数据仓库管理中的若干关键技术

于 戈 张 斌 单吉第 郑怀远
(东北大学计算机科学与工程系 沈阳 110006)

摘 要 Data warehousing technique is a novel and important technique for reusing large-scale, distributed and heterogeneous data sources. Due to the complexity of the system itself and applications to be supported, there exist many open problems to be solved. From the aspect of the system, a data warehouse is also needed a management system, just as a database does. This paper will discuss new problems and related techniques, including system architecture, data translation and transformation, update monitoring, data integration and optimization processing.

关键词 Database, Data warehouse, Data integration.

一、前言

随着信息技术的高度发展,极大地推动了数据库应用的规模、范围和深度。数据库应用已经从点(单台机器,独立系统),线(多台机器,局域网联接的某个部门),发展到面(网络化计算机,多网络联接的企业集成系统),甚至互联网 INTERNET 联接的全球信息系统。在新的形势下,迫切需要对已有数据的重新组织,再加工和再利用,最终提供给高层应用。其中典型的应用包括决策支持(DSS)、在线分析(OLAP)、数据采掘(Data Mining)等。通常,对涉及多数据源操作的可采用多数据库系统(MDBS)技术,即在有已有的数据库系统(可以是分布的和异构的)之上,建立一个集成系统,支持对多库数据的交换与共享。在一个集成系统中,维护和管理对局部数据进行描述的集成模式,并提供集成数据语言,支持对集成数据的定义和操作。在一个多库系统中,一个全局查询的执行主要分为三步:

1)根据集成模式,确定全局查询所要涉及的源数据场地,将查询划分成相应的子查询,并发送到相应场地上。

2)在源场地上,将查询翻译成局部格式表示的局部查询,交付局部场地执行。执行完毕后,将结果转换成全局格式,返回给上层。

3)全局层收集各局部场地发送来的局部查询结果,进行组装和合并,将最终结果返回给用户。

在一个多库系统中,根据局部模式生成的集成模式预先存放在上层的集成目录中,而集成的数据是虚拟的,当需要时才从源场地提取。其优点是可以满足任意的随机查询的需要,保证数据一致性和正确性,节省全局系统的存储空间。但是,多库系统在以下几个方面存在着限制,而影响到它在一些应用中的实用性。

1)可用性方面:当某个源场地发生故障,或者通信网络发生故障时,系统因不能进行分布处理,将导致瘫痪。在某些情况下,源场地不可能直接通过网络在线地联入到集成系统中,只能通过其它方式(如交换软盘)进行数据共享。现在的多库系统不能支持这样的应用环境。

2)系统响应速度方面:因全局查询需要多级转换和通信传输,其延迟和低层系统的效率限制了系统的响应速度。

3)系统性能方面:系统的整体性能取决于源场地中的性能最低的系统,严重影响到系统性能的发挥。

4)系统开销方面:每次查询都要启动多个局部系统,其通信开销和运行开销较大。

因此,人们提出了另一种解决途径—数据仓库技术[1],并受到工业界的广泛重视,已经开发或正在开发一些数据仓库的产品,例如,美国 IBM 公司的 CDF 系统[2],DEC 公司的 RDB/VMS[3],SYBASE 公司的 Warehouse WORKS[4],INFORMIX 公司的数据仓库系统[5]等。但是,这些商用数据仓库产品都假定数据仓库与数据源使用相同的数据模型,通常为关系模型,这样不需要数据结构的变换,并且从源场地到数据仓库的数据采集采用一种离线的批处理方式,即只有自下而上的加载操作,而缺少自上而

下的抽取操作。因此，为了开发满足更复杂要求的通用数据仓库系统，学术界也正在积极开展研究，例如，美国 Maryland 大学的 ADMS 系统[5]，Colorado 大学的 H_2O 系统[6]，Stanford 大学的 WHIPS 计划[7]等。

广义上讲，所谓数据仓库就是一个专门的数据仓储(Repository)，用来保存从多个数据库或其它信息源选取的已有数据，并为上层应用提供统一的用户接口，用以完成数据查询和分析。其优点有：1)查询效率高。因为大多数查询结果可从仓库中直接取得，无须去访问源数据库。这样就节省了全局与局部间的转换代价和通信开销，也不受到局部系统 的速度限制。2)系统可用性好。由于数据都事先从局部系统中抽出，局部系统的故障和性能不会影响到全局系统。3)系统开销小。当重复查询相同的数据时，都可从仓库中取到。与常规集成系统相比，节省了多次访问局部的开销。

