ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2012/06(03)-0221-16
DOI: 10.3778/j.issn.1673-9418.2012.03.003

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-51616056

iViewer：利用数据服务即时生成跨域数据视图*

温 彦[1,2+], 刘 晨[2], 韩燕波[2]

1. 中国科学院 研究生院, 北京 100190
2. 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

iViewer: Service-Based View Construction Method for Just-in-Time Sharing Business Data across Organizations*

WEN Yan[1,2+], LIU Chen[2], HAN Yanbo[2]

1. Graduate University of Chinese Academy of Sciences, Beijing 100190, China
2. Software Integration and Service Computing Branch Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

+ Corresponding author: E-mail: wenyan@software.ict.ac.cn

WEN Yan, LIU Chen, HAN Yanbo. iViewer: service-based view construction method for just-in-time sharing business data across organizations. Journal of Frontiers of Computer Science and Technology, 2012, 6(3): 221–236.

Abstract: Today, more and more enterprises and organizations choose to depend on the Internet to realize business collaborations. For business users, they often need to gather and analyze information from multiple organizations correctly and immediately to make improvisations. Key issues behind this process are how to construct cross-organizational data views in a just-in-time way and maintain the consistency between data views and sources. This paper proposes a dynamic cross-organizational view construction method fitting in with the Internet environment, called iViewer. The main ideas of iViewer method are to encapsulate autonomous and heterogeneous data sources as a set of data services, and compose them to generate a data view in a visualized and user-friendly way. It develops a polling based view updating algorithm to maintain the consistency between the data views and sources, so that the data view can be updated as the data sources get changed. The paper discusses the principles and procedures of iViewer in

*The National Natural Science Foundation of China under Grant No. 60970131 (国家自然科学基金); the Key Program of the National Natural Science Foundation of China under Grant No. 61033006 (国家自然科学基金重点项目); the National Grand Basic Research 973 Program of China under Grant No. 2007CB31080 (国家重点基础研究发展规划(973)).

Received 2011-08, Accepted 2011-10.

detail, and proves the effects of the iViewer method in practical “fire-emergency-equipments-dispatching” case, in which a cross-organizational view of fire rescuing equipment information from multiple departments is dynamically constructed for the command center.

Key words: data service; data view; cross-organizational data; just-in-time construction; view maintenance

摘 要：越来越多的企业和组织选择通过广域、开放的互联网作为其协作平台，业务决策者往往需要即时汇聚并综合分析来自多个部门的资源信息以进行临机决策。如何即时构建满足用户需求的跨组织数据视图，动态维护视图和数据源之间的一致性是需要求解的一个关键问题。提出了一种互联网环境下跨组织业务数据视图的动态生成方法 iViewer，利用数据服务来封装自治、异构和动态变化的数据源；通过可视化和易用的数据服务组合操作来动态构建数据视图；提出了一种基于轮询的视图动态更新算法，维护数据源和数据视图的一致性，从而使得数据视图能够随数据源的变化而自主变化；详述了 iViewer 方法的原理和过程，并通过一个火灾应急处置场景中，面向指挥中心的跨部门火灾救援设备数据视图的动态生成过程例证了 iViewer 方法的效果。

关键词：数据服务；数据视图；跨组织数据；即时构建；视图维护

文献标识码：A 中图分类号：TP31

1 引言

近年来，广域、开放、聚众的互联网正在演变为迄今人类最大的协同计算平台，越来越多的企业和组织都选择依托互联网来实现动态业务协作，即时共享业务数据，提升自身的竞争能力。对于业务用户尤其是决策者而言，构造一个能够即时汇聚跨组织资源信息的数据视图以辅助临机决策过程往往是非常重要的。例如，在火灾应急处置过程中，可用的消防设备、消防人员以及消防车辆等信息均是分布在不同的组织或部门中的(如政府应急办、消防队、企业，甚至是平民家中)，为了快速、合理地作出火灾救援的各类决策，决策人员需要即时获得一个能够动态反映可用消防设备的位置、库存量，以及消防人员、消防车辆的调度派遣情况的数据视图。

在互联网环境下，构造上述数据视图的难点在于：互联网提供了一个开放、自治的数据共享环境；数据源在数据结构、数据格式、访问方式以及数据约束等多个方面的异构性；上述方面随着数据源自身需求的变化而不断自主演化。例如，在上述火灾应急处置场景中，火灾救援设备、人员以及车辆的信息来自于多个自治的组织，它们可能以文件、数据库等形式存储，且这些信息是动态变化的；同时设备会随着救灾进程而不断消耗，人员有可能会轮换休息或被派遣执行其他任务，车辆有可能因为交通事故而逾期未归等。如何构建一个综合的业务数据视图，能够集成异构数据源，并动态反映其变化，即时向决策人员呈现最新内容是一个核心挑战。

在传统方法中，数据视图的构造多基于一个静态的数据中心(关系数据库或采用模式集成方法构造的数据集成系统[1-2])，通过分析用户提交的查询来构建数据视图，用户可以在本地维护数据视图中的数据，这种视图也被称为物化视图(materialized view)[3]。数据源由于插入、删除和修改而引起数据更新时，物化视图需要及时地进行更新，以保证视图数据与数据源的一致性(consistency)和新鲜性(freshness)，这个过程就是视图维护(view maintenance)。视图维护通常采用增量维护的方式，计算已有视图中的数据与数据源的差异，只更新两者相异的部分。当前已有多种视图维护算法，例如 Strobe 算法、Sweep 算法、PVM(parallel view maintenance)算法、自维护算法[3]等。然而，传统的增量更新算法在互联网环境下具有很大的实现难度：首先，物化

视图的增量更新需要数据源在数据发生变化时主动通知数据源的使用者，是一种侵入式的数据更新方法。互联网环境下的数据源是自治的，与使用者是松耦合的，数据源无法知道和预测其所有的使用者，从而也无法提供实时的变更通知。其次，互联网环境下的跨组织业务数据缺乏统一的数据模型和语义规范，导致数据的增量部分很难被计算，其发布的更新消息也很难被广泛理解和使用。因此，在互联网环境下，探索一种跨组织的、松耦合的、非侵入式的数据视图构建与维护方法具有重要意义。

针对上述问题，本文提出了一种基于数据服务的数据视图动态构建方法 iViewer。该方法引入服务的思想，将数据源暴露为数据服务，并将其作为数据视图构建的基本元素；同时，将数据视图定义为一组数据服务的组合模式，并在应用过程中，基于视图所包含的数据服务的更新特征，不断对其轮询调用，以获取数据源的最新数据，从而维护数据视图的新鲜性和正确性。结合一个火灾应急处置场景中，面向指挥中心的跨部门火灾救援设备数据视图的动态生成过程来例证 iViewer 方法的效果。

本文组织结构如下：第 2 章介绍 iViewer 方法的基本原理；第 3 章给出数据服务和数据视图的定义，并探讨基于数据服务组合来构建数据视图的方法和技巧；第 4 章介绍数据服务更新特征的定义和描述方法，并据此给出基于轮询的数据视图的即时更新算法；第 5 章通过一个实际案例来分析 iViewer 方法的实际效果，并介绍了相关实现；第 6 章对相关工作进行分析和比较；第 7 章给出结论和下一步的工作。

