计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190500085

基于增量日志的数据组合视图定位更新方法

张元鸣　李梦妮　黄浪游　陆佳炜　肖　刚
浙江工业大学计算机科学与技术学院　杭州 310023

摘　要　数据服务作为一种面向跨域异构数据源的统一数据模型,能够将数据资源以服务的形式进行发布,并根据用户的数据需求,通过组合若干个数据服务生成数据组合视图。然而,由于数据源是自治的,当数据发生变化时,如何以最小的代价实时更新数据组合视图是数据服务技术需要解决的关键问题。为此,提出一种基于增量日志的数据组合视图定位更新方法,先根据数据源日志的增量变化获取最新变更数据,然后通过定位属性计算组合视图中差异元组的索引号,并根据变更类型直接对组合视图中的差异元组执行数据更新操作,最终给出了基于日志的更新数据实时获取算法和数据组合视图定位更新算法。在跨域异构电梯数据服务系统中对本更新方法进行了评价,结果表明,当变更元组数量所占比例远小于元组总数或数据组合视图的属性个数较多时,定位更新方法的更新效率比现有方法高。

关键词:数据服务;数据组合视图;定位更新;增量日志;实时性
中图法分类号　TP311

Data Composition View Positioning Update Approach with Incremental Logs

ZHANG Yuan-ming,LI Meng-ni,HUANG Lang-you,LU Jia-wei and XIAO Gang
College of Computer Science and Technology,Zhejiang University of Technology,Hangzhou 310023,China

Abstract　Data resources stored in different units and departments in cloud environment are cross-domain,heterogeneous and complex. As a unified data model for cross-origin and heterogeneous data sources,data service can publish data sources in the form of services,and generate data composition view by composing several data services according to users' data requirements. Since the data sources are autonomous,it becomes a key issue to update data composition view in real time with minimal cost. This paper proposes a data composition view positioning update approach based on incremental logs. The latest data changes of data sources are captured according to incremental logs,and then the attributes and tuples in data composition view are indexed. The index numbers of different tuples can be calculated with positioning attributes. The corresponding tuple update operations can be performed according to data changes' type. A log-based update data acquisition algorithm and a data composition view positioning update algorithm are presented. The proposed approach has been evaluated in a cross-origin heterogeneous elevator data service system by using datasets from multiple departments. When the proportion of the number of changed tuples is much smaller than the total number of tuples,the update efficiency of positioning update approach is much higher than existing methods. When the number of attributes of the data composition view is larger,the update efficiency of the positioning update approach is much higher than existing methods.

Keywords　Data service,Data composition view,Positioning update,Incremental logs,Real time

1　引言

随着云计算、物联网、移动互联网等信息技术的快速发展和广泛应用,几乎每个人、每台设备都成为了数据的生产者,使得数据的类型和规模以前所未有的速度增长。这些数据往往采用大量不同类型的格式存储,具有跨域、异构和自治等特点。在服务计算(Service Computing)模式下,不仅软件以服务的方式被发布在互联网上,称为 Web 服务(Web Service)[1],而且数据也以服务的形式被发布在互联网上,称为数据服务(Data Service)[2]。这种模式屏蔽了底层数据多源、异构等特征,为跨域数据的集成提供了一个统一的数据模型。通过组合方法生成的数据组合视图为用户提供了更加丰富的数据资源。与传统的物化视图[3]相比,该技术无须将数据复制到本地,成为数据即服务(Data as a Service,DaaS)的关键

到稿日期:2019-05-17　返修日期:2019-08-15　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:浙江省重大科技专项(2018C01064);浙江省自然科学基金项目(LY19F020034)
This work was supported by the Major Science and Technology Projects of Zhejiang Province (2018C01064) and Natural Science Foundation of Zhejiang Province(LY19F020034).
通信作者:张元鸣(zym@zjut.edu.cn)

技术[4]，有利于实现跨域异构数据的共享和分析。

数据组合视图是通过组合并执行多个数据服务得到的虚拟数据视图[5]。由于数据源是自治的，如果在数据发生变更时重新执行数据服务来更新数据组合视图，将导致大量不必要的数据传输，也难以满足实时性要求。因此，如何以最小的代价实时更新数据组合视图是数据服务技术面临的一个关键问题。针对该问题，国内外学者提出了基于轮询机制的重载更新方法[6]、基于计数模型的增量更新方法[7]和基于触发机制的 Delta 更新方法[8]等。轮询机制以轮询的方式检查数据的新鲜度，需要根据一系列的数据更新特征预先确定轮询频率，数据的实时性较弱；增量更新避免了重新加载数据组合视图，只计算视图中的差异部分并进行更新；触发机制则是在数据更新时通过事件触发更新操作，执行相关的数据服务。

