A spreadsheet algebra for a direct data manipulation query interface//Proceedings of the International Conference on Data Engineering. Shanghai, China, 2009: 417-428

[16] Kongdenfha W, Benatallah B, Vayssiere J et al. Rapid development of spreadsheet-based web mashups//Proceedings of the International Conference on World Wide Web. Madrid, Spain, 2009: 851-860

[17] Yang J, Karlapalem K, Li Q. Algorithm for materialized view design in data warehousing environment//Jarke M, Carey M J, Dittrich K R eds. Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97). Athens: Morgan Kaufmann Publishers, 1997: 136-145



ZHANG Peng, born in 1984, Ph.D. candidate. His major research interests include workflow management, service composition, data management.

WANG Gui-Ling, born in 1978, Ph. D., assistant researcher. Her major research interests include end-user programming, data integration.

JI Guang, born in 1984, Ph.D. candidate. His major research interests include service composition, service network.

LIU Chen, born in 1980, Ph. D., assistant researcher. His major research interests include semantic service, workflow management.

Background

The form of information systems is evolving: with the rapid development of networking and data management technologies, different enterprises and departments have accumulated large amount of applications and data. Many emerging systems are built on these cross-domain resources. On the other hand, application systems change dramatically due to rapid requirement shift, and professional-dominated application development can no longer satisfy the users. A lot of light weight applications, such as data mashups, have been created by end users with flexible and rapid development tools.

Data service provides unified data model for cross-domain data integration, and enable users to generate data view through data services composition. A challenging problem is once a primitive data service is updated, how to propagate the update bottom-up to the top level via several intermediate composite data services, and to ensure the update efficiency of the data view.

In this paper, the data operations and their properties are analyzed to facilitate transformation of composite data services, and generate some equivalent composite data services from a user defined composite data service. In addition, the time cost model for the incremental update of a composite data service is built. The model measures the time cost of updating a composite data service within a time unit in terms of the update frequency and data volume of the primitive data services. Based on the model, a data view update optimization algorithm is proposed to minimize the time cost.