2　iViewer 方法的基本原理

构建互联网环境下的跨组织业务数据视图，需要解决两个关键问题：第一是要解决跨组织数据的异构性，为数据提供一个统一的数据表示和操控方式，使得用户能够根据业务需求的变化动态选取合适的数据内容，作为数据视图构建的基本元素。为此，引入了数据服务的思想，使用标准化的方式对数据源进行封装，屏蔽跨域数据源的异构性。与传统 Web 服务不同的是，数据服务将数据和对数据的操作都视为数据集成过程中的第一元素，强调对数据本身的建模，暴露了业务数据自身的多方面特征，如数据模式信息、数据源信息、数据质量信息等，而这些是传统 Web 服务所不具备的。第二是要即时响应数据源的自治变化，动态维护数据视图和数据源之间的一致性。解决这一问题有两种常见的方法，即基于事件的方法和基于轮询的方法。基于事件的方法由数据源主动地向使用者发送更新事件，事件中包含了数据变化的内容和时间等重要信息。然而，这种方式通常是侵入式的、紧耦合的，数据源需要事先知道所有使用者的信息，并且需要一个统一的事件模型和事件语义。基于轮询的方法则根据一定的时间间隔由数据源的使用者向数据源主动发起访问请求，由使用者自主检测和发现数据源的变化，从而实现数据的更新。它的优点是非侵入式的、松耦合的，而且实现较为简单。

在初期研究工作中，iViewer 首先采用了轮询的方法。实现轮询方法的一个难点在于如何保障视图更新的性能。频繁地访问数据源以及时获得正确的和新鲜的数据会造成数据源的瓶颈，同时重复数据的多次传输也会带来网路带宽的浪费。针对这一问题，采用了如下基本技巧：尽管数据源无法向其使用者提供实时的数据更新情况，但是为了提供更好的数据使用效果，它可以将自身的数据更新特征连同其数据本身一同对外发布，由数据的使用者根据此特征来决定数据更新的策略。典型的更新特征包括更新频率、更新方式、更新数据量等，使用者可以根据这些特征获得接近最新鲜的数据，同时保障数据获取的整体性能。

图 1 展示了应用 iViewer 方法构建数据视图的基本原理和实现过程，分为三个主要步骤：

(1) 数据服务化

为了实现互联网上异构数据源间的互操作能力，需要将它们封装为具有统一的描述模式和访问方式的数据服务，这一过程称为服务化。数据库和网

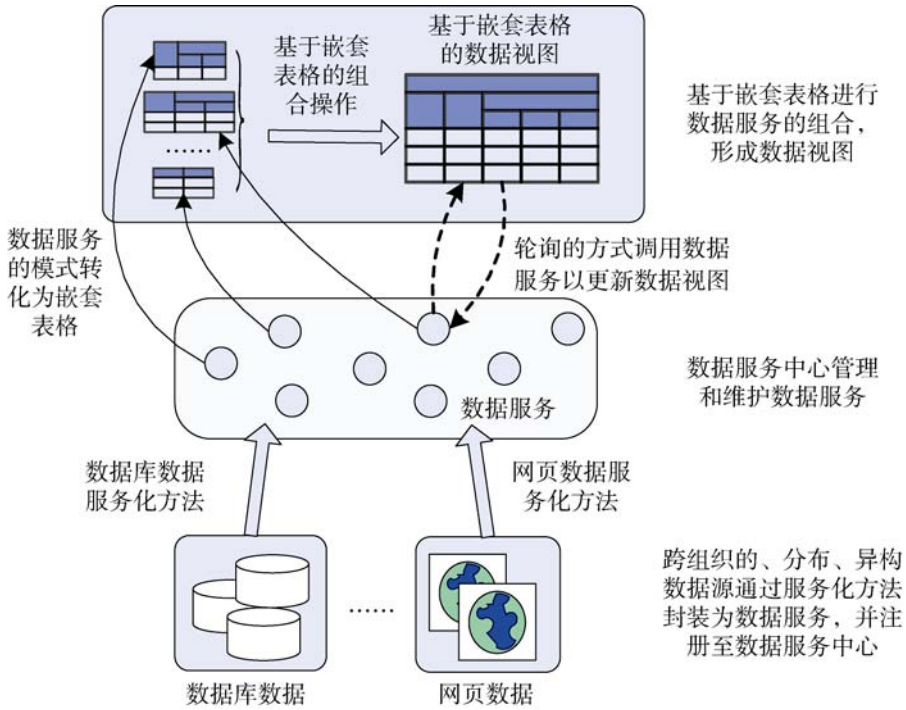

Fig.1 An overview of iViewer approach

图 **1** **iViewer** 方法的基本原理

页是互联网上最常见的数据类型, iViewer 方法首先提供了对这两类数据进行服务化的基本方法和具体实现。当前的数据服务化方法大都基于封装器技术，通过该技术可以实现对数据库、Web 等信息源的数据自动抽取，并转化为结构化数据，从而能够发布为具有统一数据结构、格式和访问方式的数据服务。与传统服务化技术不同的是, iViewer 方法在服务化过程中尝试对数据源自身进行建模，刻画数据模型、数据质量、数据更新频率和方式等重要信息，并据此获得更好的数据视图构建和维护效果。通过服务化过程而生成的数据服务，将被统一注册到数据服务中心进行有序化的组织和管理。

(2) 数据服务组合

数据服务模型为动态、灵活地构建数据视图提供了有效的支持，用户通过对数据服务进行不同形式的组合操作，可形成满足不同业务需求的数据视图。iViewer 方法采用嵌套表结构来为数据服务组合过程提供可视化的服务组合算子。嵌套表格相对于传统方式能够更直观地表达数据，适用于描述复杂对象，在其上可实现“直接操纵”的所见即所得的数据组合操作方式，即操作结果可在嵌套表格上直接呈现，并且用户可在可视结果上直接进行下一步组合操作。嵌套数据模型具有坚实的理论基础，在处理组合操作上可映射至嵌套关系模型的查询，因此其查询优化方法可为数据服务的组合优化提供良好的支持。

(3) 数据视图更新

为了提供能够适应互联网环境的数据视图更新方式，保障数据服务的自治性，采取了基于轮询的数据视图更新方法。然而由于无法预测数据源更新的频率，轮询方法可能会造成对数据源或网络等的性能瓶颈。本文的视图更新方法充分考虑了数据服务的数据更新特征，如更新频率、更新方式等，以应对上述瓶颈，并以此作为视图更新的依据。

针对上面的三个基本步骤，已经在前期工作中尝试了网页等互联网数据的服务化方法[4]，因此，在本文的后续章节中，将重点阐述数据服务组合和数据视图即时性保障的基本方法和技巧。

3 基于数据服务组合的可视化数据视图生成

3.1 数据服务的定义

分布在互联网上的数据源大多是自治的、动态

演化的，其描述信息大多是事先未知的，传统的数据集成方法缺乏一个统一的数据模型和查询方法来即时、动态地获取业务数据，构建数据视图，并能应对数据源的自治变化。为了弥补这一缺陷，人们开始尝试引入服务的思想，以服务的形式对数据加以封装和提供，并将其作为数据集成的基本元素[5]，以应对数据源的未知性和动态性。数据服务化有利于对后端数据源的数据结构及其元数据信息进行有效管理，便于数据源的动态发现，有效降低集成过程中数据源变更所带来的维护代价。Web 服务是当前常见的一种数据服务形态[5]。而且随着服务计算的不断发展，也出现了基于 XML(extensible markup language)和 JSON(JavaScript object notation)等自由格式的 REST(representational state transfer)服务。需要指出的是，Web 服务并不易于业务用户直接用来操控数据，这里存在两个问题：首先，Web 服务是为开发人员设计的，使用 Web 服务必须要了解相关的技术细节。其次，Web 服务在建模时并没有考虑服务运行时所依赖的业务数据的模式等反映数据特征的关键信息，用户仅能通过服务的输入输出消息来观察业务数据的局部变化。针对 Web 服务的缺点，已有研究工作[6]指出数据服务应包含数据质量、数据能力、服务质量等多方面的特征描述。iViewer 方法借鉴文献[6]的工作，基于中立的数据结构给出一种新的能够反映业务数据模式和特征的数据服务，并且基于数据服务能够方便业务用户构建数据视图。数据服务的定义如下：

