OLAP 数据挖掘 一体化模型 数据库 (10)

45-49

计算机科学2000Vol. 27№ 5

OLAP 与数据挖掘一体化模型的研究与发展*)

The Research and Development of a Unified Framework for OLAP and Data Mining

石 磊 石 云

(郑州大学计算机系 郑州 450052) (中国科学院软件研究所 北京 100080)

TP11 TP311.13

Abstract To make the best use of the capabilities of OLAP and data mining tools in the decision analysis, a unified framework should be constructed. It is a promising research field in the knowledge discovery in databases. This paper analyzes the achievements that have been made on such field, and proposes the research direction for the future.

Keywords OLAP, Data mining, OLAM, Influence domain

1 引言

目前,OLAP(联机分析处理)与数据挖掘是信息系统领域内的研究重点。OLAP 作为一种多维分析工具,可提供数据多层面、多角度的逻辑视图[1]。用户提出问题或假设,OLAP 负责提取关于该问题的详细信息,并将结果呈现给用户。数据挖掘是在数据集合中寻找模式的决策支持过程[2],能从大量数据中发现潜在数据模式并作出预测性分析,是现有的人工智能、统计学等成熟技术在特定系统中具体的应用。

数据挖掘与 OLAP 都属于分析型工具,但二者之间有着明显的区别。数据挖掘的分析过程是自动的,用户不必提出确切的问题,只需工具去挖掘隐藏的模式并预测未来的趋势,这样有利于发现未知的事实。而 OLAP 更多地依靠用户输入问题和假设,由于用户先入为主的局限性限制了问题和假设的范围,从而会影响最终的结论。从对数据分析的深度的角度来讲,OLAP 位于较浅的层次,数据挖掘可以发现 OLAP 所不能发现的更为复杂而细致的信息。

在大型数据库和数据仓库的应用中,数据挖掘存在的主要问题是实现相当困难。数据库或数据仓库中存有大量数据和成百上千个属性,由于挖掘分析过程是自动的,用户仅仅指定挖掘任务,而不提供搜索线索,导致搜索空间太大,生成相当多的模式,其中绝大部分可能属于常识或无意义的模式,是用户不感兴趣的。OLAP 分析虽然可给用户提供在不同角度、不同抽象级别的视图,但是由于事先对用户需求的了解可能不十分全面深入,视图中缺乏所应包含的维度,从不同的视图得到的结果可能并不相同,容易产生错误引导,用户需做大量的"数据打捞"工作并参照具体数据才能够猜出正确的结果,而且仍然可能遗漏数据间重要的模式和联系。

从上述分析可以看出,OLAP 与数据挖掘工具各有所长、也各有其缺陷,如果能将二者结合起来,发展一种建立在 OLAP 和数据仓库基础上的新的数据挖掘技术,将更能适合实际的需要。开发一体化模型的原始驱动力有以下几点:

(1)通常数据净化和集成工作占用的时间和精力要超过70%,而数据仓库作为 OLAP 的数据源,存储的是经过净化和集成处理的数据,数据具有较高的质量。

(2)成功的数据挖掘需要对数据进行探索性的分析。挖掘所需的数据可能只是一部分、一定范围的数据,因此对多维数据模型的钻取、旋转等操作同样也可以应用于数据挖掘的过程中,数据挖掘可以建立在多维模型的基础之上。

(3)事先预测挖掘何种类型的知识是困难的。对于用户来讲,常常不知道想挖掘什么样的知识。通过将 OLAP 与多个数据挖掘功能结合,用户可以灵活选择所需的数据挖掘功能,并动态交换数据挖掘任务。

下面将详细介绍目前在统一 OLAP 和数据挖掘理论框架中存在的两种模型:OLAM(联机分析挖掘)模型和影响域模型,并指出实现中需要解决的问题和未来的研究方向。

2 OLAM 模型

加拿大 Simon 大学教授 Han. J. W 等在数据立方体的基础上提出多维数据挖掘的概念[3~6]、其基本操作是将挖掘功能(关联、分类、预测等)与 OLAP 的钻取(drilling)结合。在数据立方体中数据挖掘可在多维

