计算机科学2004Vol. 31№. 9A

一个异构数据源集成系统的实现*)

谭立球　费耀平　李建华
(中南大学网络中心　长沙410075)

摘　要　本文介绍了一个实现的异构数据源集成系统，它采用物化视图的方式，通过包装器和集成器将异构数据源集成起来，解决数据集成过程中的部分数据冲突问题。从整体构架上，考虑了系统的人机交互友好性。

关键词　数据集成，数据冲突，模式映射，物化视图，异构数据

An Heterogeneity Datasources Integration System

TAN Li-Qiu　FEI Yao-Ping　LI Jian-Hua
(Network Ceter, Central Universiy, Changsha 410075)

Abstract　A Heterogeneity Data Integration System is presented in this paper. It uses materialized view to integrate hetergeneity data source using wrapper and integrator, and it resloves some data conflict. It takes into account the human computer interaction from framework.

Keywords　Data integration, Data conflict, Schema mapping, Materialized view, Heterogeneity data

企业信息系统越来越丰富，每个系统都保存了非常丰富的数据，既包括有当前的信息，也包括历史的信息。从宏观决策的需要，非常迫切地希望将这些信息集中起来，提供一个全局的决策支持，这就是数据集成研究的内容。数据集成是将不同的数据源集成起来，以一个统一的视图提供给用户。这里的视图可以是物化的视图，也可以是虚拟的视图。物化视图是采用中心数据库(或数据仓库[1])来保存集成后的数据，而虚拟视图则是将实际数据保留在每个数据源中。通过数据集成后，用户在构建自己的查询时，不必知道数据是在哪个位置，源数据是怎样组织的。本文介绍了一个异构数据源集成系统的实现，该系统采用物化数据视图的方式，将不同数据源的数据集成起来，构建自己的中心数据库，以供在此基础上进行决策支持工作。

1　异构数据集成系统的定义

一个异构数据集成系统可以用三元组 $I=\langle G, S, M_{G,S}\rangle$ 来表达，G 代表全局模式，由定义在字母表 A_G 上的语言 L_G 来表达，G 中的每一个元素是由字母表 A_G 构成的符号。S 是局部模式，它是一个集合 $\{S_1, S_2, \cdots, S_n\}$，其中 $S_1, S_2, \cdots, S_n$ 分别表示不同的数据源的局部模式。每个局部模式 S_i 由定义在字母表 A_{Si} 的语言 L_{Si} 来表达，S_i 中的每一个元素是由字母表 A_G 构成的符号。M 是 G 和 S 之间的映射，由一系列形如 $R_G \rightarrow R_S$ 表达，其意义就是建立在全局模式上的关系总可以用局部模式上的关系来描述，而这种关系也不是简单的一一映射的关系。

数据集成主要是解决不同数据源的异构和冲突问题。人们常将这种冲突分为系统冲突和数据冲突。系统冲突是指外围采用不同的操作系统，采用不同的数据库类型等等。系统冲突可以在包装器中来实现，如采用 ODBC 连接实现对不同的关系数据库的统一访问。数据冲突可以分为三种类型[2]，模式冲突(schematic conflict)、语义冲突(semantic conflict)以及内涵冲突(Intensional Conflict)。语义冲突和模式冲突两者的区别可以看成结构以及结构表达的语义，也就是数据是怎样被逻辑组织的，而这些数据又表达了什么意思。而内涵冲突则是指不同信息系统中表达内容的差异。每种冲突又可以进行进一步细分[2]，目前数据冲突的解决也是目前数据集成的研究重点和难点。

2　系统框架图

数据集成负责异构数据源组织起来，以一个统一的模式提供给用户。在全局模式的存储上，采用构建物理中心数据库的方法来保存实际数据。其原因一方面基于查询效率考虑，同时避免因为远程数据源的网络中断以及其他故障导致远程系统出现停止服务时查询系统仍能够正常使用，同时也可以避免