定义 1 (数据服务, data service) 数据服务是一个四元组 *ds*:=<*uri*,*dataSchema*,*operations*={*operation*}, *metadatas*={*metadata*}>，其中：

uri 是数据服务的唯一标识；

dataSchema 为数据服务所封装的业务数据的数据模式，采用良好定义的 XML Schema 来描述。

operations 是该数据服务上支持的数据操作集合。一个数据操作可被定义为一个三元组，*operation*={*name*,*inputs*,*outputs*}，*inputs* 为输入参数的集合，*outputs* 为输出参数的集合。操作名以及所有的输入输出参数均为数据模式片段。

metadata 是数据服务的元数据，*metadatas* 是元数据的集合。元数据 *metadata*=<*name*,*value*>, *name* 是该元数据的名称, *value* 是数据服务对应的该元数据的值。

XML 的标准性、封装性、简洁性，以及较强的表达能力使其成了在 Web 间表达和交换数据的实际标准，因此本文用以表征数据服务的数据模式用 XML Schema 来表示。数据服务的操作是获取数据内容的手段，它提供了对数据服务所封装的全部或者部分数据的获取方法，其输入输出参数可以映射至数据模式片段。数据服务的元数据包括诸如数据质量(实时性、一致性等)、服务质量(可用性、响应时间等)、数据生命周期(更新频率、更新方式、数据量等)以及领域分类、地理位置等信息。图 2 为一个消防部门设备信息的数据服务实例。

3.2　数据视图的定义

数据视图的生成过程实际是将数据服务进行组合，并将组合结果以可视化方式向用户呈现的过程。传统数据服务的组合方式大多基于消息，建立服务间的参数映射构建组合流程，这种方式对于用户来说是复杂的。为此，本文借鉴了嵌套关系模式[7]和嵌套表格[8]的研究工作，将数据服务与嵌套表格进行一一关联，并通过开发嵌套表格之间的可视化组合操作来帮助用户直观、简单地组合数据服务，构建新的数据视图。嵌套关系模型是一种简单、直观、具有较强表达能力的数据模型，能够对半结构化和结构化数据提供很好的表示[8]，更容易表达现实世界中的复杂数据对象。文献[7]中将嵌套关系模型定义为：

定义 2 (嵌套关系模型, nested relational model) 令 A 为全局的属性名称和关系模型名称集合，则嵌套关系模型 $R(S)$可定义为 $R(S)=(A_1,A_2,\ldots,A_m)$。其中 $R\in A$ 是关系模型的名称, S 是关系的属性列表, A_i 是原子属性或者其子关系的嵌套关系模型。如果 A_i 是形如 $R_i(S_i)$的关系模式，则该模型的名称 R_i 被称为 R 的子关系属性。

用嵌套关系模型描述的数据很容易被呈现为嵌套表格；嵌套表格由表头、行和列组成。表头描述了嵌套关系的模式信息，表头中的每一行描述了一个嵌套关系；嵌套关系中的属性名显示为列名，顶层的列名为嵌套关系名称，子关系属性对应的列被

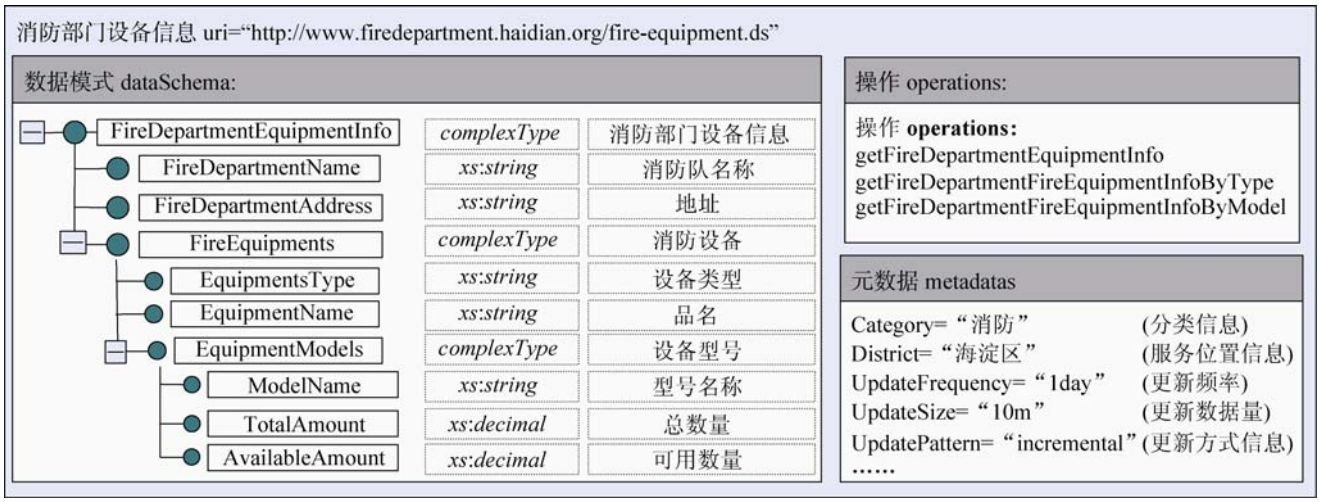

Fig.2 Data service for equipment information in a fire department

图 **2** 消防部门设备信息的数据服务

称为复合列，原子属性对应的列被称为原子列；嵌套关系的实例对应嵌套表格的实例，被显示为行。嵌套表格是一种易于用户使用的人机交互方式，很适合用来表示商业表格。图 3 给出了一个嵌套表格的示例。

图 3 中的阴影部分为嵌套表格的表头，反映了嵌套关系的数据模式，非阴影部分为嵌套表格的实例。该嵌套关系“消防部门设备信息”定义了不同消防部门的设备信息，它包括两个原子属性(“消防队名称”和“地址”)以及一个名为“消防设备”的子关系。该“消防设备”子关系描述了一个消防部门内的设备信息，它又包含两个原子属性(“设备类型”和“品名”)和一个名为“设备型号”的子关系。该“设备型号”子关系描述某个设备的具体型号和数量信息。

3.3 数据服务的可视化组合

数据服务利用基于嵌套表格的可视化组合方式，实现数据模式之间的组合，并生成最终的数据视图。

消防部门设备信息							
消防队名称	地址	消防设备					
		设备类型	品名	设备型号			
				型号名称	总数量	可用数量	
海淀区消防支队	海淀区翠微路 17 号	泡沫灭火器	PC 系列泡沫产生器	L5541956	100	90	
				L5541957	200	140	
			PH 型环泵式泡沫比例混合器	L5541952	70	70	
				L5541953	90	80	
		……	……	……	……	……	
朝阳区消防支队	朝阳区道家园一号	泡沫灭火器	PC 系列泡沫产生器	L5541956	120	90	
				L5541957	250	140	
			PH 型环泵式泡沫比例混合器	L5541952	60	50	
				L5541953	100	90	
		……	……	……	……	……	