*)本文得到国家"八六三"306主题863-306-ZD-07-4课题的资助。

和多层次的抽象空间中进行，利于灵活地挖掘知识。

OLAM 模型建立在多维数据视图的基础之上，因此，基于数据立方体的挖掘算法是其核心所在。数据立方体的计算与传统挖掘算法的结合使得数据挖掘有了极大的灵活性和交互性，根据立方体的计算和数据挖掘所进行的次序的不同组合可以有以下一些模式：

□**先进行立方体计算，后进行数据挖掘**：在数据挖掘处理开始前，先对多维数据进行一定的立方体计算，以选择合适的数据范围和恰当的抽象级别（粒度级别）。

□**先数据挖掘，再立方体计算**：先对多维数据执行数据挖掘，再通过立方体操作进一步分析挖掘结果。

□**立方体计算与数据挖掘同时进行**：在挖掘过程中，可以根据需要对数据视图做相应的多维操作。这也意味着同一个挖掘算法可以应用于多维数据视图的不同部分。

□**回溯**：允许挖掘处理回溯一步或几步，或回溯至一预置的记号处，然后沿着另外的挖掘途径进行挖掘，以便于完成交互式挖掘。

大型的数据仓库中包含有大量的数据，在数据挖掘过程中提供灵活性是很必要的，这样用户可以遍历整个数据立方体、选择挖掘空间、合适的抽象级别、测试不同的挖掘模块和可替代的挖掘算法。

2.1 OLAM 的体系结构

OLAM 的体系结构如图1所示，整个体系结构分为四个层次：数据存储层、多维数据库（MDDB）层、OLAP/OLAM 层和用户接口层。从图1中可以看到，OLAM 服务器通过用户图形接口接收用户的分析指令，在元数据的指导下，对数据立方体作一定的操作，然后将挖掘分析结果展现给用户，这个过程是动态的。

由于市场上已有许多 OLAP 产品，所以直接在已经建造好的数据立方体和 OLAP 引擎上开发 OLAM 机制是很重要的。通过图1可以看出，数据立方体有两种构造方式：一是存取和（或）集成多个数据库；一是通过支持 OLEDB 或 ODBC 连接的数据库接口将数据仓库进行过滤。

2.2 OLAM 的预期特性

OLAM 目前仍处在起步阶段，有以下应具备的特性：

（1）挖掘任何部分的能力。通过与 OLAP 操作交互，数据挖掘可以在不同的数据上以多个抽象层次进行；也可以在挖掘中执行钻取、切片与分割等 OLAP 操作。多个数据挖掘模块与 OLAP 引擎的交互将保证在数据仓库中的任何部分容易完成挖掘。

（2）支持具有多种性的立方体和支持具有复杂维度与度量的立方体。许多数据挖掘任务需要对具有多特性的立方体进行操作，这是在不同的粒度上包含多个相互依赖的查询的复杂子查询。另外，传统的数据立方体仅仅支持类目数据维和数值型度量。实际上，数据立方体的维度可以是数值型以及空间和多媒体数据。立方体的度量也可以是空间和多媒体聚合或这种对象指针的集合，支持这类非传统的数据立方体将会加强数据挖掘的能力。

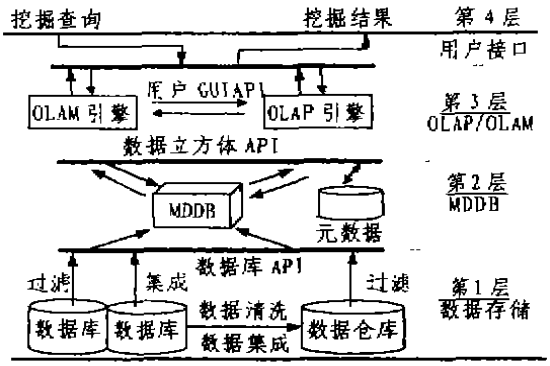

图1 OLAM 体系结构

（3）基于立方体的挖掘方法。该方法应该是 OLAM 挖掘机制的核心。基于立方体的数据挖掘已经有很多研究，包括概念描述、分类、关联、预测、聚类等。基于立方体的挖掘继承了关系型或事务型数据挖掘方法的思想，并具有许多特性，在基于立方体的有效挖掘算法领域需要投入更多的研究。