谭立球　博士生，主要从事数据集成、Web 挖掘等方面的研究。

对远程数据库的日常工作造成很大的冲击。在目标数据的组织上，要采集什么样的数据，主要采用查询驱动的方法。也就是新需求需要了解什么样的数据，因而集成什么样的数据。其基本采集原则是，通用的基础数据是必须采集的，如一个单位的组织结构。同时也根据需求的粒度，来决定采集要集成数据的粒度。从驱动的角度划分，目前数据集成的方法可以分为数据源驱动集成的方法和用户需求驱动集成的方法。数据源驱动集成的方法是不管人们的数据需求如何，只要有新的数据源出现，则将新的数据源集成进来。而基于用户需求的集成方法，则是以满足用户的原则来构建，其基本思路是，当用户提出了新的查询需求以后，则首先判断在目前的全局模式中的数据能否满足，如不能，则到目前已有的局部模式中查找，如果还不能满足，则考虑加入新的数据源。

整个系统的结构如图1所示，整个系统以中心数据库为中心，将映射生成器、集成器、包装器和监控器、查询生成器和授权配置这几个部分组织起来。系统中映射生成器主要解决全局模式和局部模式的映射；包装器负责将数据源的信息提取；集成器负责将提取后的数据进行合成，其中集成器和包装器通过消息中间件实现交互；监控器负责对整个系统进行监控，包括发现数据源的数据发生变化；授权配置和查询生成器是本集成系统的后续工作，则不在本文的范围之内。

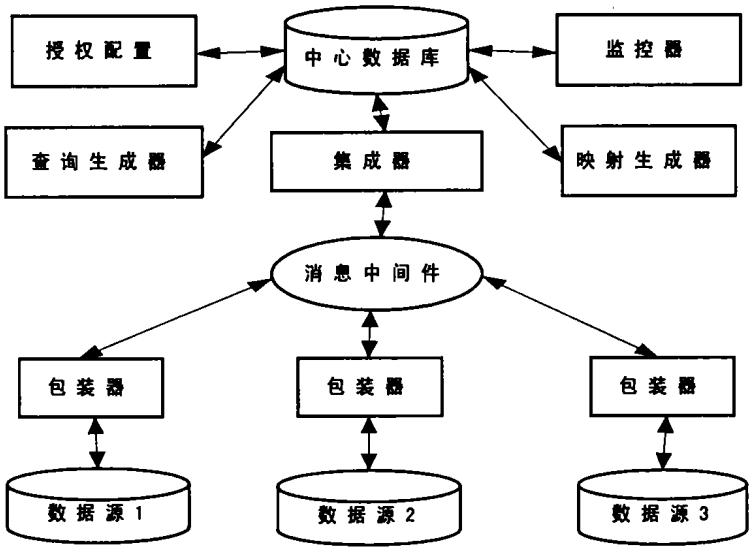

图1　数据集成系统结构

3　包装器设计

包装器在整个系统中处于底层，直接与数据源打交道，向上通过消息中间件与集成器进行交互。考虑到系统的通用及将来可扩充性，我们采用将包装器驻留在数据源的服务器上，这样可以避开文件型的数据库必须要开放网络文件共享的安全隐患。系统中完成的主要功能有：1)数据源模式信息的请求，并执行。数据源的模式信息主要包括数据源所包含的库表名，每个库表包含的字段名，以及主键及外键等。2)响应提取数据源实际数据的请求，将提取到的结果进行清洗。3)将局部执行结果封装成系统XML格式，发送到消息中间件。

4　映射生成器设计

映射生成器是解决源模式和全局模式的映射问题，也就是解决G和S模式的映射问题，是直接面向用户使用的一个工具，因而必须保持人机交互的友好性和易操作性，我们在实现时采用可视化界面来实现。由于从数据源到目标表的映射不是简单的搬迁过程，因而在映射的构造上，我们以填充满目标表为原则进行映射器的构造，也就是一个目标表的一条记录可能来自同一数据源的不同表，也可能来自不同数据源的数据。因而基于以上考虑，我们在映射生成器上采用以填充目标表为准则。映射生成器的主要任务包括：局部模式信息的维护、全局模式和局部模式的映射、数据冲突的维护以及采集任务的形成。