数据仓库的主要作用是支持涉及多数据源的全局查询应用。其适用范围包括：1)信息源中的数据的变化是稳定的，或可以预测的，如工程设计中的数据是分阶段、分批产生的。2)应用不需最新的数据，或允许有延迟，如历史档案管理、决策分析、数据采掘等。3)应用要求有较高的查询性能，而降低精度要求，如信息检索系统。

数据仓库系统不同于数据库系统，其主要区别有：1)数据的采集和追加方面，如数据的选取，一致性维护等。2)对专门应用(如数据采掘，工程设计)的支持方面，如需要特殊运算，特殊的数据处理等。本文侧重于讨论数据仓库系统在完成这些特殊功能时，在系统内部的处理上，特别是从数据集成角度看，所需解决的主要问题和可采用的技术。至于在数据仓库之上工具级的功能，例如，对决策支持系统、数据采掘的支持，则超出本文的范围。

二、系统结构

数据仓库系统的基础仍是一个数据库管理系统，通过利用一个专门的数据库管理系统，对数据仓库中的数据进行存储和维护。数据仓库可以是集中的，也可以是分布的，不同之处在于采用集中式 DBMS 还是分布式 DBMS。不失一般性，图 1 给出一个集中式数据仓库系统的系统结构。其中：系统的最低层是数据(信息)源。数据源不仅可以是数据库，还包括非传统数据，如文件，知识库，HTML 文档等。翻译器：负责数据仓库的输入，将数据源中的数据翻译成数据仓库所需的格式。监控器：负责检测数据源中数据的变化，并报告给上级。集成器：负责数据仓库的更新，对局部数据进行过滤、变换、汇总、合并和追加到仓库中。还负责生成仓库数据的模式定义。

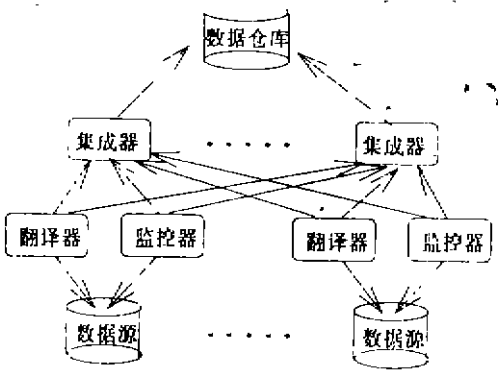

图 1　一个集中式数据仓库系统的系统结构

数据仓库中的数据类似于具体化视图(Materialized View)[8]中的数据。在某些方面，如视图维护技术，有共同之处。但是，它们在本质上是不同的，主要是因为底层数据源的异构性和自治性。因此，数据仓库需采用多级视图机制[9,10]，如图 2 所示。在底层，数据表示方式为局部模型的局部格式，如关系或文件；在中间层，数据表示为公共模型格式，如扩展关系模型或对象模型；在最高层，数据表示为集成模型格式，如关系或对象。因此，视图的具体化过程将分为两级映射：在第一级，数据将从局部数据库中，经过数据翻译，转换并具体化成符合公共模型格式的中间视图。这些具体化视图是临时性的，在生成完上层模式后不必保留。在第二级，经过消除语义冲突、数据集成和数据导出处理，将有关的实体化中间视图集成为满足专门应用的集成视图。

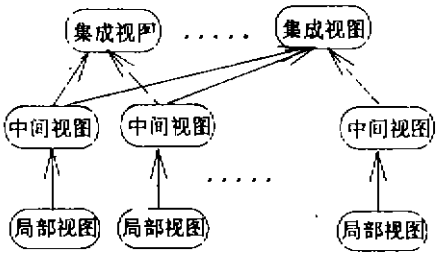

图 2　一个数据仓库的多层视图结构

三、系统主要模块及关键技术

3.1 翻译器

翻译器的数据翻译工作包括数据结构的翻译和数据类型的翻译。例如，若信息源是一个关系数据库，而数据仓库使用面向对象模型时。一般来说，其翻译规则为：关系对应于对象类，每个元组翻译成对象实例，关系的属性值翻译成对象的属性值，并根据元组的关键字值生成对象关键字(标识)。必要时还须对数据之间的语义联系进行重新构造。