嵌套表格数据模式

嵌套表格数据实例

Fig.3 A nested table example

图 **3** 嵌套表格示例

主要包括如下两个步骤：

(1) 将数据服务中利用 XML Schema 定义的数据模式转化为嵌套关系模式，并呈现为嵌套表格。

(2) 利用嵌套表格之上的一系列组合操作算子进行嵌套表格的组合，生成数据视图。

上述由 XML 模式向嵌套表格的转化过程主要实现了根据XML的标签的层次关系转化为嵌套关系的关系名、原子属性名、子关系属性名。文献[9–10]阐述了由 XML 向嵌套关系模型的转换方法，两者在大部分情况下可实现相互转化。在前期研究工作中[8]，已经给出了在嵌套表格之上的所见即所得的数据操纵算子集合，能够帮助用户方便地访问、加工和转换互联网数据。这些操作能够利用嵌套表格的直观性向用户提供简便的组合操作，并且能够覆盖主要的 SQL(structured query language)算子，包括单表的选择、投影、排序、分组聚集、函数计算等能力，也支持多表的合并和链接，提供了较强的数据检索和处理能力。主要的操作算子及其功能描述如表 1 所示。上述操作所形成的数据视图的组合逻辑保存在脚本文件中，用以持久化视图逻辑，也是视图更新时数据计算的依据。

在前期工作的基础上，本文借鉴了语义集成的研究工作[11]，对表 1 中的部分组合操作算子(Merge, Fuse,LinkService)进行了改进，将数据服务中心的全局词汇表作为判断嵌套表属性间语义相似关系的依据，并以此判断嵌套表的相似度，进行自动化的表格组合。

之前工作中的Merge/Fuse操作需要人工验证数

Table 1　Composition operators for nested tables

表 **1**　嵌套表格组合操作算子

操作类型	操作算子	功能描述	对应的嵌套关系操作
嵌套表格创建	*Import*(*ds*)	通过导入一个数据服务 *ds* 创建该服务对应的嵌套表格	*New*
	CreateSheet(*col*)	从当前的嵌套表格中拷贝一列 *col*，创建一个独立的嵌套表格，该列可以为原子属性或子关系属性	
数据内容处理	*Filter*(*col*,*condition*)	根据条件表达式 *condition* 对嵌套表格中某一列 *col* 的数据进行过滤，该列可以为原子或子关系属性	*Select*(σ)
	Sort(*col*,*order*)	对嵌套表格中的数据以某列 *col* 的值为条件进行排序，*order* 为升序或降序，该列仅为原子属性	
	Header(*count*)	截取嵌套表格开头的一部分数据(数量为 *count*)以删除不需要的数据	
	Tail(*count*)	截取嵌套表格结尾的一部分数据(数量为 *count*)以删除不需要的数据	
数据模式处理	*DeleteColumn*(*col*)	将嵌套表格中的某列 *col* 删除	*Delete*
	RenameColumn(*col*,*name*)	将嵌套表格中的某列 *col* 重命名为 *name*	*Update*
	Nest/*Unnest*(*col*)	将普通的表格转换为嵌套表格/将嵌套表格转换为普通表格	*Nest*(η)/*Unnest*(μ)
数据清洗	*AddFunction*(*col*,*function*)	对嵌套表格中的一个或多个列进行算术或字符运算，并将其结果作为新增列, *col* 为原子属性	*Insert*
	MergeInstance(*col*)	删除某列 *col* 上数值重复的数据，实际上实现了对该列的分组	*Select*(σ)
嵌套表格组合	*Merge*(*a*,*b*,<*a.X*,*b.Y*>...)	将两个嵌套表的数据合并到一个嵌套表中，等价于定义在嵌套关系数据模型之上的递归 Union 操作, *a*、*b* 均为子关系, *X*、*Y* 为 *a* 和 *b* 的原子属性，是进行合并的两列	*Union*($\cup$)
	Fuse(*a*,*b*,< *a.X*,*b.Y*>...)	同 Merge 操作类似，将两个嵌套表的数据合并到一个嵌套表中，但同时删除列值重复的数据	
	LinkService(*ds*,*mapping*)	对存在参数关联关系的多个嵌套表格的数据进行合并操作，*mapping* 为实现嵌套表格关联的两服务的列的映射	*Join*()

据服务之间是否表达同构数据，且在合并过程中是通过表头名的完全匹配实现的。而改进后的 Merge/Fuse 操作，可根据嵌套表格属性的语义关系进行匹配，并计算表格间的相似度，并能在相似度满足一定条件的情况下实现服务自动化合并。因此将 Merge 操作重新定义为 *Merge*(*ds*)，并为此定义嵌套关系 r_1 与 r_2 的相似度为：

$$similarity(r_1,r_2)=\frac{2\times matchedAtomAttr(r_1,r_2)}{\#r_1.recurAtomAttr+\#r_2.recurAtomAt} \quad (1)$$

其中，$r_1.recurAtomAttr$ 表示嵌套关系 r_1 的所有属性(原子属性和子关系属性)所递归包含的原子属性。$\#r_1.recurAtomAttr$ 表示其数目，即若 r_1 只包含原子属性，则$\#r_1.recurAtomAttr$ 为 r_1 的所有原子属性的数目，若 r_1 还包含子关系，则$\#r_1.recurAtomAttr$ 为其原子属性数目与其所有子关系的 *recurAtomAttr* 的数目之和，这是一种递归的定义方式。*MatchedAtomAttr*(r_1,r_2)表示 $r_1.recurAtomAttr$ 与 $r_2.recurAtom$ 中实现语义匹配的属性个数，即两个嵌套表格中出现的所有原子属性的匹配数目，这样计算的原因在于有的嵌套表格中将若干原子属性合并为一个子关系，而有的嵌套表格没有。对于两个原子属性 $attr_1$ 和 $attr_2$，通过函数 *match*($attr_1$,$attr_2$)来判断是否两个属性实现语义匹配，*match*($attr_1$,$attr_2$)为真的条件为 $attr_1$ 的语义与 $attr_2$ 的语义相等或为同义词，或者 $attr_1$ 的语义包含 $attr_2$ 的语义，或 $attr_2$ 的语义包含 $attr_1$ 的语义。

之前定义的 LinkService 操作需要人工指定服务间进行关联的列，而根据上述定义的 *match*($attr_1$, $attr_2$)，可使得嵌套表格能够在两个匹配的属性上实现自动关联。因此将 LinkService 操作重新定义为 *LinkService*(*ds*)。

4 数据视图的即时更新

轮询方法的基本思想是设定时间间隔，并通过不断地访问数据源以获得最新的数据。其好处在于非侵入性、松耦合以及对提供者的要求较少，因此本文尝试用轮询的方法来解决互联网环境下数据视图的更新问题。然而由于数据源更新的不可预测性，该方法在数据源或网络传输等方面会产生性能瓶颈。为了应对这种性能瓶颈，可以根据数据源的特征信息来预测并制定数据服务的访问策略。这些特征信息反映了数据源在什么时间、以何种方式、更新多少数据量的大致情况。在数据服务的注册过程中，数据源可以将自身的数据更新特征也随之发布至数据服务中心，并以数据服务的元数据形式进行存储。数据视图构造者可以利用这些特征来预测并制定数据视图的更新策略。详细的特征描述如表 2 所示。

表 2 所示的数据服务的更新特征是数据中心制定视图的即时更新策略的依据。该策略是上述多个特征综合作用的结果，在实际应用中应当灵活组合以应对不同情况。其中数据服务的更新时间和更新频率是制定视图更新策略的基本依据，在此基础上根据应用的实际需求进行策略调整。在一个系统带宽受限的环境下，应当适当减少大数据量服务的更新频率，以保证其他服务能够得到即时更新。在对某些数据的实时性要求较高的系统中，应当适当增加某些数据服务的更新优先级，以保证能够即时获得重要的数据。此外，根据数据服务的不同的更新方式可采用不同的更新方法。例如若更新方式为附加式，则只需获取所有新增数据并附加在当前数据视图中即可；若更新方式为覆写式，则可删除视图中旧的数据，再获取新增数据；对于随意式的更新方式，则需要获得所有数据，重新计算数据视图。当然这些更新方式的实现依赖于数据服务对上述数据访问方式的支持。