1)局部模式信息的维护

对于局部模式的元数据，通过专用的API，可以自动获取远程数据源的模式信息。其主要的功能有：a)读取远程数据源的库表名称；b)用户可以选择获取某一个表或所有表的模式信息，如字段号、字段属性、是否关键字、是否外键等信息；c)刷新某表的模式信息；d)对某些模式信息进行中文标定，如数据表每字段取一个中文名称，以利于用户沟通；e)对全局模式的元信息进行维护，该步骤主要通过管理员手工来实现的。

2)全局模式和局部模式映射维护

该映射是通过可视化界面来完成的。关于数据模式层冲突的解决都是在此处来实现的。主要完成的功能有:a)定义局部模式和全局模式的映射规则。整个映射是以支持填充目标表的填充原则。也就是根据目标表的结构,从数据源中寻找满足填充条件的数据。整个过程支持多个源表到一个目标表的映射,同时多个源表也可以来自不同的数据源;b)在映射规则的形成时除了用户可以定义标准的 SQL 函数之外,还可以支持自己定义函数的运算。

3)数据冲突的解决

对于模式层的冲突,我们采用定义前面所说全局模式与局部模式的映射来解决。但对于数据层的冲突,就比较复杂了,必须深入到细节数据具体情况具体处理。如需要加入一些参考表来解决数据冲突的问题。

4)采集任务的形成

映射规则形成以后,主要对于这些映射规则进行组合,形成一条采集任务,最终能将数据源的数据,填充在目标数据库中。

一个采集任务是能够完整地填充满目标数据库的一条记录。一个采集任务由若干原子任务构成,每个原子任务完成目标数据的部分工作,并且每个原子任务可能被分发到不同的数据源中去执行。

对于有些映射规则,可以直接集成到 sql 语句,因而在数据采集时就完成了数据的处理。而有些采集原子任务,必须对采集以后的结果进行二次处理,因而这些映射规则以与原子任务关联的方式保存下来。

5 集成器的设计

集成器在整个数据集成中处于中心和枢纽位置,其工作流程如图2所示。集成器接收到一个处理任务后,搜寻相关的原子任务。对于一组任务和原子任务,并且用 XML 文件来描述。其文件描述的内容主要有采集任务的 ID 号码,名称,以及所包含的原子任务。而原子任务的描述信息包括该原子任务是发向哪个数据源的,以及任务的内容。