日志文件是数据库系统中一类重要的文件，存储了数据的相关更新历史记录[9]。例如，Oracle 和 SQL Server 等数据库采用事务日志，按时间逆序记录数据更新的前像和后像；MySQL 数据库则通过二进制日志 binlog 记录数据更新。相关应用程序接口可以读取到各类数据库的日志文件，通过同步日志文件的变化感知数据的变化[10]。为此，本文提出了基于增量日志的数据组合视图定位更新方法，先通过数据源的增量日志获取最新的变更数据，然后通过定位属性计算组合视图中差异元组的索引号，并根据变更类型直接对组合视图中的元组执行数据更新操作。本文的主要贡献包括：

(1)设计了数据组合视图更新框架，通过监测数据源的日志变化来获取增量日志，并提取最新的变更数据。

(2)提出了数据组合视图定位更新机制，根据定位属性快速匹配、定位到组合视图中的差异数据，并快速更新。

(3)在真实的数据服务系统中对组合视图定位更新方法进行了评价，分析了定位更新方法的同步延时和数据组合视图的更新效率。

2 相关研究

数据服务以类似于 Web 服务的形式，将数据资源作为服务[11]，通过其所提供的接口进行访问，并输出一个确定模式的数据集，其是当前 DaaS 领域的一个研究热点。通过组合已发布的数据服务，可以生成灵活多样的数据组合视图，能够有效地支持跨域异构数据的共享与分析[12]。

在数据服务组合与数据组合视图生成方面，Altinel 等[13]给出了一个数据集成工具 Damia，该工具提供了基于浏览器的用户界面，用户可以直观地选取数据服务并用操作符连接输出数据组合视图。Zhang 等[5]根据数据内在的依赖关系构建数据服务依赖图，将数据服务组合问题建模为数据服务搜索问题，并根据数据需求自动搜索依赖图得到最优复合数据服务，生成符合需求的数据组合视图。Zhang 等[14]将跨域数据源转换成嵌套关系模型[15]的数据服务，根据视图的需求、数据服务的请求频率等设置的条件，生成多个运行效果等价的候选复合数据服务，并将这些复合数据服务及其运行方式建模为 0-1 规划的数学模型，利用遗传算法为用户推荐一组最优的复合数据服务，使得数据组合视图的构建代价最小。Han 等[16]提供了一个电子表格形式操作界面的工具 Mashroom，该工具将各种异构数据源封装为具有统一描述模式和访问方式的数据服务，并为用户提供了一个可视化的集成环境和界面级的数据服务聚合操作，可以灵活地按需汇聚数据，协助用户快速构建嵌套表格形式的数据组合视图，根据不同业务动态地获取满足需求的数据内容。针对数据服务返回的结果具有不确定性的问题，Amdouni 等[17]通过一种概率模型处理数据服务描述、执行和组合过程中的不确定性，基于可能世界理论定义数据服务组合的语义，并给出一组最优数据服务组合方案；而 Malki 等[18]研究了数据服务组合过程中语义不确定性的问题。Vu 等[19]提出了一个数据服务描述模型，除了数据服务本身的基本信息外，还集成了数据契约、数据相关性、服务质量等内容，为数据服务发现和组合提供了丰富的信息，有利于提高数据集成的自动化程度。

在数据组合视图更新方面，Wen 等[6]提出了一种基于轮询的数据组合视图重载更新方法，该方法预先对数据源进行建模，将数据模型、数据质量、数据更新频率、方式和数据量等数据更新特征作为数据组合视图更新的依据，预估数据组合视图的更新频率，通过执行数据服务生成新的数据组合视图。Gupta 等[7]提出了视图增量更新思想，通过增量更新程序对视图中的差异部分进行计算和更新，减小了计算量。Zhang 等[8]提出了一种基于嵌套关系模型的数据组合视图 Delta 更新方法，该方法将数据服务和数据组合视图的所有关联记录到本地文件中，运用指针为数据组合视图中的元组建立数据服务的引用，使得每当数据源发生数据更新操作时就触发相关数据服务的执行，并计算出差异元组，实现嵌套视图的增量更新。Liu 等[20]为异构数据源提供了统一的语义，不同数据源的模式映射到全局本体，通过模式映射组合数据服务，为用户提供了一个交互式的数据组合视图以满足复杂的数据请求，随着本地数据模型的更改，只须对映射进行相应的调整，无须修改全局数据模型。

