数据预处理在数据仓库体系结构中的应用

贺 敏[1] 王蔚韬[1] 何光辉[2]

(重庆大学计算机学院 重庆400044)[1] (重庆大学数理学院 重庆400044)[2]

摘 要 本文通过对原有数据仓库体系结构下的数据预处理效率的不足进行了分析,在体系结构中引入数据缓存区模块,将数据预处理过程分为两个阶段,从而对数据预处理的效率有了很大的提高,并结合 Microsoft 的 DTS 对其进行了简单应用。

关键词 体系结构,数据缓存区,数据预处理,数据转换服务

Application of the Pretreatment of the Data in the Data Warehouse Architecture

HE Min[1] WANG Wei-Tao[1] HE Guang-Hui[2]

(College of Computer Science[1],College of Mathematics and Physics[2],Chongqing University,Chongqing 400044)

Abstract Data pretreatment efficiency under the original data warehouse architecture of this text is not enough to analyze, introduce the data buffer memory area module in the system structure, divide the pretreatment course of the data into two stages, thus the efficiency to the pretreatment of the data has been improved a lot, combining Microsoft DTS has carried on simple application to it.

Keywords Architecture, Buffer memory area of the data, Pretreatment of the data, Data transformation service

1992年以来,以被誉为"数据仓库之父"的 W. H. Inmon 的《Building the Data Warehouse》一书为标志[1],数据仓库技术以惊人的速度发展,并成为数据库研究的一大热点。数据仓库是一个面向主题的、集成的、时变的、非易失的数据集合,支持管理部门的决策过程[2]。这个简短而又全面的定义指出了数据仓库的四个关键特征:面向主题、集成、时变、非易失,将数据仓库与其他数据存储系统(如关系数据库系统、事务处理系统和文件系统)相区别。正是由于这些特征,道出了数据仓库构建过程中的两个关键环节:1)数据预处理,如数据抽取,数据清理,数据转换和数据装载等。2)支持决策的数据分析,如联机分析处理(OLAP)技术和数据挖掘(DW)技术,将数据转换为有用的信息。

数据仓库作为数据存储和分析的平台,其数据源包括各种异构的可操作的外部数据库和其他的外部数据。数据源中可能包含噪声数据、空缺数据和不一致数据,这对数据分析将会产生不良后果。因此,如何预处理数据从而提高数据质量,对数据分析的结果是否正确有效有着重大的影响,而数据分析的结果直接制约着决策者的决策。另据国外专家估计,数据预处理过程在数据仓库商业应用中占了60%至80%的时间和成本[3]。所以,提高数据预处理的效率,提高数据质量,是数据仓库系统发挥有效作用的重中之重。本文对这方面做了一定的研究。

1 数据仓库的体系结构

数据仓库的数据源非常丰富与复杂,既包括各种异构的操作型数据库,又包括一些非结构化的外部数据,这些数据与数据仓库的数据结构和组织方式可能存在很大的差异,再加上这些数据中可能还包含噪声、空缺、不一致的数据,因此数据源的数据必须经过严格的抽取、清洗和转换,才能加载到数据仓库,为数据分析和数据挖掘服务。而这个预处理的过程十分复杂,通常会耗费比较长的处理时间。而我们通常采用的数据仓库体系结构如图1所示。

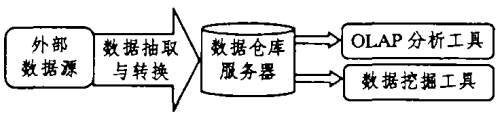

图1 通常数据仓库的体系结构

基于这种体系结构构建数据仓库时,外部数据源通过 ETL 工具的处理直接加载到数据仓库中。这种处理方式存在一些不足之处:1)由于数据预处理本身的复杂性,直接整合必将导致该过程既占用许多外部操作型数据库的资源和时间(而这对实时系统影响很大),也会影响数据仓库装载数据的效率。2)在源数据通过抽取、清理、转换后,向数据仓库进行传输时如发生系统故障或网络故障,就只能全部重做整个数据预处理过程,极大地浪费了资源和时间。