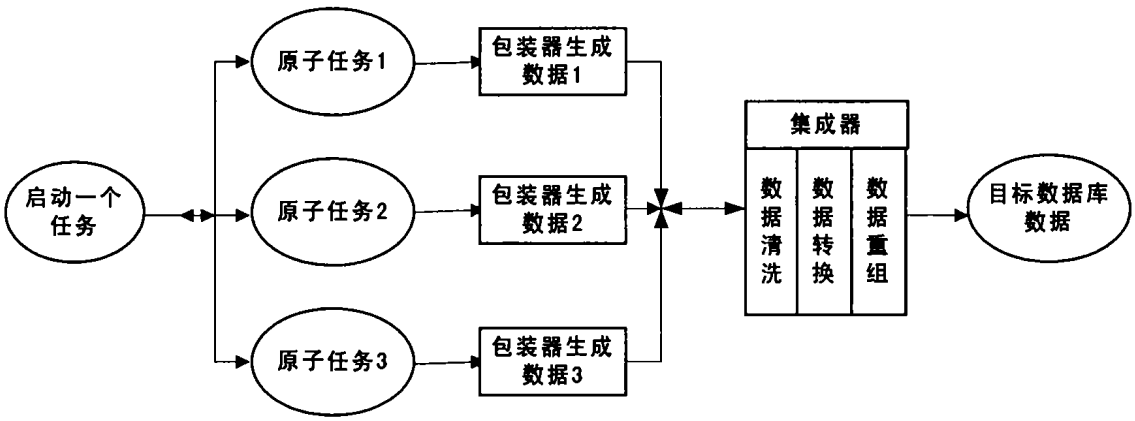

图2 数据集成流程图

集成器根据每个原子任务的描述,通过消息代理中间件,发给相应的包装器。包装器在执行完采集任务后,同时以 XML 文件的形式将采集结果发送给集成器。集成器只有在收到该任务下的所有原子任务的执行结果以后,才开始对这些结果进行重组。在包装器的处理过程中,完成了对一些简单的数据冲突的处理,而对于复杂情况的数据冲突处理都是在此处完成的。如从两个不同的数据源中数据合成一个新的数据,针对每种映射情况,集成器都要专门进行处理。经过清洗、重组后形成最终的目标数据,可以归入目标数据库中。

6 监控器的设计

对于一个自治系统来说,监控器是一个重要的部分,它负责启动中心数据库的刷新。对于数据更新而言,比较理想的方式就是增量更新,也就是每次只更新那些发生变化的数据。在数据集成中,数据的增量更新是一个难点。人们常采用的方法有触发器、时表、日志法。触发器的方法是在数据源的数据库中设立触发器,数据一旦发生写入、删除以及修改,都会将这些记录记载下来,时标法则是对数据库最近更新时间与最近一次采集的时间进行比较。日志法则是要求每个应用程序对于发生的数据变更进行日志。在项目中,我们采用了采集日志和时标相结合的方法。监控器主要有两个方面的作用,一个是监控数据源数据是否发生变化,一旦发生变化,则相应地启动相关的采集任务,同时也周期性地扫描目前已有采集任务的触发点。在实际过程中,我们定制了可以有远程数据源主动发起数据更新服务,这样实际加入了实际使用人员的参与,达到了一个较好的数据更新与负载的均衡。

小结 本文介绍了我们实现的一个异构数据源集成系统,它采用物化视图的方式,通过包装器和集成器将异构数据源集成起来,解决了数据集成过程

中的部分数据冲突问题。从整体构架上，考虑了系统的通用性和人机交互友好性。今后的研究方向主要集中于采用更多更好的方法解决数据集成中的数据冲突问题，如采用基于本体的集成[3]等。

参 考 文 献

1 Calvanese D，degiacomo G，Lenzerini M，Nardi D，Rosati R. Data Integration in Data Warehousing. Internation Journal of Cooperative Information Systems，2001，10(3)：237～271

2 Building L L Y，Ozsu T，Liu L. Accessing heterogeneous data through homogenization and integration mediators. In：Proc. 2nd IFCIS Conf. on Cooperative Information Systems (CoopIS-97)，1997. 130～139

3 Buccella A，Cechich A. An Ontology Approach to Data Integration. JCS&T，3(2)

(上接第111页)

来优化文档查询，只要扫描一遍规模较小的 DTD 树，即可省去大量的对文档树所做的无用的遍历。平均来说，文档树的结点数与 DTD 树的结点数的比越大，我们方法的效率越高。