现有的数据组合视图更新方法还存在一定的不足。例如，文献[6]基于轮询方式重新执行复合数据服务，生成新的组合视图，适用于更新频率较固定的数据资源，且可能造成资源浪费；文献[7]提出的计数更新方法需要遍历视图中的所有元组来查找差异部分，更新效率较低；文献[6,8]通过事件触发关联数据服务的执行来更新组合视图中的差异部分，虽然实时性较高，但配置比较复杂且会占用较多的系统资源。针对上述问题，本文利用数据源的增量日志来获取更新数据，并对数据组合视图的属性和元组建立索引，实现视图中差异元组的快速定位和更新。该方法一方面减少了不必要的数据传输，另一方面可以提高数据组合视图的更新效率，具有较高的实时性。

3 数据组合视图更新框架

本节先给出数据服务的基本概念，然后给出一个数据组合视图更新的框架。

定义 1(原子数据服务) 可独立访问且语义不可再分的数据服务称为原子数据服务(Atomic Data Service，ADS)。

一般地，原子数据服务封装了若干个基本属性、数据源地址和访问权限，执行原子数据服务的结果是一个数据视图。

定义 2（数据服务依赖图） 根据原子数据服务之间的数据依赖关系构建的依赖图称为数据服务依赖图（Data Service Dependence Graph，DSDG）。

数据服务依赖图描述了原子数据服务之间的数据依赖关系。根据该依赖关系，可以对原子数据进行组合，得到复合数据服务。

定义 3（复合数据服务） 由若干原子数据服务组成且可被独立访问的数据服务称为复合数据服务（Composite Data Service，CDS）。

设 $Schema$ 表示复合数据服务的模式结构，$Tuple$ 表示执行复合数据服务得到的元组集合，则复合数据服务的执行结果可以表示为 $CDS=\langle Schema, Tuple\rangle$。

定义 4（数据组合视图） 通过执行复合数据服务，并对要求的属性列进行投影操作后生成的虚拟数据视图称为数据组合视图（Data Composition View，DCV），可以表示为 $DCV(CDS)=\langle PI(Schema), PI(Tuple)\rangle$。

定义 5（集合操作） 对于 $DCV(CDS_1)$ 和 $DCV(CDS_2)$，设 $PI(Schema_1)=PI(Schema_2)=R$，通过集合操作生成新的 $DCV(CDS_3)$，即 $DCV(CDS_3)=DCV(CDS_1)*DCV(CDS_2)$，则 $PI(Schema_3)=R$，$PI(Tuple_3)=PI(Tuple_1)*PI(Tuple_2)$，其中 $*$ 可以为交、并或差运算中的一种，分别对应符号 $\cap$，$\cup$，$-$。

原子数据服务、复合数据服务、数据服务依赖图和数据组合视图之间具有如下关系：首先将各类跨域异构的数据源封装为原子数据服务，并注册在数据服务中心；根据原子数据服务的依赖关系构建数据服务依赖图，该依赖图反映了原子数据服务之间的依赖关系；根据数据需求，通过搜索相关的原子数据服务，组合生成复合数据服务，最终通过执行复合数据服务得到数据组合视图。

当数据源中的数据发生变化时，如何实时更新数据组合视图成为数据服务技术需要解决的一个关键问题。由于数据更新操作将被记录在数据库系统的日志文件中，因此在数据服务中增加一个日志组件用以监测日志的变化并进行增量日志的同步，通过解析该增量日志能够得到更新的数据。

图 1 给出了一种面向数据服务的组合视图更新框架。每个数据源抽取出的数据服务部署在数据拥有者指定的节点上，以保证数据的安全性和数据服务的稳定性，日志组件与数据源一一对应，同步的增量日志完成解析后会汇总到数据服务中心统一处理。日志组件主要负责数据源的日志同步和日志内容解析，包括 Tracker 模块、Parser 模块和 Store 模块。