这种情况下,我们只能通过改进算法,提高 ETL 工具软件的效率来节约数据预处理过程所耗费的资源和时间,但这种收效是不显著的。考虑到算法研究中"以空间换时间"的思想,再结合目前硬件成本的下降,我们可以在数据仓库的体系结构中添加一个专门进行数据预处理的存储区域,我们姑且称它为数据缓存区,取其在数据源和数据仓库之间进行缓冲的意思。现在的体系结构如图2所示。

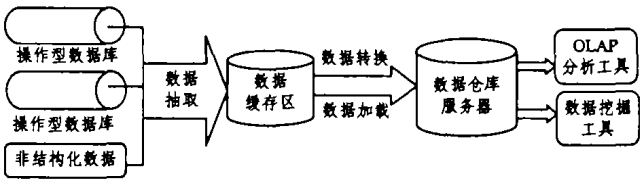

图2 改进的数据仓库体系结构

数据缓存区是为了数据预处理高效顺利进行而引入的阶段性数据存储区域,它是外部数据源进入数据仓库前的缓冲区。数据仓库主题所需要的各个外部数据系统的数据首先直

接快速导入数据缓存区，然后再从数据缓存区经过数据清理、映射和转换等复杂的数据预处理操作加载到数据仓库中。通过引入数据缓存区，我们就把原本复杂的数据预处理过程分解为两个阶段：数据源→数据缓存区→数据仓库。这样一来，我们在第一阶段的数据传输过程中，就可以避免复杂的数据处理，只是简单地针对主题所需进行数据抽取，从而保证数据的快速导入，并且可以大大减小对外部操作型数据系统的压力(这对实时系统显得尤为重要)。在数据仓库系统的运行和使用过程中，数据缓存区的作用还体现在以下几个方面：

·根据经验，跨越网络特别是广域网的数据库操作会大大降低数据处理的效率，而且处理的复杂程度越高，网络对处理效率的影响越严重，数据缓存区的引入可以大大加速数据仓库后台数据预处理过程的实现。

·可以在一定程度上屏蔽外部数据系统的变化对第二阶段的数据预处理过程的影响。即在外部数据系统的数据结构改变时，只需在第一阶段的数据抽取操作时进行适当的数据结构转换，就可以避免对后续过程的修改。

·如在向数据仓库加载数据时发生系统故障，因为有了数据缓存区的存在，就可以避免再从外部数据系统导入数据。

·数据预处理过程分解为两个阶段进行实施，还可以在一定程度上降低ETL工具设计的复杂性。

由上可以看出，在数据仓库的体系结构中引入数据缓存区后，对数据仓库的运行效率有了很大的提高，对基于原有体系结构的数据预处理过程的不足之处也有了很大的改进。

引入数据缓存区后，数据预处理模块可以分为元数据管理、数据缓存区、ETL工具三个部分(这种情况下的数据预处理模块的数据流程如图3所示)。元数据主要描述源和目标数据的详细信息及映射关系，定义数据抽取、转换和加载的一些商业规则和转换规则，以及数据更新的一些控制信息。通过元数据管理，可以实现数据预处理过程的统一、规范，并且符合重用的思想。ETL工具部分我们将在本文的第2节详述。而数据缓存区部分作为数据预处理的主要平台，它的结构模型的选择也相当重要。因为好的数据模型对今后数据抽取和数据转换的实现有着很大的影响。结合目前市场运行的数据库还是以关系数据库为主，其是数据仓库的主要外部数据来源，所以我们在数据缓存区中选择关系数据库作为数据存储的方式，这样与外部操作型数据系统进行数据传输时相对方便一些。另外，现在的关系数据库技术都比较成熟，其DBMS的功能都很强大，利用其本身提供的数据导入导出工具就可以完成大部分的工作，例如Microsoft公司的SQL Server 2000提供的DTS数据转换服务工具就经常作为数据预处理的主要工具。由于DTS是使用OLE DB作为数据访问接口，能够对各种支持OLE DB的主流数据库(如ORACLE，SQL Server，DB2等)之间数据抽取和转换提供良好的解决方案，因此本文在数据预处理的实施部分也是以DTS为工具，对数据抽取和数据转换过程进行分析。