关于结构化数据的翻译问题，在多库系统中已作了很多研究。数据翻译中的一个难点是非结构化数据(如文件)的翻译。这即使在多库系统中也没有很好解决，一种方法是将其对象化，利用对象的封装性，实现对非结构化数据的处理。

此外，一个数据仓库所提供的翻译器，不应只局限于固定的几种信息源，特别是对于信息源不能确定的场合。最好是能提供"即插即用"方式，开发出能够自动化或半自动化的翻译器生成技术。一种途径是采用某种互操作性标准，如根据 OMG 组织提出的 CORBA[9] 标准中的 OO 模型和接口语言，定义数据源与翻译器的输入接口，帮助用户将数据源联接到数据仓库的翻译器上。

3.2 监控器

数据仓库提供的是一种"离线"数据，数据仓库中的数据与其源场地的数据在一致性上存在着时间差。从时态数据库角度看，它们为不同的版本。在数据仓库中，根据应用的要求，需规定两者之间的一致性程度，如完全一致，或者部分一致，或者不必考虑。

在强调一致性的情况下，需要建立一种监控机制，通过检测数据源发生的变化，监视数据仓库与数据源之间的数据误差，或者说数据增量，并报告给上层的集成器。实际上，即使不需维护一致性，也可使用监控器，将产生的新数据通告给上层。

监控器的建立与数据源的类型有关。对于不同类型的数据源，应采用不同的监控方式，以提高监控效率，减少监控开销。数据源根据其自治性程度和开放性程度，可划分为以下 3 种类型：

1)全开放型。数据源可以向外界揭示其内部发生的变化。对于这种系统，一种监控方法是从上层向其发送询问，查询是否已发生变化。另一种方法是，在这种系统中定义触发条件，按条件自动地向上通告，例如，按更新操作，或按时间周期。这相当于系统应具有主动数据库的功能。

2)半开放型。系统不提供通告接口，但提供某种参考文档给外界检查，如提供日志记录，差分文件等。

3)全封闭型。系统内部对外部完全封闭，这时只有通过检测和比较全部数据文件的快照才能探知发生的变化。

3.3 集成器

集成器除了负责进行数据仓库初始化和目录管理之外，另一项重要任务是接收监控器的变化通告，并将数据源的新变化，反映到数据仓库中。对数据仓库中数据的更新方式，属于增量更新，即只修改底库中发生变化的数据有关的那些部分。

由于数据仓库中的多视图结构和对专门应用支持的需要，其数据的维护与普通的具体化视图维护相比，存在着许多特殊问题需要解决。

在数据管理方面的主要区别有：

1)数据仓库中，数据的存在方式比常规视图更加复杂。例如，它们不是用同一种数据模型表示。可能底层为关系，而上层为对象。因此，转换算法更为复杂。

2)在多数据源的情况下，需要对多视图进行合并和集成。这时，可能会出现重复的和非一致性数据问题。如同异构式多库系统一样，当同一个数据来自不同的数据源时，它们在数据结构或者在语义上存在着冲突。这就要求集成器具有解决冲突的能力。应采用多库系统中的数据集成技术。这些问题在常规视图系统中是不存在的。

3)数据仓库的数据是历史数据，而不象常规视图，总是当前数据的直接映射。数据仓库应采用时态数据库的历史数据管理技术。

4)根据应用的需要，数据仓库必须对源数据进行再加工，生成更综合的数据。例如，为了支持数据采掘，需提供聚合数据和汇总数据，或更复杂形式的数据。这样，数据仓库需提供复杂的数据变化功能，如聚合、汇总和采样等。虽然具体化视图管理上也提出了类似的功能，但并没有很好地解决。此外，数据仓库还需要有过滤无效数据、填补缺省值、消除不一致性等功能。

在实现方面的主要区别有：

1)在常规视图系统中，视图的处理与基本数据的处理为紧密耦合，它们可以同属于一个事务，这在实现上要容易一些，但在数据仓库系统中，由于源场地的自治性，上层不能对数据源进行控制，它们是松散耦合的，因而难以进行同步控制和一致性控制等。