（1）Tracker 模块：该模块负责与数据源端进行交互，监听数据源发送的增量日志；

（2）Parser 模块：由于原始的日志信息无法直接利用，该模块主要负责将增量日志解析成事件 event 并转发给 Store 模块；

（3）Store 模块：该模块主要暂存接收的 event，以供数据服务中心消费。

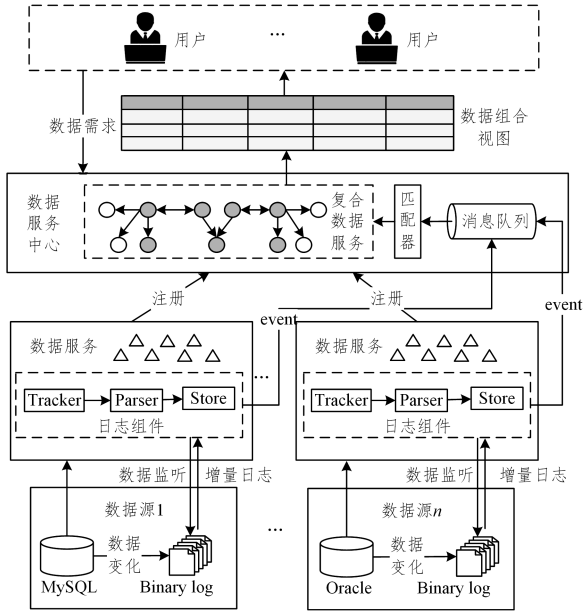

图 1 数据组合视图更新框架

Fig. 1 Data composition view update framework

数据服务中心依次提取 event 中的变更类型和变更内容，通过匹配器对数据组合视图进行检测。若数据组合视图包含了变更属性，通过与数据组合视图中的数据进行匹配，对数据组合视图与最新数据源的差异数据进行更新。

4 数据组合视图定位更新机制

4.1 更新数据获取

数据服务中的日志组件被用来同步检测点之后的日志数据，数据源端收到请求后，若日志检测点之后存在日志或写入了新的日志，则将增量日志及其日志位点信息传送给日志组件；日志组件监听到数据源端发送的信息后，对增量日志进行解析，并返回新的日志检测点；数据源端根据新的检测点读取日志，循环往复，获取更新数据。算法 1 给出了更新数据实时获取的详细步骤，算法获取更新数据之后返回新的监听位置。

算法 1 更新数据实时获取算法

```
Input: Connection information and log position
Output: New log position
1. open the log mechanism of data source;
2. configure the Data source Connection information;
3. set the log position;
4. if connect to data source then // Tracker module
5.    monitor log file;
6.    while listen to arrival of log data do
7.       get incremental logs; // Parser module
8.       event←Parse(incremental log); // Parser module
9.       Store. enqueue(event); // Store module
10.         return new log position;
11.         if network connection interrupted then
12.            break;
13.         end if
14.      end while
15. end if
```

以 MySQL 数据库为例，数据服务中的日志组件实时获取数据库的 binlog，提取并解析增量日志，以事件的形式进行记录，内容包含同步时间和日志位置信息以及其中每个事务的起始和结束时间、数据操作内容等信息，具体事件数据格式如表 1 所列。

表 1 事件的数据格式

Table 1 Data format of event

Entry	Header	logfile_name	binlog 文件名
		logfile_offset	binlog 字节偏移位置
		execute_time	变更时间
		schema_name	模式名称
		table_name	表名称
		event_type	变更类型
	entry_type	事务头 BEGIN/事务尾 END/数据 row_data	
	store_value	byte 数据，对应类型为 Row_change	
Row_chang	is_ddl	是否是 DDL 操作	
	sql	DDL 操作的 sql 语句	
	row_datas	before_columns	变更前数据，可为多条
		after_columns	变更后数据，可为多条
Column	index	列索引	
	sql_type	jdbc 类型	
	name	属性名称	
	is_key	是否为主键	
	updated	是否变更	
	is_null	是否为 null	
	value	值	

当 MySQL 数据库发生了更新操作时，解析增量日志得到相应的 event，其主要内容如图 2 所示。其中，Batch Id 是增量日志的唯一标识，其中可能包含多个事务；Start 和 End 分别对应增量日志同步的起始时间和完成时间；Event 的剩余部分依次列出每个事务对应的所在 binlog 文件名、执行时间、拉取延迟时间、变更类型和变更对象等信息，指明变更对象所在的数据库名称、表名以及其中所有属性的具体信息。