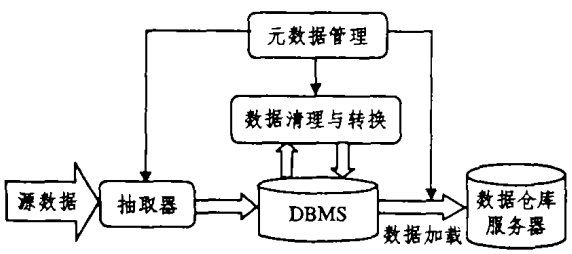

图3 数据预处理模块数据流程图

2 基于数据缓存区的数据预处理的实施

数据预处理有很多方法，可以将它们概括为数据抽取(Data Extraction)、数据转换(Dtat Transformation)和数据加载(Data Loading)，也就是常说的ETL方法。在以数据缓存区作为数据预处理的平台后，我们可以把这三种方法所要实现的功能作如下定义：

·数据抽取：从数据源提取数据存入数据缓存区。为了避免复杂计算，这一步我们可以只对空缺数据进行简单处理，尽量减少对外部实时系统的影响。另外，在抽取时，我们应该对外部数据加上时间戳，以便今后对数据仓库的增量更新。

·数据转换：对导入数据缓存区的数据进行进一步的处理，这些处理包括清理噪声数据、重复数据，对异构数据源和不一致的数据进行转换，以及根据数据仓库的需要对数据进行集成。这一步相当重要也最为复杂，在我们选用关系数据库作为数据缓存区的存储模式后，利用DBMS的强大功能，可以相对减轻此处的工作量。

·数据加载：把处理干净的数据从数据缓存区导入数据仓库的物理存储区域，并清空缓存区的数据，进入等待状态。

有了这些功能的分解和定义，再借助于Microsoft SQL Server 2000提供的数据转换服务工具(DTS)，数据预处理的实现就相对容易了。DTS是不同OLE DB数据源之间移动和转换数据的灵活工具，包括下列特性[4]：

①灵活性。除了处理自然SQL Server对象之外，DTS还可以处理任何OLE DB数据源的数据。这些数据源可以是数据的源和目标。例如，使用DTS将数据从Oracle数据库移到Microsoft Access数据库。

②转换。DTS不仅可以将数据从一个表移到另一个表种，还可以在这个过程中转换数据。这些转换包括查找另一个表中的值以进行涉及多个列的计算和对每行数据运行复杂的VBScript过程。

③脚本。DTS可以利用各种脚本语言进行扩展，包括VBScript，JavaScript与PerlScript。

④工作流。DTS提供的工作流设计器可以将多个DTS任务链接成一个包，任务可以按顺序执行或并行执行，包可以根据任务成功、失败或完成进行决策。

DTS有三种不同用法：最简单的是用数据导入导出向导(Import and Export Wizard)进行简单DTS任务，但灵活性较差；还有DTS包设计器(Pack Designer)，这个图形化设计工具可以生成多个DTS任务并将其链接入完整的工作流应用程序；最后，DTS还有完整的COM接口，可以在任何COM客户端应用程序中利用它的功能。通过COM编程DTS是使用DTS的最复杂的方法，提供了从应用程序中利用DTS的完全灵活性。下面，我们在数据缓存区中选用Microsoft SQL Server 2000作为其DBMS，以DTS作为主要的预处理工具，简要介绍其在数据抽取、转换和加载中的使用。为了方便起见，我们可以假定数据源也是采用的Microsoft SQL Server 2000系统。

从数据源到数据缓存区的数据抽取操作，由于处理比较简单，因此我们采用数据导入向导来完成。数据导入向导以图形界面形象地指导我们一步步往下操作，方便简捷。其主要步骤包括：设置数据源，设置数据目的地，从源表到目的表的映射，将数据移动任务存储为包，再执行包。如果以上步骤无错，其实际移动的执行过程如图4所示。