这就需要采用与视图系统不同的算法，如多库系统的事务管理方法。

2)由于数据仓库中的数据可以允许不与源数据保持严格一致，不必像常规视图系统那样，每当一个数据修改后，其视图则时随之更新。为了提高效率，数据仓库可采用其它高效的方式，如批处理方式。

3)在常规的视图维护算法中，可以自动生成用于维护视图的主动数据库规则。每条规则可当影响视图的某个更新动作而触发，然后对具体化视图进行修改。在数据仓库中，同样可以采用这种"规则驱动"方法。为点击器的每条通告定义一项规则，当底库发生变化时，集成器自动地修改数据仓库中的相应部分。但在数据仓库中，还需要更复杂的操作，如获取额外的新的数据，对数据进行变换等。

4)一个集成器的建立基于数据仓库模型，按照某种应用的要求，从数据源中提取其所需的数据。就像对不同的数据源，需配置不同的翻译器一样，对不同的应用，需要不同的集成器。因此应提供一种高级的非过程化的说明手段，帮助用户建立自己专后的面向主题的集成器。

3.4 优化处理

为了提高数据仓库的效率，需要采用一些专门的优化措施，以节省存储空间，加快响应速度，减少维护开销等。可采用的技术主要有以下几种。

1)过滤源场地上无关的修改。由于源场地上的数据修改操作可能影响到上层的数据仓库中的数据，通常作法是将所有正在执行的修改操作(如插入，删除和更新)都通知给集成器。但实际上，某些操作并不一定导致上层的数据发生变化。例如，假设数据仓库中的数据为关系R的属性A,B的投影。当底库的R上的C属性被修改时，与上层无关。因此，监控器应能过滤掉这些无关的修改操作，减少系统处理开销。

2)可自维护性。当集成器从监控器接收到一个修改通告后，为了在数据仓库中反映其变化，不仅要修改其对应的数据，可能还要访问其它数据。这些附加数据可能可属于一个数据源，也可能属于不同场地上的数据源。例如，假设数据仓库中的数据为关系R和S的联接。当R中增加一个新元组t时，就需要取出S的所有元组同t进行联接。但访问这些附加数据可能因网络通信需要很长的延迟，或因场地故障根本不能实时处理。一种解决方法是在数据仓库中预先保存所有的相关数据，使得所有的计算都可在数据仓库本身进行。这类似于具体化视图中的可自维护性。但在数据量很大的情况下，未必可行。因此，如何确定最小数量的相关数据存储是一项需要考虑的问题，而如何折衷数据预先存储的代价与实时访问数据代价是另一个需要考虑的问题。

3)多视图的优化。为了支持不同的应用，或者不同的功能要求。数据仓库应能提供多个视图。这些视图在数据上可能存在重复部分，即每个具体化视图之间存在冗余数据。为了节省存储空间和降低一致性维护开销，可以只生成一个具体化视图，或少数几个具体化视图，其它视图都可在它们之上导出。但是，需要考虑这些导出视图的查询速度的下降问题。

3.5 其它问题

数据仓库管理系统所涉及的问题远不止上述那些。限于篇幅，这里只简略列出其它几种：

1)数据仓库的维护管理：包括数据仓库设计、初始化和目录管理。

2)数据源的演变：当数据源发生改变时，如模式修改，增加新类型数据，删除旧类型数据时，数据仓库应能适应这些变化，并屏蔽这些变化对上层应用带来的影响。

3)老化数据：数据仓库一般具有保存历史数据的能力，即使数据源中不再保留对应的基础数据。同时，对历史数据的保存应有期限，对于不再使用的数据应从仓库中清除或转储归档。这要求具备一个有效的管理机制。

结束语 在需要利用多个分布的，异构的数据源的场合，数据仓库技术可以作为多数据库系统的一种补充的解决方案。对于某些专门应用环境，数据仓库技术显得更为优越。因为在传统多库系统中，对源数据的请求、处理和合并是在提出查询后进行的，所以其效率难以发挥和提高。而在数据仓库系统中，免去了这些过程，使得数据可直接、快捷地提供出来。