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Batch Id:[5],count:[3],memsize:[180],Time:2018-12-09 12:23:36
Start:[mysql-bin.000011:2672:1544329416000(2018-12-09 12:23:36)]
End:[mysql-bin.000011:2879:1544329416000(2018-12-09 12:23:36)]
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
binlog[mysql-bin.000011:2672],executeTime:1544329416000
(2018-12-09 12:23:36),gtid:(),delay:380ms
- - - - - - - - - - - - - - - - - - - - - - - - -
BEGIN  Thread id:22
binlog[mysql-bin.000011:2806],name[design,customer],eventType:UP-
DATE,executeTime:1544329416000(2018-12-09 12:23:36),gtid:(),delay:
380 ms
id:3    type=int(11) unsigned
name:a    type=varchar(255)
sex:0    type=int(11)
address:XXX    type=varchar(255)    update=true
...
- - - - - - - - - - - - - - - - - - - - - - - - -
END  transaction id:638
binlog[mysql-bin.000011:2879],executeTime:1544329416000
(2018-12-09 12:23:36),gtid:(),delay:380ms
...
```

图 2 增量日志中的事件记录

Fig. 2 Events in incremental logs

4.2 数据组合视图更新算法

在数据源更新不频繁，且数据组合视图包含的数据较少或者更新数据比例较大的情况下，可以重新执行复合数据服务生成新的数据组合视图，或者遍历所有的数据对差异部分进行相应的更新操作。然而，当数据源更新频繁，且数据组合视图包含的数据较多，更新数据占比较小时，重新生成或遍历生成新的数据组合视图所需的时间将会增加，难以满足实时性要求。

为了提高数据组合视图的更新效率，本文提出了一种数据组合视图定位更新方法，对数据组合视图中的属性和元组建立索引，并选择某些属性当作定位属性，以此作为更新数据组合视图的主要依据。由于主键属性值可以唯一确定其他属性值，因此一般选择视图中的主键属性作为定位属性。然后，将增量日志中定位属性的属性值与数据组合视图中定位属性的属性值进行匹配，如果匹配成功，则根据相关索引对该视图中与数据源存在数据差异的元组进行定位并直接执行数据更新操作，否则不予更新。