在上述的更新策略下，数据视图的更新方法和过程如下：对于构成数据视图的每一个数据服务，根据该服务的更新频率和更新时间预测下一次应当对该服务调用的时间。当该时刻到达时，调用该数据服务，利用数据服务的返回结果更新数据视图。若此时刻需要对多个数据服务进行更新，则根据数据更新策略决定多个服务的调用顺序，依次调用并更新数据视图。

本文在一个实际的示范应用系统中，根据上述数据服务特征和视图更新过程，制定了如下所示的更新算法：

Table 2　Data updating characteristics for data services

表 2　数据更新特征描述

特征名称	特征描述	量化方式
更新频率 updateFrequency	该特征表明服务所封装的数据源的更新时间间隔。数据视图的更新按照组成该视图的各个服务的更新频率来进行。并非所有的数据服务都有固定的更新频率，此时该特征反映了一种大致或者能够满足应用需求的更新频率。此外更新频率应当根据实际应用场景的需求动态变化，例如某些突发事件可能导致更新频率的增加	该特征由数据提供者注册数据服务时提供，可在数据视图构造时根据视图的业务语义对服务更新频率的要求对其进行修正。该特征按时间段计量(如 s)，可以为固定的数值，可以为基于时间段或实际应用场景的更新频率函数
更新时间 updateTime	该特征表明服务所封装的数据源是否有特定的更新时间，该特征定义的依据在于有相当的数据源选择在固定时间集中更新数据	该特征由数据提供者发布数据服务时提供，按时间点计量(如每天的某一时刻)
更新数据量 size	该特征反映了调用数据服务时所传输的数据量	该特征由数据提供者注册数据服务时提供，也可结合多次服务调用过程中所获得的实际的数据量来预测下一次调用的数据量，数据量按数据占用的存储空间计量(如 Kb)
支持的更新方式 updatePattern	数据服务支持的更新方式表明了数据视图在调用数据服务获得数据后，能够以何种方式进行视图更新，典型的有： (1) 附加式——数据以附加的形式更新，历史数据不发生改变 (2) 覆写式——更新时先删除原来的全部或部分数据，然后添加新的数据 (3) 随意式——数据更新可能伴随增加、删除、修改等	该特征由数据提供者注册数据服务时提供，以文本的方式标注服务的更新方式
更新优先级 priority	该特征实际反映了不同的数据服务的实时性需求，优先级越高，表明该数据服务在视图中越重要，视图更新时应当优先保证该服务的更新	该特征可在数据视图组合生成过程中由视图构造者赋予各个服务不同的业务优先级语义。可对数据服务进行优先级标注，数值越大的优先级越高

Function view.update　/*视图更新函数*/

输入：数据视图 view 以及组成该视图的数据服务集合 $set(ds)=ds_1,ds_2,\ldots,ds_n$

while (! view.destroyed) {　/*只要视图没有被销毁(此处的销毁是指对用户的可视视图的销毁)，则不需要再更新可视化的数据视图，而该视图的组合逻辑可能被持久化保存以便重用*/

if ($set'(ds) \subset set(ds)$ needs to be updated) {　/*需要调用 $set(ds)$的某个子集中的数据服务更新视图*/

List(*ds*)=calculateAndRankUpdateIndex(*set'*(*ds*)) /*计算服务的更新系数并且按照从大到小的顺序排列*/

for each *ds* in *List*(*ds*)　/*对于列表中的每个数据服务*/

ds.update();　/*调用其更新函数*/

Function ds.update　/*数据服务 *ds* 的更新函数*/

if (*ds.updatePattern* == appending)　/*若服务的更新方式为附加式更新*/

ds.getNewData();　/*获取数据服务的新增数据*/

else if (*ds.updatePattern*==overwriting)　/*若服务的更新方式为覆写式更新*/

view.deleteOldData();　/*则从视图中删除旧数据*/

ds.getNewData();　/*获取数据服务的新增数据*/

else　/*若服务的更新方式为随意式更新*/

ds.getAllData();　/*需要获取数据服务的全部数据*/

computeAndUpdateView()　/*根据视图的组合逻辑对视图进行重新计算和更新*/

ds.wait(*updatePeriod*)　/*等待该服务的更新时间间隔至下一次调用时间*/

在需要更新多个数据服务时，设计了如下的更新策略。假设服务$ds_1,ds_2,\ldots,ds_n$，它们各自拥有属性 *updateFrequency*, *priority*, *size*, *updatePattern*, *last-UpdateTime*，分别表示服务的更新频率、更新优先级、更新数据量、更新方式、上次调用时间，则服务的更新系数定义为：

$$ds.updateIndex=\frac{(a\times ds.priority^{g}+b)}{(c\times ds.size^{h}+d)}\times (e\times(\frac{t-ds.lastUpdateTime}{ds.updateFrequency})^{i}+f),a,c,g,h,i>0 \quad (2)$$

式(2)中，a~f 均为对各特征进行单位归一的系数，t 为当前时间。为了使式中的变量的相互作用关系合理，应当恰当地设计变量 a~f。式(2)表达了服务 ds 的更新系数与其更新优先级为递增关系，与其更新的数据量呈递减关系，且当前时间距离上次调用时间的间隔与更新的频率比值越大，表明距上次调用该服务的间隔越久，更新系数越大。视图中各个数据服务的更新系数的作用在于在视图更新时，若存在多个服务需要调用，更新系数越大的服务，其更新需求越迫切，越应当优先处理。

5 系统实现与案例分析

5.1 场景描述

为了应对火灾等突发事件，尤其是大型火灾，应急部门应当从多个消防、医疗部门获得实时的火灾救援设备和人员的数据，形成统一的数据视图，以便能够即时地从各部门调配资源，处理突发事件。下面以在某市火灾应急处置过程中，火灾救援设备即时共享的示范应用为例来验证 iViewer 方法的实际应用效果。

在火灾突发事件的处理中，最重要的措施包括灭火和烧伤人员的救治，因此涉及的主要部门包括消防部门和医疗部门。在实际中，上述两类部门主要按照行政区域进行划分，即包括海淀区消防支队、朝阳区消防支队以及海淀区卫生局、朝阳区卫生局等部门。各个部门维护其本地的消防及医疗资源等信息。消防部门维护的信息包括各种消防设备、消防车辆、消防队员的可用性信息；医疗部门维护的信息包括各种医疗设备以及救护车、医务人员的可用性信息。上述各种资源在各部门本地以不同厂商的数据库、文件或 Web 信息系统的方式存储和维护，这些数据为分布的、异构的。为了实现这些资源的共享，需要对它们进行服务化。

火灾发生后，为了使决策者能够根据各个部门的资源的实际情况，及时地从火灾地点邻近的消防和医疗部门中调配资源以完成灭火任务，应当向决策者提供各部门资源的全局数据视图。而在传统方法中用户需要编写复杂查询语句，且很难通过一次查询获得所需的所有结果，并呈现为全局视图。同时在火灾处置过程中，消防、医疗设备，人员等资源是在不断消耗、不断变化的，为决策者提供的数据视图应该能够即时反映这种变化。而传统数据查询方法提供的查询结果是静态的，用户必须重新发出查询请求以获得最新数据。利用本文的数据视图生成和更新方法，可以满足上述对全局数据视图呈现和即时更新的需求。

5.2 iViewer 方法的实现与应用过程

(1) 服务化

为了实现对各个消防、医疗部门的分布、异构资源进行共享，需要对其进行服务化操作。服务化过程抽取了各个部门所需共享的资源的数据模式，形成数据服务的数据模式，并且提供了若干获取数据实例的操作，服务的执行能够获得各个部门的最新资源信息。各部门将数据封装后形成的数据服务发布到互联网上，并注册至数据服务中心。各个部门的数据服务如表 3 所示。