（4）选择或添加数据挖掘算法。不像关系型查询能用不同的处理对同一查询生成相同的答案，不同的数据挖掘算法可能会生成完全不同的挖掘结果。因此，提供多种可选的数据挖掘算法很重要。另外，如果提供标准开放的 API，并且 OLAM 系统经过很好地模块化，用户就有可能增加或修改数据挖掘算法，用户定义的数据挖掘算法可以较好地利用一些系统构件以及知识可视化工具，并与已有的数据挖掘功能合成。

（5）多个数据挖掘功能之间的交互。OLAM 的长处不仅仅在于选择一系列的数据挖掘功能，也在于在多个数据挖掘和 OLAP 功能之间交互，例如，首先切割立方体的一部分，基于一指定的类属性将该部分分类，并查找关联规则，然后下挖，在更细的粒度上发现关联规则。这样就能够在选定的数据空间任意漫游，用多个挖掘工具挖掘知识。

（6）快速响应和高性能挖掘。OLAM 若想获得快速响应和高的性能，会比 OLAP 困难，因为数据挖掘的计算代价通常比 OLAP 昂贵。快速响应对于交互式挖掘是至关重要的，有时甚至为了得到快速响应而牺

性精度。效率是探索式数据挖掘的主要挑战。由于采用数据立方体技术泛化大量的数据，可以获取较高性能作为响应时间和挖掘粒度的折衷。

(7)可视化工具。为了有效地显示OLAM并与挖掘处理交互，必须开发多种知识和数据可视化工具。图表、曲线、决策树、规则图、立方体视图、boxplot图等是描述数据挖掘结果的有效工具，能帮助用户监测数据挖掘的过程并与挖掘过程交互。

(8)可扩展性。从图1可以看出，OLAM系统在顶端与用户及知识可视化软件包通讯，在底端与数据立方体通讯。它应该高度模块化，并具有可扩展性，因为它可能会与多个子系统合成并以多种方式扩展。应该扩展该技术至高级的和/或特殊用途的数据库系统、包括扩展的关系型、面向对象的、文本、空间、时间、多媒体和异种数据库以及Internet信息系统。对复杂类型的数据，包括结构化、半结构化和非结构化数据的OLAM也是一重要的研究方向。

2.3 实现OLAM机制的讨论

由于数据挖掘功能在计算上的代价比OLAP操作要昂贵许多，在大型数据库或数据仓库中实现OLAM的关键是执行效率的提高和对用户请求的快速响应。必须考虑如下因素：

(1)模块化设计和标准API接口　OLAM系统集成了许多数据挖掘模块、不同种类的数据立方体和可视化工具。因此，高度模块化的设计和标准的API接口对OLAM系统的开发、测试和共享数据挖掘模块很重要。微软和OLAP Council分别提供了OLEDB和MDAPI，将会是数据仓库APIs标准化的重要开端。开发可共享的可视化工具包也很重要，尤其是基于Java的、独立于平台的知识可视化工具。

(2)高性能数据立方体技术　这种技术对OLAP挖掘很重要，由于一个挖掘系统需要计算大量维度之间的关系或详细细节，这样的数据不可能都预先实体化，有必要联机动态计算数据立方体的一部分。另外，多特性数据立方体的有效计算，以及支持具有复杂维度和度量的非传统的数据立方体，对有效的数据挖掘都很重要，因此需进一步开发数据立方体技术。