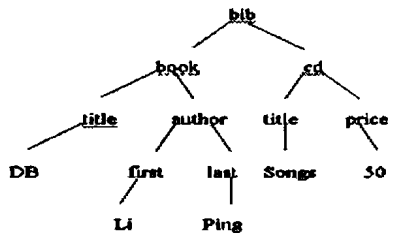

图6 文档树

我们来看一下它们的 I/O 情况，不用 DTD，直接对 XML 文档树进行扫描查询，则要扫描一遍 XML 文档，假设内存还有 s 个缓冲区，XML 文档有 d 个块，这时扫描一遍 XML 文档树的 I/O 为[d/s]，且每次查询时都要有[d/s]次 I/O；若用 DTD 树对 XML 文档树进行扫描，一般的查询方法是在扫描文档树的同时，参考 DTD 树。这样，每扫描一条路径，都要相应地扫描一次 DTD 树，假设 DTD 树有 n 个块，若利用回溯法，假设 XML 有 r 个叶子结点，即相应地有 r 条路径，设利用回溯法需扫描的路径有 r1条(r1＜＝r)，所以，这时在这棵 XML 树上做一次查询扫描 DTD 树共需要 r1[n/s]次 I/O，扫描文档树中的 r1条路径需[r1d/r/s]次 I/O，所以共需 I/O 次数为 r1[n/s]＋[r1d/r/s]；通常 DTD 的规模远远小于 XML 文档树的规模，从以上分析，容易看出，利用 DTD 模式查询 XML 树与直接查询 XML 文档树相比较，通常会大大地减少 I/O 次数，并且 r1越小，即根据 DTD 树在扫描 XML 文档树时去掉的分支越多，前者的方法会更有效；而本篇文章中提出的方法，在扫描一遍 DTD 树得到真路径时，I/O 次数为[n/s]，得到的真路径数组中只记着真路径中经过的结点的地址，所以占很少的空间，如果经常有同样的查询(指能用到找出的真路径)，可以让真路径数组常驻内存，可直接用。设真路径中结点有 trn 个块，因为它们只为 DTD 树中的部分结点，所以 trn＜＝n，通常情况下，trn≪n，扫描真路径时，是与数组中的有用的真路径的第几个结点比的，所以可直接从真路径数组中直接到外存中找到所需 DTD 树中结点的地址，可直接将其取到内存中，并且，往往不是每条真路径都有用，所以真正用到的真路径的结点还不是全从外存中取一遍，即在对 XML 树做查询的过程中并不是再全部访问一遍 DTD 树中真路径所经过的结点，而只是其中的一部分，所以查询过程中，扫描真路径的 I/O 次数要小于等于[trn/s]，所以据我们的方法对 XML 文档树做一次查询需要的 I/O 次数要小于等于[n/s]＋[trn/s]＋[r1d/r/s]，r1为在限界的条件下扫描的 XML 文档树中的路径条数，所以通常 r1＞2，并且 trn＜n，所以有：[n/s]＋[trn/s]＋[r1d/r/s]＜2[n/s]＋[r1d/r/s]＜r1[n/s]＋[r1d/r/s]，所以我们的方法更能减少 I/O 次数，更能提高效率。

参 考 文 献

1 Structural Function Inlining Technique for Structurally Recursive XML Queries

2 clark J，DeRose S. XML Path Language(XPath) Version 1.0 w3c Recommendation. Nov. 1999. http://ww. w3. org/TR/1999/REC-xpath-19991116

3 Bacon D F，Graham S L，Sharp O J. Compiler transformations for high-performance computing. ACM Computing Surveys，Dem. 1994，26(4)：345～420

4 Boag S，Chamberlin D，Fernandz M，Florescu D，Robie J，Simeon J，Stefanescu M. Xquery 1.0：An XML query language. Working Draft. http://www，w3. org/TR/2001/WD-xquery＝20011220. Dec. 2001

5 Buneman P，Fernandez M，Suciu D. UnQL：a query language and algebra for semistructured data based on structural recursion. The VLDB Journal，2000，9：76～110

6 Chamberlin D，Fankhauser P，Marchiori M，Robie J. XML query use cases. Working Draft. http://www. w3. org/TR/2001/WD-xml query-use-cases-10011220. Dec. 2001

7 Fankhauser P，Fernandez M，Malhotra A，Rys M，Simeon J，Wadler P. The Xquery 1.0 formal semantics. Working Draft. http://www. w3. org/TR/2001/WD-query-semantics-20010607. June 2001