(2) 数据视图生成

本案例所要展示的全局数据视图的构建目标是按照行政区域建立多个区域的消防、医疗设备的可用数量全局视图。该视图的构建过程借助了可视化嵌套表格操作。数据服务的数据模式转化成的嵌套表格如图 4 所示。

图 5 所示为在嵌套表格上进行组合操作，并生成数据视图的过程。其中步骤 1 和步骤 3 通过增加一个常数列分别标识设备的资源大类，即“消防”和“医疗”；步骤 2 和步骤 4 分别约束了设备的内容：消防设备应当为“泡沫”和“干粉”，而医疗设备则包含“担架”和“氧气罐”；步骤 5 合并了海淀区的消防和医疗设备；步骤 6 分别计算了各个设备类型的可用设备总量；步骤 7 和步骤 8 分别筛选了当前可用(即为“工作”状态)的消防和医疗人员信息；步骤 9 和步骤 10 分别标识了人员的资源大类，即“消防”和“医疗”；步骤 11 合并了海淀区的消防和医疗人员；步骤 12 分别计算了消防和医疗的可用人员总量；

Table 3　Data services corresponding to the distributed, heterogeneous data in different departments
表 **3**　各部门分布、异构数据进行服务化后形成的数据服务

数据服务名称	数据服务定义	定义解释	备注
海淀区消防支队消防设备数据服务	ds_1=<uri="FireDepartEquipInfoDS",dataSchema="FireDepartEquipInfo=<DepartName,District,FireEquips=Array<EquipType,EquipName,EquipModels=Array<ModelName,TotalAmount,AvailableAmount>>>", operations={<getFireDepartEquipInfo,inputs=ϕ,outputs={FireDepartEquipInfo}>	FireDepartEquipInfo：消防部门设备信息; DepartName：部门名称; District：行政区域; FireEquips：消防设备; EquipType：设备类型; EquipName：设备名称; EquipModels：设备型号; ModelName：型号名称; TotalAmount：总数量; AvailableAmount：可用数量	其他行政区域的消防部门提供类似服务，设朝阳区消防支队的消防设备信息数据服务为 ds_5，西城区的为 ds_6
海淀区消防支队消防员数据服务	ds_2=<uri="FireDepartFiremenInfoDS",dataSchema="FireDepartFiremenInfo=<DepartName,District,Firemen=Array<Name,Telephone,CurrentStatus>>",operations={<getFireDepartFiremenInfo,inputs=ϕ,outputs={FireDepartFiremenInfo}>	FireDepartFiremenInfo：消防部门消防员信息; Firemen：消防员; Name：消防员姓名; Telephone：消防员电话; CurrentStatus：消防员当前状态，表示“任务中”、“休假”或“工作”等状态; DepartName、District 的含义与其在 ds_1 中的相同	其他行政区域的消防部门提供类似服务，设朝阳区消防支队的消防员信息数据服务为 ds_7，西城区的为 ds_8
海淀区卫生局医疗设备数据服务	ds_3=<uri="MedicalDepartEquipInfoDS",dataSchema="MedaicalDepartEquipInfo=<DepartName,District,MedicalEquips=Array<EquipType,EquipName,EquipModels=Array<ModelName,TotalAmount,AvailableAmount>>>",operations={<getMedicalDepartEquipInfo,inputs=ϕ,outputs={MedicalDepartEquipInfo}>	MedicalDepartEquipInfo：医疗部门设备信息; MedicalEquips：医疗设备; DepartName、District、EquipType：设备类型; EquipName、EquipModels、ModelName、TotalAmount、AvailableAmount 的含义与其在 ds_1 中的相同	其他行政区域的医疗部门提供类似服务，设朝阳区卫生局的医疗设备信息数据服务为 ds_9，西城区的为 ds_{10}
海淀区卫生局医务人员数据服务	ds_4=<uri="MedicalDepartStaffInfoDS",dataSchema="MedicalDepartStaffInfo=<DepartName,District,MedicalStaff=Array<Name,Telephone,CurrentStatus,ChargeCase>>",operations={<getMedicalDepartStaffInfo,inputs=ϕ,outputs={MedicalDepartStaffInfo}>	MedicalDepartStaffInfo 医疗部门医务人员信息; MedicalStaff：医务人员; ChargeCase 表示该医务人员主治的疾病类型; DepartName、District、Name、Telephone、CurrentStatus 的含义与其在 ds_2 中的相同	其他行政区的医疗部门提供类似服务，设朝阳区卫生局的医务人员信息数据服务为 ds_{11}，西城区的为 ds_{12}

FireDepartEquipInfo						
Depart Name	District	FireEquips				
		Equip Type	Equip Name	EquipModels		
				Model Name	Total Amount	Available Amount

FireDepartFiremenInfo				
Depart Name	District	Firemen		
		Name	Telephone	CurrentStatus

MedicalDepartEquipInfo						
Depart Name	District	MedicalEquips				
		Equip Type	Equip Name	EquipModels		
				Model Name	Total Amount	Available Amount

MedicalDepartStaffInfo					
Depart Name	District	MedicalStaff			
		Name	Telephone	Current Status	Charge Case

Fig.4　Nested tables corresponding to data services
图 **4**　数据服务转化成的嵌套表格

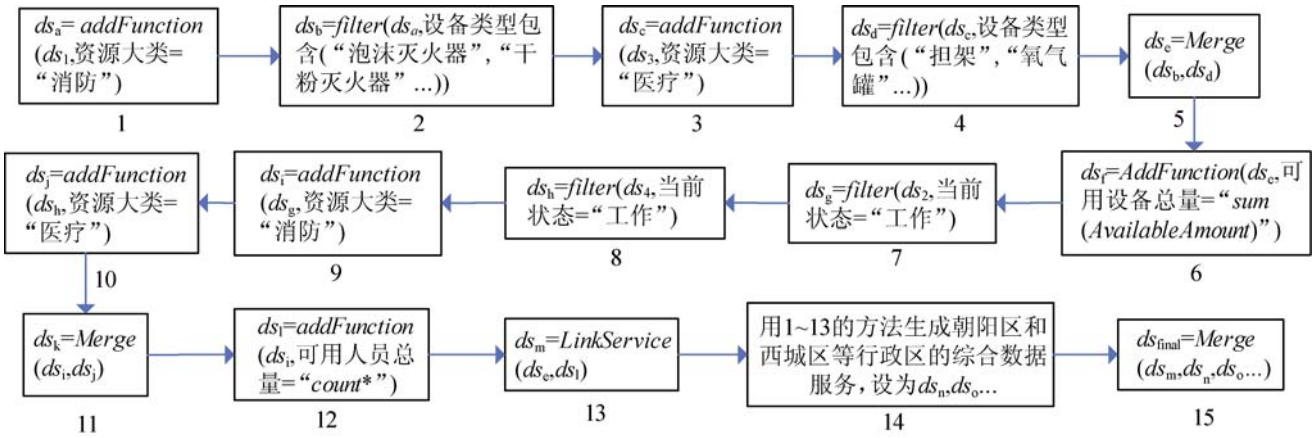

Fig.5 The data view construction process through data service composition operations

图 **5** 对数据服务进行组合操作形成数据视图的过程

步骤13将设备信息和人员信息按照行政区和资源大类的匹配进行关联；步骤13计算了其他行政区的消防、医疗设备和人员的总体信息；步骤 14 将各个区的总体信息进行了合并，形成了最终的数据视图，如图 6 所示。