本文讨论了数据仓库的数据管理中出现的新问题及其有关的关键技术。总之，为了构造一个数据仓库管理系统，需要设计数据仓库用的集成说明语言，通告规则语言，监控器接口，翻译器接口，以及生成集成器和修改监控器相关部分的算法等。

目前，数据仓库技术受到计算机厂家的重视，一些产品正在开发和应用中。为了开发一个通用的、灵活的、可伸缩的、高效的数据仓库系统，如本文所述，许多问题还需要进一步深入研究。

参考文献

[1] 王珊、萨立、从数据库到数据仓库，计算机世界，1998年7月15日

OODB 数据模型及查询代数

钟 武 胡守仁

(国防科技大学计算机系 长沙 410073)

摘 要 On the basis of the relational model, relaxing the restruction on that the number of attributes held by every tuple in a relation set must be identical and introducing inheritance and encapsulation mechanisms of OO techniques, this paper describes an OODB model formally by means of establishing Tuple class and Set class. According to the model, 17 query algebra operators which are closure and safety, are defined in the form of predvate calculus.

关键词 OODB model, Query algebra, Predicate calculus.

一、引言

传统的关系模型打破 1NF 限制，已发展成嵌套的关系模型，能更方便地描述复杂对象之间的关系。随着 OO 技术的不断发展，人们也迫切希望将继承和封装机制引入到数据库中，使数据库模型含有更多的语义，能更自然地描述复杂对象间的关系。封装使得元组与在该元组上的操作能结合在一起，元组中的属性值可以是某类复合对象，而该对象又可以是一个元组或一个基于某元组类的集合，从而能自然地描述汇聚这类语义，同时支持语义的相对性。继承使得数据库设计和编程成为可重用(向下继承)，一个元组对象能被其所属类的超类所引用(向上继承)，从而能有效地描述泛化和特化这类语义。继承概念的引入，允许基于某类元组的集合能包含属性数目不等的元组，只要其所属类是该集合基于的元组类的子类。

从上述观点出发，我们建立了一个 OODB 模型，并用谓词演算的形式描述了基于该模型的查询代数。

二、OODB 模型

设 δ 为无限的属性名集，Ω 为系统提供的基本数据类型名(如：integer，real 等)集，O_t 为无限的元组对象集，O_s 为无限的集合对象集。这里 Ω、δ、O_t、O_s 两两不相交。

定义 1 OODB 的模式 Σ 为三元组 $(\Gamma, \leq, \psi)$。其中：①类名集合为 $L \cup \{Object, Set, Tuple\}$，该集合与 Ω、δ、O_t、O_s 不相交。这里，L 为用户定义的类的类名集。②$\leq$ 是一个偏序关系，用于表征类名所对应类之间的继承关系；这里有，$Set \leq Object$，$Tuple \leq Object$。③$\psi: L \to \Gamma$。为类名 $C(\in L)$ 指派一个局部类型 (ST, M)。局部类型 (ST, M) 定义如下：

设类型名集 $H = \Omega \cup L$，则 M 是作用于 ST 上的操作方法集，ST 是如下三种形式的结构：

• ST 为 $\varnothing$。表示无类型结构，此时，方法集 M

[2] IMB Inc., CIM CDF General Information Manual, Oct. 1989

[3] W. H. Inmon and C. Kelley. RdB/WMS Developing the Data Warehouse. QED Publishing Group, Boston, Massachusetts, 1993

[4] Sybase Inc., Sybase Warehouse Works 体系结构, Sybase'95 新技术, 1995

[5] N. Roussopoulos et al., The Maryland ADMS project: views R Us, IEEE Data Eng. Bulletin, 18(2), 1995

[6] G. Zhou et al., Supporting data integration and warehousing using H2O. Same to [5]

[7] J. Hammer et al., The Stanford Data Warehousing Project. Same to [5]

[8] A. Gupta & I. Mumick. Maintenance of materialized views: problems, techniques, and applications. Same to [5]

[9] The Guide to CORBA, OMG TC Document, Sept. 1994

[10] 于戈等，CIMS 信息集成平台多层集成机理的研究，第四届中国 CIMS 学术会议论文集，武汉，1994 年 10 月，pp. 75-76

[11] INFORMIX, Data Warehousing for Enterprise-wide Decision Making, Sept. 1994