在数据组合视图的生成阶段，根据定位属性值对视图中的元组进行分块，相同的定位属性值被划分在同一块中，并对块内的元组赋予索引号。如果定位属性包括多个属性，则选取其中一个即可，从而能够根据块号和索引号进行快速定位。只要准确地定位到视图中的元组，就可以实时更新视图。与重载更新方法[6]相比，定位更新所需的网络传输数据量大幅减少；与计数更新方法[7]相比，定位更新能够直接定位需要更新的元组，找到变更元组的时间更少；与 Delta 更新方法[8]相比，定位更新方法能够避免重新执行部分数据服务与相关集合操作。

图 3 给出了一个电梯维修数据组合视图的更新过程，右侧的视图包括注册代码、设备编号、规格型号、零件 ID、维修项目、维修状态等属性。其中的数据来源于维修部门和制造部门，视图的定位属性是注册代码和零件 ID，按注册代码的属性值对视图进行分块。数据源的更新类型包括以下 3 类。

(1)INSERT 类型：若维修部门在其所属的数据源中新增了一条电梯维修记录，数据服务的日志组件获取更新数据，其中零件 ID 和注册代码是定位属性，它们的属性值分别为 m0151 和 3110…025。根据注册代码的属性值，可以唯一确定数据组合视图中对应设备型号和规格型号的属性值以及新增数据所属的块号，将新增数据拼接成完整元组，对数据组合视图的 1 号块执行插入操作。

(2)DELETE 类型：若制造部门在其所属的数据源中删除了注册代码为 3210…006 的电梯的基本信息记录，根据定位属性值匹配数据组合视图，定位得到块号为 2。由于数据组合视图的划块属性就是注册代码，因此可以直接对 2 号块内的元组执行删除操作。

(3)UPDATE 类型：若维修部门又将一条电梯维修记录的维修状态修改为已完成，则根据该定位属性值同理定位得到块号为 1，元组索引号为 2，对该元组中维修状态的属性值进行更新操作。

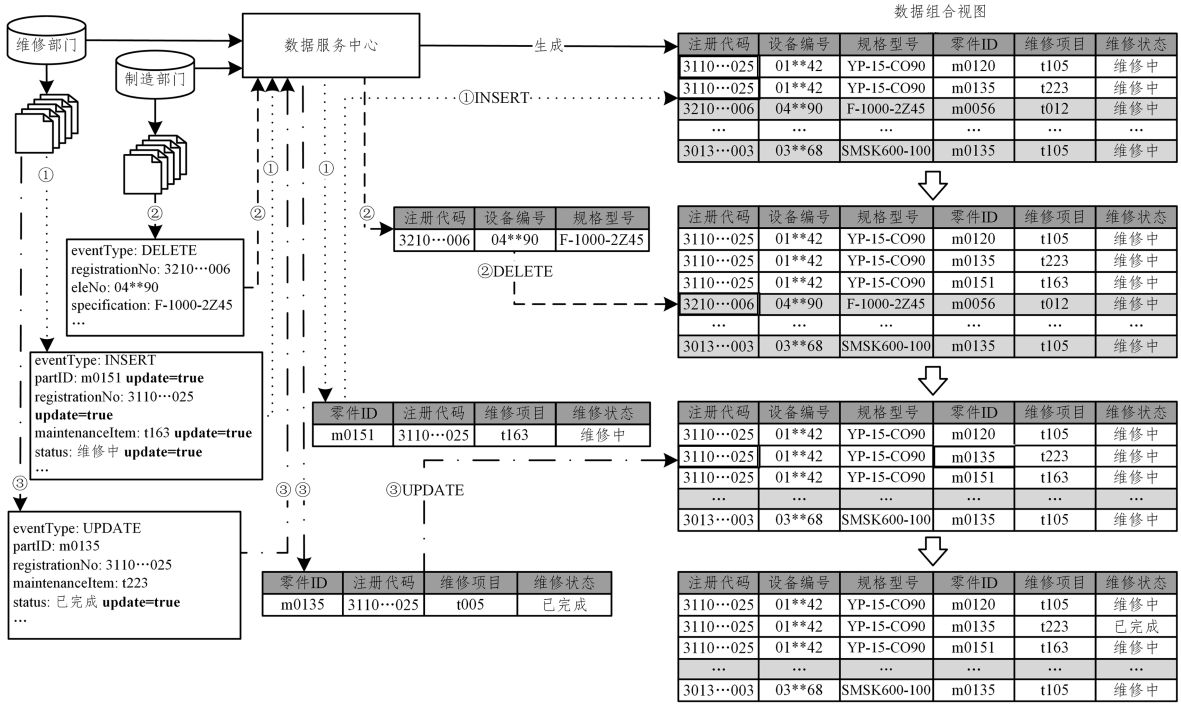

图 3　电梯维修状态数据组合视图定位更新过程

Fig. 3　Elevator maintenance data composition view positioning update procedure

算法 2 给出了数据组合视图的定位更新算法。从增量日志中提取变更类型、Schema 中的属性及其属性值，再选择其中的主键属性作为定位属性，与视图中对应的定位属性进行匹配。若变更类型为 INSERT，则根据定位属性值得到其在数据组合视图中的块号，再根据定位属性之间的依赖关系对新增数据和块其余属性值进行拼接，最后将得到的元组添加进所属块。若变更类型为 DELETE，且定位属性与数据组合视图的划块属性相同，则根据定位属性值定位到块号，直接将块内的所有元组删除；否则需要根据所有定位属性值依次定位变更元组的索引号并进行删除操作。若变更类型为 UPDATE，则根据定位属性值定位到数据组合视图中更新元组的索引号，并对变更属性值进行更新操作。

算法 2　数据组合视图定位更新算法