(3)基于约束的OLAP挖掘　OLAM要求对数据挖掘请求有快速响应，需要采用有效的、基于约束的数据挖掘算法。

(4)逐渐精化数据挖掘质量　可考虑采用下列OLAP挖掘方法：首先在大数据集上用快速挖掘算法标识出感兴趣的模式/区域，然后用代价较高但较精确的算法进行详细分析。

(5)做书签和回溯技术　OLAM借助于数据立方体导航，提供给用户充分的自由，运用任一数据挖掘算法序列来探索和发现知识。当从一个数据挖掘状态转换至另一状态时常常可有很多选择。可做个书签，如果发现一个路径无意义，就回到原先的状态并探索其它的方法，这种做标记和回溯机制可以防止用户"迷失在OLAM空间"中。

(6)建造探索式分析工具和面向应用的语义层　因存在有多个数据挖掘功能，如何在某一具体应用中选定合适的数据挖掘功能是一个问题，必须熟悉应用问题、数据特征以及数据挖掘功能的作用，有时需要执行交互探索式分析来选择合适的功能。因此，建造探索式分析工具以及构建面向应用的语义层是两个重要的解决方案。OLAP挖掘提供探索式分析工具，进一步的研究应该放在为具体应用自动选择数据挖掘功能上。

目前还没有完整的OLAM产品出现，加拿大Simon大学研制的DBMiner系统[7]可以说是OLAM的一个雏型。随着该OLAM模型的不断成熟，可能会有相当多的此类产品出现。

3 影响域模型

上面已经详细地论述了OLAM模型的优点及特性，其不足之处是没有建立起一个统一的模型，只是将数据立方体作为数据挖掘中数据的存储结构和计算基础，没有涵盖问题的全部搜索空间，无法将OLAP与数据挖掘真正有机地结合在一起。OLAP和数据挖掘必须在相同理论的框架下协同工作，该框架无论是在存储结构上，还是在计算模型上，应既不同于OLAP，又不同于数据挖掘。

K. Parsaye在该方面作了一些有益的尝试。他把决策支持空间从应用层次上分成4个子空间[8]：数据空间、聚合(OLAP)空间、影响空间和变化空间。其中，数据空间处理基于关键字(key-based)的决策查询，最典型的是联机事务处理(OLTP)系统；聚合空间对数据空间中数据元素进行聚合运算(如Sum，Average，Max，Min等)，处理有关聚合运算的决策查询，典型的有联机分析处理(OLAP)和多维空间；影响空间处理逻辑性质的决策支持，能够提供比其它空间丰富得多的有用信息，这些信息就是通过数据挖掘而得到的；变化空间负责回答某种变化的过程和速率问题。

K. Parsaye在决策支持四个空间概念的基础上提出影响域模型[9]，对OLAP中数据立方体和星型模式的概念分别进行了拓展，以涵盖问题的整个搜索空间，能够比较全面地反映多维数据挖掘的实质，OLAP挖掘发生在由数据、聚合和影响空间形成的混合空间中，通过SQL引擎和OLAP/ROLAP引擎对数据空间和聚合空间存取。所建立的OLAP挖掘的体系结构如图2所示。其中，关系数据库或数据仓库存储粒度数据，数

据的存储地不一定是数据仓库，也可以是数据矿藏[10]。图2中 MOLAP(线 A)或 ROLAP(线 B)引擎在多个维度上提供对聚合数据的快速存取；多维的发现引擎在多个维度上执行发现，然后合并结果。该引擎通过 SQL 访问粒度数据，通过 OLAP/ROLAP 引擎访问多维数据。这样 OLAP 发现引擎通过数据和聚合空间，将结果合并和结合，对影响域的混合空间提供存取。

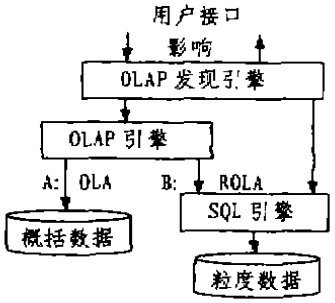

图2 OLAP 挖掘的体系结构

3.1 从立方体到影响域

影响域与多维空间的立方体在逻辑上等价，但立方体上计算的是聚合，而影响域上计算的是蕴涵(implication)。影响域同立方体一样具有属性和值，不同点在于它还具有置信度(confidence)。立方体将维度映射至度量，而影响域将维度和度量映射至置信度。一个影响域可视为一个函数，从立方体和"度量"映射至一置信度级别。