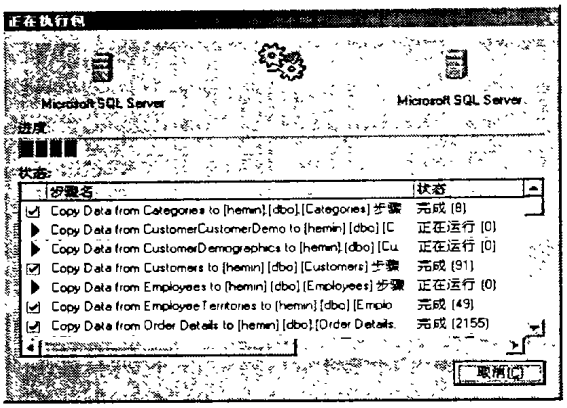

图4 DTS 包执行示意图

数据抽取成功执行后，我们就可以在数据缓存区中对数据进行转换和加载的操作。由于转换操作比较复杂，因此我们采用 DTS 包设计器来实现。包设计器也是图形化界面，与导入向导原理区别不大，主要的不同就在于转换操作可以通过 VBScript 脚本语言编程实现。在图5中选择新建按钮，在弹出的界面中输入用户编写的脚本语言即可。

结论 本文对数据预处理效率的提高方面进行了一定的研究，通过对原有数据仓库的体系结构下进行的预处理的不足的分析，引入了数据缓存区的概念，从而把原本复杂的数据预处理过程分解为两个阶段来实施，在一定程度上解决了原有的不足，提高了数据抽取、转换和加载的效率。但由于能力与时间的问题，对数据预处理在这种体系结构下的具体实现还缺乏更深入的研究，这也将是我们今后努力的方向。

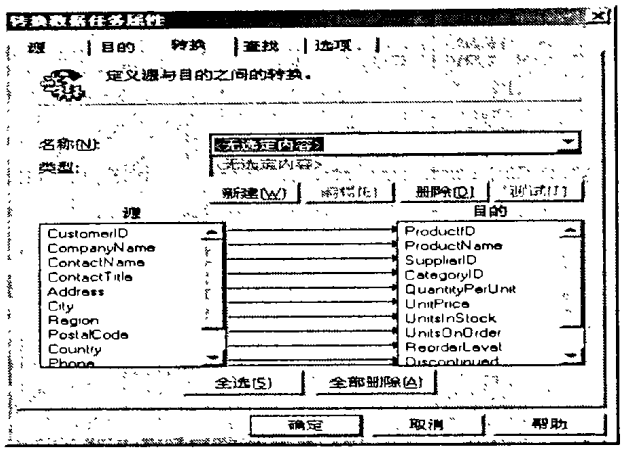

图5 转换数据任务属性

参 考 文 献

1 Inmon W H. Building the data warehouse [M]. Slough: QED Publishine Group, 1996
2 (美)Han Jiawei, Kamber M 著，范明，孟小峰，等译. 数据挖掘概念与技术. 北京：机械工业出版社，2001
3 Pereira E. Slash Business Intelligence Development Time and Costs by 80%. datawarehouse. ittoolbox. com, 2002
4 (美)Gunderloy M, Jorden J L 著，仲潘，等译. SQL Server 2000从入门到精通. 北京：电子工业出版社，2001

(上接第75页)

表2显示的数据相差无几，可以认为是实验中的误差，这种情况也是在预料之中。关于这一点文[2]中已有详细说明。原因是，在中心节点网络磁盘阵列中，重构进程运行于中心节点，本地磁盘和网络磁盘都是固定的，而与故障磁盘的位置无关。中心节点磁盘即为本地磁盘，其他节点磁盘为网络磁盘。如果试图使校验条纹聚集到本地磁盘，则另一部分条纹必然聚集到网络磁盘。当中心节点磁盘发生故障，则重构过程中磁盘访问多数为本地磁盘访问，性能较佳。但如果其他节点磁盘发生故障，重构过程中磁盘访问多数为网络磁盘访问，性能会很差。总体来说，布局优化过程中没有条纹聚集的趋势，条纹是完全散布的。从上表还可以看出，校验单元能够较均匀的分布到各个磁盘上。