(3) 数据视图的即时更新

在上述生成的数据视图上，基于第 4 章所示的轮询方法，模拟了数据视图的即时更新效果。在此场景中，假设各个消防和医疗部门每个一定的时间间隔对数据源内资源的实时信息更新一次。同时在火灾处理的实际场景中，应急中心会尽量调配距离火灾发生地点较近的消防部门和医疗部门的资源。因此应当为火灾发生地点所在行政区的消防和医疗部门所提供的数据服务设定较高的更新优先级，以便能够及时地更新当地消防和医疗部门的数据。各个数据源内的数据更新频率、更新数据量不同，且各个数据源在视图中的优先级不同，因此数据服务中心利用每个数据服务的特征制定数据视图的

救火资源信息												
行政区	资源大类	设备信息						可用人员总量	人员信息			
		设备类型	可用设备总量	设备名称	设备型号				姓名	电话	当前状态	主治疾病
					型号名称	总数量	可用数量					
海淀区	消防	泡沫灭火器	380	PC系列泡沫产生器	L5541956	100	90	可用消防员总量	张明	xxx	工作	Null
					L5541957	200	140					
				PH型环泵式泡沫比例混合器	L5541952	70	70		王立	xxx	工作	Null
					L5541953	90	80		刘川	xxx	工作	Null
		干粉灭火器	140	硝酸铵盐干粉灭火器	MFL9	60	60		陈曦	xxx	工作	Null
					MFL8	80	80					
	医疗	担架	350	救护车担架	YDC-1A1	300	200	可用医务人员总量	郭峰	xxx	工作	烧伤
				折叠式担架	YDC-2A	200	150					
		氧气罐	800	氧气罐	YQG-2G	1 000	800		……	……	……	……
朝阳区	消防	泡沫	……	……	……	……	……	可用消防员总量	……	……	……	……
		干粉	……	……	……	……	……					
	医疗	担架	……	……	……	……	……	可用医务人员总量	……	……	……	……
		氧气罐	……	……	……	……	……					

Fig.6 The generated data view after composition operations

图 **6** 组合操作后形成的数据视图

更新策略，保证更新系数高的数据服务能够优先更新，从而使得更新鲜(更新频率高)、更重要(更新优先级高)、长时间等待的数据源能够优先被访问，保证视图的时新性。在实验中，选择了在应用运行过程中某个恰好多个服务需要更新的时间点，获得视图中各个数据服务的各项数据特征。同时假设 $m=(t-ds.lastUpdateTime)/ds.frequency$，$t$ 为当前时间。令 $a=1$，$b=0$，$c=(1/10)^{0.3}$，$d=0$，$e=1/m$，$f=m/(3\times priority)$，$g=1$，$i=0.3$，$h=1$ 为单位归一化的系数，则在此场景下式(2)变为：

$$ds.updateIndex=\frac{ds.priority+\frac{t-ds.lastUpdateTime}{ds.frequency\times 3}}{(ds.size/10)^{0.3}} \tag{3}$$

据式(3)算出的各服务的更新系数如表 4 所示。

Table 4　The updating coefficients for multiple data services at some specific moment

表 **4**　某一时刻多个待更新数据服务的更新系数

数据服务	数据量/Kb	更新频率/min	上次调用距当前的间隔时间/min	优先级	更新系数
ds_1	300	5	5.03	10	7.43
ds_2	600	5	5.14	10	6.04
ds_3	400	7	7.26	10	6.82
ds_4	700	7	7.18	10	5.76
ds_5	300	5	6.69	9	6.79
ds_6	600	5	7.10	9	5.53
ds_7	400	7	8.24	9	6.19
ds_8	700	7	8.97	9	5.26
ds_9	300	5	8.98	9	6.91
ds_{10}	600	5	8.74	9	5.60
ds_{11}	400	7	10.93	9	6.28
ds_{12}	700	7	11.13	9	5.31

由表 4 可知，数据服务 ds_1~ds_4 位于火灾发生地所在的行政区，因此优先级最高，更新系数与同类的优先级较低的服务相比也较高($ds_1.updateIndex>ds_5.updateIndex$)。其中 ds_2 和 ds_4 由于更新的数据量较大，其更新系数较 ds_1 和 ds_3 低；ds_9~ds_{12} 虽然其优先级较低，但是由于某些原因这些服务较长时间没有调用，其更新系数与同类的相等优先级的数据服务相比也更高($ds_9.updateIndex>ds_5.updateIndex$)。

在该示范应用中，基于第 4 章的视图更新算法和上述的更新系数计算方法，验证了视图更新方法的效果。根据表 4，当前需要更新的数据服务包括 ds_1~ds_{12}，且对各个服务按照更新系数由高到低的顺序进行调用，调用数据服务并返回数据平均需要大约 6 s 的时间，且每次调用结果包含新数据，能够反映为视图的更新。由于多个数据服务的顺次调用，使得数据视图在每个数据服务调用完成后均重新计算视图数据，更新了视图，这实际上完成的是单个数据服务更新引起的视图更新，平均花费 7.3 s，完成上述 ds_1~ds_{12} 数据服务的调用和视图的更新，总共花费 87 s。在更新过程中由于 ds_1 更新系数最大，其更新结果最先呈现在视图上。

6　相关工作

借鉴传统的 Web 服务的思想，越来越多的研究者和工业界开始利用服务的形式来提供数据，标准化、松耦合均是数据服务从 Web 服务处沿袭的优势特征，为异构数据源的集成提供了解决思路。直接基于 Web 服务来构建数据集成是面向服务架构(serviceoriented architecture，SOA)出现后数据集成的主流方式之一，它提供了对多源异构数据的统一访问方式。文献[5,12]均为利用传统 Web 服务进行数据集成的工作，它们将数据源封装为一组数据访问服务的集合，利用 Web 服务的接口来获取数据。然而传统的 Web 服务并不适合于数据的封装和获取。首先，它的描述文件中包含过多技术细节，如协议、端口等，不利于用户理解数据；其次，Web 服务的描述中主要包含其操作以及操作的输入输出，目标是描述服务的功能，对于数据方面的特征涉及很少。文献[6]详述了一个提供数据的数据服务中应当包含的数据层面和服务层面的描述内容。例如一个典型的数据服务的目标是提供数据的 CRUD (create、read、update、delete，创建、读取、更新、删除)操作。数据服务应当包含数据的一些典型特征，如数据源信息、数据模式、数据质量、数据生命周期等，操作是获取数据的方式，因此也继承了传统

服务的例如服务质量(quality of service, QoS)等特征。该作者在文献[13]中分析了评估和发布数据服务的方法。在意识到数据服务与传统服务的差别后，不同的研究者尝试对数据服务给出不同的定义。文献[14]认为数据密集型服务提供对大规模数据存储的访问，并且提供对数据存储的检索、操作和分析的工具；文献[12]认为数据服务是对数据源模式的有限的调用方法的集合，调用返回数据但是不改变事物的状态。上述诸多对数据服务的认识和定义与本文中的大致一致 数据服务的核心是数据，服务中的操作或方法表达了对数据的访问方式。

Web 服务已经成为实现应用集成和数据集成的主流方式之一，在 Web 服务的大趋势下，数据集成可以通过多个数据服务的组合形成统一的数据视图来实现。文献[5]将 Web 服务进行组合生成数据视图。然而当前的主要集成方法仍然是通过 Web 服务的输入输出参数匹配的方法形成数据视图，用户需要了解服务的输入输出消息的数据格式和约束，建立输入输出小的语义映射关系，以实现数据的组合，这个过程仍然是复杂的。近年来涌现出了大量用户主导的数据组合方式——Mashup。这种方式下允许非专业用户通过组合多个数据服务的内容，以生成个性化的数据系统和视图，能够适应一些多变的、自定义的场景。文献[15]通过流图的方式定义数据服务之间的数据流转，以此来形成组合数据服务的逻辑；文献[16]为通过示例(example)进行即时数据服务组合的方法。本文数据服务组合方法基于嵌套表格，它更容易被业务人员理解和操作，且其所见即所得的操作方式能够直接向用户呈现组合操作的结果，并且允许用户直接在结果上进行进一步的组合操作。