影响域的大小通常比立方体所基于的域要大得多，因为 OLAM 分析常常在更细的粒度上分析更多的维度，或对多个特性之间的关系进行探索。由于每次重新计算的代价太昂贵，所以需要在比星型模式存贮有更多的聚合的模式上进行。为了"遍历"影响域，需要交叉 OLAP 运算与影响性分析。

影响域概念可用面向对象的思想描述，有助于生成一个较好的结构化的框架。影响域的特性包含：基本维度(类)；属性；对象或实例(每个具有一个唯一的标识符)；层次；度量；置信度。其中，基本维度是诸如产品、商店和客户的类。每个类/维度具有一属性集合。如产品维度具有属性价格、颜色等。每个类/维度有对象或要素作为实例，对象的每个属性具有一个值。在类和属性内存在层次，如商标类是产品的父类，而商店通过地区分组、地区按城市分组，城市按省分组，以此类推。概念{地区，城市，省}是一个层次。

星型模式通常直接映射在该对象结构中，每个维表都可看成一个对象，对象的属性代表在维表中的列，度量在各个维度构成的空间上进行计算。

影响域代表在维度集合上的蕴涵运算，影响域中的一个点称为一个坐标或一个实例。影响域代数提供影响域上的操作，正如关系代数提供在库表上的操作一样。可对一个立方体进行投影，即去掉一个维度，投影在去掉的维度上执行聚合；也可进行选择操作以获得另外的立方体，立方体之间的聚合操作也是需要的。

影响域的代数对影响性因子提供操作，就象立方体代数提供 OLAP 操作一样。可以对域进行投影，即丢弃一维。投影操作在丢弃一个维度后。有效地执行新的影响性运算，但是该计算是非平凡的，必须小心进行。

3.2 从星型模式到旋转模式

星型模式的例子见图3，设计星型模式是用来处理聚合运算的，它能很好地用于 OLAP，但它本身不带数据挖掘功能，不能用于 OLAM。为了处理影响性分析，需要将星型模式扩展。

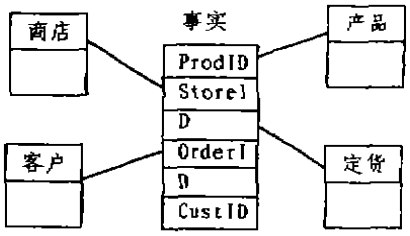

图3 一个星型结构通常直接映射为对象结构

影响性分析中的每一点需要聚焦在星型模式中的维表上(如产品或商店，如图3所示)。对于每个库表来说，需要比星型模式存贮更多的数据，因为在分析中需要用附加的聚合或选择的数据项以丰富维表内容。分析的焦点可以认为是从客户旋转至商店以及产品等等。由于分析的焦点似乎是绕着星型模式旋转，因而将此结构命名为旋转模式。

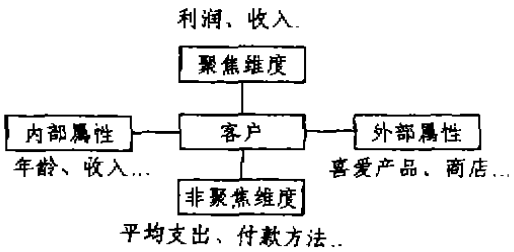

图4 旋转模式的库表具有五个主要部分

如图4中所示，旋转模式中的库表具有五个主要部

分:聚焦维度(如客户);为聚焦而计算的维度(如利润);内部属性(如客户年龄等);外部属性(如某客户最喜爱的产品颜色);非聚焦维度(如某客户平均一次购买的商品的数目)。在执行影响性分析时,焦点沿着不同的基本维度(或类)旋转。

影响域模型的旋转模式不能用于数据仓库,它仅仅用于数据挖掘,因为旋转模式适合于影响空间,但不适合数据空间。可对数据仓库采用稍稍改动的星型模式,对数据挖掘采用旋转模式。目前,基于影响域的模型还没有相应的产品出现。