(3)分布式磁盘阵列的优化布局结果

这里选取条纹数 $r=117$，条纹长度 $k=10$，磁盘数为 $v=40$，本地磁盘与网络磁盘速度之比设为3:1，即 $e=3$。节点数分别为2,4,5,8。表3列出了重构目标函数值的实验结果及最大的 Pi 值，并且和文[2]中的数据进行了对比。

从表3可以看出使用本文中提到的双目标加权遗传算法得出的校验单元分布非常均匀，重构函数值比模拟退火得到的值要小得多，与加权校验散布的重构负载值有所差别。之所以会产生重构负载值比加权校验散布要差的原因是，本文提出的算法考虑了两个目标函数，而加权校验散布只考虑了一个目标函数。在双目标加权遗传算法只考虑重构负载目标函数时，计算的结果与加权校验散布的结果相差无几。这就说明，双目标加权遗传算法同时兼顾了重构负载值最小和校验散布均匀两个标准，较之于模拟退火和加权校验散布在分布式磁盘阵列系统环境下，更加符合理想布局的标准。

结束语 本文选取理想布局标准中的2和3作为双目标，将用于本地磁盘阵列系统的目标函数进行加权变化，揉合遗传算法的思想，提出了使用双目标加权遗传算法解决网络磁盘阵列系统下的校验散布布局优化的问题，并给出了实验结果。通过以上的理论分析和实际结果可以看出使用多目标遗传算法解决该问题的可行性及有效性。

现在很多人认为“遗传算法、自适应系统、细胞自动机、混沌理论与人工智能一样，都是对今后十年的计算技术有重大影响的关键技术”，多目标遗传算法的应用也已成为人们越来越关注的问题。本文提出的使用双目标加权遗传算法解决网络磁盘阵列系统下的校验散布布局优化的问题，是多目标遗传算法的又一个应用。关于该问题的研究还有很多值得深入和探讨的地方，例如，使用多目标遗传算法解决双故障容错数据布局的问题等。

参 考 文 献

1 董雅莉：[硕士研究生毕业(学位)论文]. 南开大学，2003
2 王刚：[博士研究生毕业(学位)论文]. 南开大学，2002
3 王刚，刘晓光，刘璟. 网络 RAID 布局研究. 计算机科学，2002(5)：11～13
4 Alvarez G A, Burkhard W A, Stockmeyer L J, Cristian F. Declustered Disk Array Architectures with Optimal and Near-Optimal Parallelism. In: Proc. of the 25th Annual ACM/IEEE Intl. Symposium on Computer Architecture, June 1998
5 Schwarz T J E, Steinberg J, Burkhard W A. Permutation Development Data Layout (PDDL) Disk Array Declustering. In: Proc. of the Fifth Intl. Symposium on High-Performance Computer Architecture, 1999. 214～217
6 Merchant A, Yu P. Design and Modeling of Clustered RAID. In: Proc. of the Intl. Symposium on Fault-Tolerant Computing, 1992. 140～149
7 Schwabe E J, Sutherland I M, Holmer B K. Evaluating Approximately Balanced Parity-Declustered Data Layouts for Disk Arrays. Parallel Computing, 1997, 23(4-5): 501～523
8 Deb K, Pratap A, Afarwal S, Meyarivan T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, April 2002, 6(2)
9 Holland M, Gibson G A, Sieworuk D P. Architectures and Algorithms for On-Line Failure Recovery in Redundant Disk Arrays. Journal of Parallel and Distributed Databases 2, 1994
10 Srinivas N, Deb K. Multiobjective function optimization using nondominated sorting genetic algorithms. Evol. Comput., 1995, 2(3): 221～248