数据视图的更新方法主要需要协调和平衡视图与数据源的一致性与实现即时更新所需的代价，由数据源主导的增量更新方式[17]是物化视图更新的主要方式，它在数据源发生更新时计算视图与数据源的差异，只更新变化部分。按照维护操作的时机来看，维护策略可分为即时更新、周期更新和延时更新三种。即时更新在数据源变化时实时发送消息至视图，是一种强一致性的保障方式，但当数据源更新频繁时会造成性能瓶颈；周期更新则异步定期更新，可对更新实现批处理，然而也可能由于积累大量更新请求而造成系统负荷过大；延迟更新将维护推迟到查询时发生，可避免无用更新，但可能造成对查询的响应速度减慢的问题。增量更新的方式在紧耦合的应用领域内适用性较强，视图可以要求数据源传播其数据变更。然而在互联网环境下，很难形成数据源主导的增量更新方式，因为这种侵入式的更新方式违背了数据源的自治性，同时要求事件的描述和类型需要有统一的模型和语义支持，在互联网环境下实现难度较大。基于轮询的方法的好处在于非侵入，对提供者的要求较少，且能够实现对数据源的按需访问。但是由于需要频繁地访问数据源，甚至每次访问都需要重新获取全部数据，很容易形成性能瓶颈。为了应对这种性能瓶颈，本文利用了数据源的数据更新特征作为制定数据服务的访问策略的依据。

7 结束语

为了应对跨组织、异构、自治、动态的数据共享需求，动态、即时的构建满足用户需求的全局数据视图，本文提出了一种基于数据服务的数据视图动态构建方法 iViewer。该方法引入数据服务的思想，将数据源暴露为数据服务，用以屏蔽数据源的异构性，同时动态提供新鲜的数据。本文将数据服务作为数据视图构建的基本元素，将数据视图定义为一组数据服务的组合模式，利用嵌套数据模型和嵌套表格上的服务组合操作算子实现了数据视图的构建。在实际应用过程中，通过基于数据服务更新特征的轮询方法触发组合的数据服务的调用，获取数据源的最新数据，从而维护数据视图的新鲜性和正确性。通过应急系统中火灾突发事件处理的案例，详细分析了利用数据服务生成数据视图的过程和方法，展示了 iViewer 方法的效果。未来的工作包括：数据服务组合操作算子的优化设计、基于事件数据视图更新算法等。

References:

[1] Lenzerini M. Data integration: a theoretical perspective[C]// Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS ’02), Madison, Wisconsin, USA, 2002. New York, NY, USA: ACM, 2002: 233–246.

[2] Halevy A, Rajaraman A, Ordille J. Data integration: the teenage years[C]//Proceedings of the 32nd International Conference on Very Large Databases (VLDB ’06), Seoul, Korea, 2006: 9–16.

[3] Gupta A, Mumick I S. Maintenance of materialized views: problems, techniques, and applications[J]. IEEE Data Engineering Bulletin: Special Issue on Materialized Views and Data Warehousing, 1995, 18(2): 3–18.

[4] Yang Shaohua, Zhang Liyong, Han Yanbo. A markup language for generating Web services out of Web-based information resources and software support thereof[J]. Journal of Computer Research and Development, 2006, 43(Supp): 11–17.

[5] Zhu Fujun, Turner M, Kotsiopoulos L, et al. Dynamic data integration using Web services[C]//Proceedings of the 2004 IEEE International Conference on Web Services (ICWS ’04), San Diego, California, USA, 2004. Washington, DC, USA: IEEE Computer Society, 2004: 262–269.

[6] Truong H L, Dustdar S. On analyzing and specifying concerns for data as a service[C]//Proceedings of the 2009 IEEE Asia-Pacific Services Computing Conference (APSCC), Singapore, 2009. Washington, DC, USA: IEEE Computer Society, 2009: 87–94.

[7] Colby L S. A recursive algebra and query optimization for nested relations[J]. ACM SIGMOD Record, 1989, 18(2): 273–283.

[8] Wang Guiling, Yang Shaohua, Han Yanbo. Mashroom: end-user mashup programming using nested tables[C]// Proceedings of the 18th International Conference on World Wide Web (WWW 2009), Madrid, Spain, 2009: 861–870.

[9] Yan Li, Liu Jian, Ma Z M. Formal translation from fuzzy XML to fuzzy nested relational database schema[J]. Studies in Fuzziness and Soft Computing: Soft Computing in XML Data Management, 2010, 255: 35–54.

[10] Lau H L, Ng W. Storing XML documents in nested relational sequence relations[R]. The Hong Kong University of Science and Technology, 2002.

[11] Smith B, Ashburner M, Rosse C, et al. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration[J]. Nature Biotechnology, 2007, 25(11): 1251–1255.

[12] Barhamgi M, Benslimane D, Ouksel A M. Composing and optimizing data providing Web services[C]//Proceedings of the 17th International Conference on World Wide Web (WWW 2008), Beijing, China, 2008: 1141–1142.

[13] Truong H L, Dustdar S. On evaluating and publishing data concerns for data as a service[C]//Proceedings of the 2010 IEEE Asia-Pacific Services Computing Conference (APSCC), Hangzhou, China, 2010. Washington, DC, USA: IEEE Computer Society, 2010: 363–370.

[14] Caverlee J, Liu L, Rocco D. Discovering and ranking Web services with BASIL: a personalized approach with biased focus[C]//Proceedings of the 2nd International Conference on Service Oriented Computing (SCC ’04). New York, NY, USA: ACM, 2004: 153–162.

[15] Simmen D E, Altinel M, Markl V, et al. Data mashups for intranet applications[C]//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD ’08), Vancouver, BC, Canada, 2008. New York, NY, USA: ACM, 2008: 1171–1182.

[16] Tuchinda R, Szekely P A, Knoblock C A. Building mashups by example[C]//Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI ’08), Canary Islands, Spain, 2008. New York, NY, USA: ACM, 2008: 139–148.

[17] Lee K Y, S J H, Kim M H. Efficient incremental view maintenance in data warehouse[C]//Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM ’01), Atlanta, Georgia, USA, 2001. New York, NY, USA: ACM, 2001: 349–356.

附中文参考文献：

[4] 杨少华，张利永，韩燕波. 一种支持 Web 信息资源服务化的标记语言及其软件工具[J]. 计算机研究与发展, 2006, 43(增刊): 11–17。

WEN Yan was born in 1984. She is a Ph.D. candidate at Institute of Computing Technology, Chinese Academy of Sciences. Her reearch interests include service computing and data integration, etc.

温彦(1984—), 女, 山西榆次人, 中国科学院计算技术研究所博士研究生, 主要研究领域为服务计算, 数据集成等。

LIU Chen was born in 1980. He received his Ph.D. degree from Chinese Academy of Sciences in 2008. Now he is an associate researcher at Institute of Computing Technology, Chinese Academy of Sciences. His research interests include service computing and data integration, etc.

刘晨(1980—), 男, 四川成都人, 2008 年于中国科学院获得博士学位, 现为中国科学院计算技术研究所助理研究员, 主要研究领域为服务计算, 数据集成等。

HAN Yanbo was born in 1962. He is a professor and Ph.D. supervisor at Institute of Computing Technology, Chinese Academy of Sciences. His research interests include distributed computing, service computing and enterprise integration, etc.

韩燕波(1962—), 男, 中国科学院计算技术研究所教授、博士生导师, 主要研究领域为分布式计算, 服务计算, 企业集成等。