结论 通过以上分析,可以得出下面的结论:OLAM是OLAP与数据挖掘相结合的产物,它兼有OLAP多维分析的在线性、灵活性和数据挖掘对数据处理的深入性,是数据库(数据仓库)应用工具未来发展的方向。目前,这个领域中的研究工作尚处于起步阶段,还有很多问题需要得到解决,包括技术问题和非技术问题,这不仅给广大研究工作者带来挑战,同时也带来了机遇。

主要参考文献

1 Codd E F, et al. Beyond decision support. Computerworld, 1993, 27(30)

2 Fayyad U M, et al. Advances in knowledge discovery and data mining. California: AAAI/MIT Press, 1996

3 Han J W. Towards on-line analytical mining in large databases. ACM SIGMOD Record, 1998, 27: 97～107

4 Han J W, et al. Issues for On-Line Analytical Mining of Data Warehouses. In: Proc. of 1998 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), Seattle, Washington, June 1998

5 Han J W. OLAP Mining: An Integration of OLAP with Data Mining. In: Proc. 1997 IFIP Conf. on Data Semantics (DS-7), Leysin, Switzerland, Oct. 1997. 1～11

6 Han J W. Conference Tutorial Notes: Integration of Data Mining and Data Warehousing Technologies. In: 1997 Int'l Conf. on Data Engineering (ICDE'97), Birmingham, England, April 1997

7 Han J W, et al. DBMiner: A System for Data Mining in Relational Databases and Data Warehouses. In: Proc. CASCON'97: Meeting of Minds, Toronto, Canada, Nov. 1997

8 Parsaye K. Surveying decision supprot. Database Programming and Design, 1996, 9(4): 27～33

9 Parsaye K. OLAP & Data Mining: Bridging the Gap. Database Programming and Design, 1997, 10(2): 30～37

10 Parsaye K. Data Mines for Data Warehouses. Database Programming and Design, Sept. 1996

(上接第38页)

代码提供部件得到类代码后,首先将其存储在本地的代码服务器中,以备再次使用,然后将类代码返回给类装入器(标号⑤)。在得到所需的类代码后,对象恢复部件就可以恢复对象,提交给系统(标号⑥),之后对象恢复部件被撤消。

总结 实现Agent对象迁移的总的思路是使Java成为适合于可移动软件Agent的编程语言,主要是解决独立运行能力和类在全局范围内的标识问题。在对Java语言的序列化、类装入、RMI技术,以及类代码格式进行深入的研究后,我们将序列化和类装入技术结合,实现了Agent对象的状态和类代码的同步传送,从而在JMSA系统中使Java编写的Agent对象具备了迁移能力。

参考文献

1 陈 羽中,麦中凡. 可移动的软件Agent研究. 计算机科学, 1998(5)

2 陈 羽中,毕 凯,麦中凡. 可移动软件Agent系统代码迁移机制的研究. 1998

3 陈 羽中. 可移动软件Agent系统的研究、设计与实现:[北京航空航天大学硕士生毕业论文]. 1999

4 Gray R, Kotz D. Mobile agents for mobile computing. [Technical Report]. Department of Computer Science Dartmouth College, 1996

5 Knabe F C. Language Support for Mobile Agnets: [PhD thesis]. School of Computer Science, Carnegie Mellon University, 1995

6 Picco G P, et al. Expressing Code Mobility in Mobile UNITY: [Technical Report]. Washington University, 1997

7 McManus C. The basic of Java class loaders. Available at: http// www javaworld. com, October 1996

8 Krumel A. Revolutionary RMI: Dynamic class loading and behavior objects. Available at: http: // www. javaworld. com, December 1998

9 Sun Microsystems Inc. Object Serialization Specification. 1997

10 Sun Microsystems Inc. Java Remote Method Invocation Specification. Revision 1. 50, JDK 1. 2, October 1998

11 Sun Microsystems Inc. Java Platform 1. 2 API Specification, Version 1. 2. 1998

12 Lindholm T, Yellin F. The Java Virtual Machine Specification, Version 2. Sun Microsystems Inc. 1999