```
Input: Event
Output: DCV^new
1. Type←get the event type from the Event;
2. Content←get the content of schema. attributes from the Event;
3. DCV←get the DCV from Cache;
4. pos_attributes←get the positional attributes from DCV
5. partition←select one of pos_attributes
6. if differences exist in DCV then
7.     block←getBlock(partition);
8.     switch Type do
9.         case INSERT
10.        insert←get insert data from Content;
11.        tuple←newTuple(block, pos_attributes, insert);
12.        DCV^new←insertTuple(tuple);
13.        break;
14.        case DELETE
15.        if Content. pos_attribute equals to partition then
16.            DCV^new←deleteTuple(block);
17.        else
18.            indexes←getIndex(pos_attributes)
19.            DCV^new←deleteTuple(indexes);
20.        end if
21.        break;
22.    case UPDATE
23.        index←getIndex(block)
24.        new_data←get the update attribute;
25.        DCV^new←updateTuple(new_data);
26.        break;
27.    otherwise // in case of ALTER, TRUNCATE or DROP
28.        DELETE(DCV);
29.        return null;
30.    endsw
31.  endsw
32. end if
33. return DCV^new.
```

设 n 为数据组合视图的属性个数，算法 2 通过定位属性值定位数据组合视图中需要更新的元组，在根据块号可以直接执行更新操作的情况下，时间复杂度为 O(1)；其余情况下需要通过块号继续检索块内索引号进行更新操作，此时的时间复杂度为 O(n)。因此，整个算法的时间复杂度为 O(n)。

5　实验结果

本节利用电梯企业多个部门的数据集[21]来评价数据组合视图定位更新方法，将各部门的数据集封装为数据服务，并注册在数据服务中心。数据服务中心的运行环境是一个 8 核

主频为 2.50GHz 的 CPU，16GB 内存，40GB 系统盘和 250GB 数据盘，操作系统采用 64 位 Ubuntu 16.04。

5.1 更新数据同步延时

更新数据同步延时用来评价增量日志内容传输的实时性，主要由两部分组成：数据源的数据操作发生到增量日志写入磁盘的过程和增量日志同步至数据服务中心完成解析的过程。更新效率用来评价数据组合视图更新所花费的时间代价。实验针对 INSERT，DELETE 和 UPDATE 3 种变更类型，每次增量日志所解析出的元组总数分别为 1，50，100，200，400，测试从数据变更操作写入日志到增量日志同步至数据服务端之间的延迟时间。每组测试 10 次，将测试结果中的最大值和最小值去掉后取平均值作为实验的最终结果。

图 4 给出了不同变更类型的同步延时变化情况，3 种变更类型的同步延时随着数据量的增加都有小幅度的增加，但都能够在 1s 内获取到数据源的增量日志，能够保证后续数据组合视图更新的实时性。

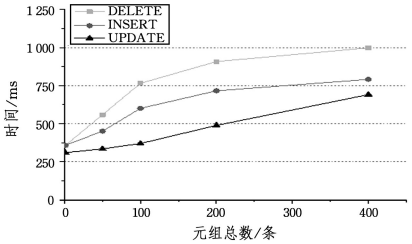

图 4 不同变更类型的增量日志同步延时

Fig. 4 Incremental logs synchronization delay for different change types

5.2 数据组合视图更新效率

本节进一步对重载更新方法[6]、计数更新方法[7]、Delta 更新方法[8]与本文提出的定位更新方法进行对比分析，重点比较这些方法的更新效率。

数据组合视图更新的主要影响因素有 Schema 中的属性个数和 Tuple 中的元组总数。本实验设计两组测试数据集，如表 2 所列。

表 2 实验数据集

Table 2 Experimental datasets

测试数据集类型	表个数	属性个数	数据量（元组数）
数据集 1	6	6	100，200，400，600，800
数据集 2	6	2，4，6，8，10	200

(1)测试数据集 1 保持表个数和属性个数不变，元组总数分别为 100，200，400，600，800；

(2)测试数据集 2 保持表个数和元组总数不变，属性个数分别为 2，4，6，8，10。

以随机的方式选取相应数量的表和属性生成数据组合视图，数据缓存在本地，对选取的属性进行变更操作，每次的变更元组数量在 50 以内，以测试结果的平均值作为实验结果。

图 5 给出了不同数据量的数据组合视图更新效率。当变更元组数量占元组总数的 50%左右时，3 种更新方式更新数据组合视图所需的时间开销比较接近；而随着元组总数的增加，变更元组数量与元组总数的比值不断变小，定位更新法在数据组合视图更新效率方面的优势更明显。综合来看，定位更新法能取得较好的数据组合视图更新效率。

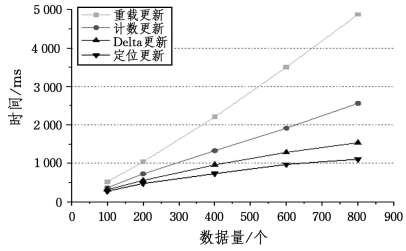

图 5 不同数据量的视图更新效率

Fig. 5 View update efficiency for different number of tuples

图 6 给出了不同属性个数的数据组合视图更新效率。可以发现，随着属性个数的增加，重载更新方法需要执行的复合数据服务包含的原子数据服务数量会增加，网络请求次数和传输数据量相应增多，生成新的数据组合视图的代价将会不断增加，更新数据组合视图的耗时明显多于其他 3 种方式。而定位更新方法较 Delta 更新方法在视图更新效率上没有明显的优势，说明了属性个数对定位更新方法的影响较大。总体来讲，随着模式集中属性个数的增加，定位更新方法、计数更新方法和 Delta 更新方法的耗时也都随之增加，定位更新方法需要维护更多的定位属性，计数更新方法需要更长的遍历时间，Delta 更新方法需要更长的差异元组计算时间。从实验结果看，定位方法仍然优于其他的更新方法。

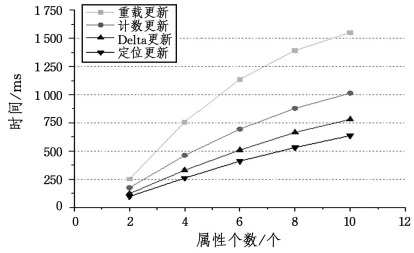

图 6 不同属性个数的视图更新效率

Fig. 6 View update efficiency for different number of attributes

结束语 为了高效地更新数据组合视图，本文提出了一种基于增量日志的数据组合视图定位更新方法：在数据服务中增加日志组件以监测日志的变化，通过增量日志提取最新的变更数据；按定位属性值对数据组合视图中的元组进行分块，对块内的元组赋予索引号，通过定位属性快速定位组合视图中的差异数据，并根据变更类型直接对差异元组执行数据更新操作。实验结果表明，本文提出的数据组合视图更新方法具有更高的实时性和更新效率。

下一步工作将通过数据服务技术封装半结构化和非结构化数据，如 XML 文件、办公文档、图片等，为跨域异构环境下的数据集成、共享、分析和应用提供统一的数据模型。

参 考 文 献

[1] LEMOS A L, DANIEL F, BENATALLAH B. Web service composition[J]. ACM Computing Surveys, 2016, 48(3): 1-41.

[2] CAREY M J,ONOSE N,PETROPOULOS M. Data services [J]. Communications of the ACM,2012,55(6):86.

[3] KURZADKAR S,BAJPAYEE A. Anatomization of miscellaneous approaches for selection and maintenance of Materialized view[C]// 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO). IEEE,2015:1-5.

[4] TERZO O,RUIU P,BUCCI E,et al. Data as a service (DaaS) for sharing and processing of large data collections in the cloud [C]// 2013 Seventh International Conference on Complex,Intelligent,and Software Intensive Systems. IEEE,2013:475-480.

[5] ZHANG Y M,YE C L,HUANG L Y,et al. Research on data service dependency graph model and automatic composition[J]. Journal of Chinese Computer Systems,2018,39(3):450-456.

[6] WEN Y,LIU C,HAN Y B. iViewer:service-based view construction method for just-in-time sharing business data across organizations[J]. Journal of Frontiers of Computer Science & Technology,2012,6(3):221-236.

[7] GUPTA A,MUMICK I S. Materialized views[M]. The MIT Press,1999.

[8] ZHANG P,HAN Y B,WANG G L. Implementing dynamic nested view update based on data service[J]. Chinese Journal of Computers,2013,36(2):226-237.

[9] RAGOTHAMAN P,PANDA B. Analyzing transaction logs for effective damage assessment[M]// Research Directions in Data and Applications Security. Boston,MA:Springer US,2003:89-101.

[10] ZOU X X,JIA W J,PAN J H. Research of log-based change data capture[J]. Journal of Chinese Computer Systems, 2012, 33(3):531-536.

[11] CAREY M,REVELIOTIS P,THATTE S,et al. Data service modeling in the AquaLogic data services platform[C]// 2008 IEEE Congress on Services. IEEE,2008:78-80.

[12] ZHANG Y,ZHU L M,XU X W,et al. Data service API design for data analytics[M]// Services Computing—SCC 2018. Cham:Springer International Publishing,2018:87-102.

[13] ALTINEL M,BROWN P,CLINE S,et al. Damia:A Data Mashup Fabric for Intranet Applications[C]// Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna:VLDB Endowment,2007:1370-1373.

[14] ZHANG P,WANG G L,JI G,et al. Optimization update for data composition view based on data service[J]. Chinese Journal of Computers,2011,34(12):2344-2354.

[15] CAO B,BADIA A. SQL query optimization through nested relational algebra[J]. ACM Transactions on Database Systems, 2007,32(3):18.

[16] HAN Y B,WANG G L,JI G,et al. Situational data integration with data services and nested table[J]. Service Oriented Computing and Applications,2013,7(2):129-150.

[17] AMDOUNI S,BARHAMGI M,BENSLIMANE D,et al. Handling uncertainty in data services composition[C]// 2014 IEEE International Conference on Services Computing. IEEE, 2014: 653-660.

[18] MALKI A,BARHAMGI M,BENSLIMANE S M,et al. Composing data services with uncertain semantics[J]. IEEE Transactions on Knowledge and Data Engineering,2015,27(4):936-949.

[19] VU Q H,PHAM T V,TRUONG H L,et al. DEMODS:a description model for data-as-a-service[C]// 2012 IEEE 26th International Conference on Advanced Information Networking and Applications. IEEE,2012:605-612.

[20] LIU X,HU C,LI Y,et al. The advanced data service architecture for modern enterprise information system[C]// 2014 International Conference on Information Science & Applications (ICISA). IEEE,2014:1-4.

[21] ZHANG Z J,ZHANG Y M,LU J W,et al. CMfgIA:a cloud manufacturing application mode for industry alliance[J]. The International Journal of Advanced Manufacturing Technology, 2018,98(9/10/11/12):2967-2985.

ZHANG Yuan-ming, born in 1977, Ph.D, associate professor. His main research interests include service computing,cloud computing and big data processing.