数据库 互操作性 数据模型 计算机网络

计算机科学1994 Vol. 21 №. 6

48-51

异种数据库互操作性：概念及面临的问题*)

唐雪飞 熊 萍 刘锦德
（电子科技大学微机所 成都610054）

TP311.13

A

摘 要 数据库互操作性所研究的是如何提供对计算机网络环境中的异种多数据库进行透明访问的方法。本文介绍了数据库互操作性的概念和特征，并与其它数据库访问机制进行了比较。文中还重点分析了数据库互操作性所面临的问题。

1 引言

数据库技术的发展已导致了许多数据库系统的出现。在一台大型计算机上通常存在着多种数据库，以及在计算机网络上也常有数以千计的数据库可供访问。正因为如此，越来越多的用户希望能够同时访问和处理来自几个数据库的数据。这些数据库的一个基本特点就是在物理和逻辑上都存在着差异。物理上可能会涉及不同的数据格式、登录过程、并行控制等方面；逻辑上则可能表现在数据操纵语言(DML)，甚至整个数据模型不同等方面。即使各数据库使用相同的数据模型，也常常会出现语义上的差异，导致不同的人们对同一事实的不同理解。

网络技术的发展为用户访问和处理异种多数据库的数据提供了可能，但由于存在着以上的差异，要变成现实并非易事。有人可能试图基于全局模式的思想，把所有数据库定义成一个逻辑上单一的、集成化的数据库以解决问题。然而，建立一个全局模式通常很困难[8,9]。更一般的途径是，让用户面对异种这个事实，为他们提供处理这些来自异种数据库且并未集成化的数据的能力，这便是本文要讨论的异种数据库的互操作性(以下均称数据库互操作性)。下面，主要从异种数据库互操作性的基本概念和特征、与其它数据访问机制的比较及面临的问题诸方面，对数据库互操作性进行较全面的阐述。

2 数据库互操作性的定义

所谓互操作性，是指当一应用在网络上某节点运行时，可以透明地动用网络上其它节点的数据、处理能力和其它资源[11]。数据库互操作性把此概念扩展到了数据库系统上，是指在计算机网络环境中不仅要求实现用户对多个异种数据库的完全透明的访问(包括查询和更新等)，而且还支持不同数据库系统间的相互动作，即各自数据库系统上的用户在应用工具的支持下，实施对对方数据的诸如排序、筛选、合并、更新等操作。用户在这样的环境下，对多个数据库的使用，不必关心它们的数据模式匹配、数据操纵语言的转换、所处的物理位置等细节，数据库互操作性已经屏蔽了不同数据库在物理上和逻辑上的所有差异。需要明确说明的是，对数据库互操作性的任何讨论，都是基于以网络作为物理支撑的分布式环境，也只有讨论在这种环境下的不同数据库的互操作性才有实际意义。

3 数据库互操作性的特征

数据库互操作性涉及数据访问各个层次的问题，包括多种协议的网络支持、多种数据库访问方法的支持，以及对多种终端用户环境的支持等。数据库互操作性的主要特征包括[1]：

- 透明多数据库访问
- 双向数据库访问
- 数据库特有功能的访问
- 避免资源重复和单线索操作
- 支持多客户平台，使用方便
- 提供有关远程数据库环境的信息

数据库互操作性是开放系统的重要组成部分[11]，其最终目的是提供一种统一的、适合任意数据库的方法，来实现网络环境下对不同数据库的透明访问。所以，并不针对某一特定的数据库或数据访

*)本文得到电子科技大学青年科学基金和电科院基金的资助。唐雪飞 博士生，从事数据库互操作性和开放分布式处理的研究。熊 萍 硕士生，从事数据库互操作性的研究。

问方法。对用户和开发者来说，它遵循这样的原则：屏蔽数据库所处环境间以及各数据库间的差异，通过透明多数据库访问，用户和开发者便可以方便地存取不同数据库的数据，而不必关心数据所在的环境。

实现数据库互操作性必须提供双向数据库访问的能力，即读、写数据库的能力。这一点也正是现有许多数据访问方法难以解决的问题之一，因为对远程数据库更新需要分布式事务处理的支持。一旦提供了双向数据库访问能力，数据库互操作平台便可支持几乎所有的数据库应用，如决策支持系统、管理信息系统、分布式事务处理等。

另外，对用户和开发者，数据库互操作性不仅能提供数据访问操作，还应能执行后端数据库的特有功能，如运行在该 DBMS 上的程序或者一些专用的查询功能。在提供对多个后端数据库的透明访问的同时，数据库互操作平台不应该妨碍用户和开发者使用那些数据库的特有功能。这些功能很可能是用户选择该数据库的一个重要原因。所以，保持对这些特有功能的访问，对数据库互操作性同样是重要的。

对后端异种数据库的操作要求了解并利用后端 DBMS 的一些关键特性，如目录信息和查询优化程序等。一种可供选择的方法是，在前端复制一份目录和查询优化程序，来代替后端数据库的相应机制。这样做可能会提高响应速度（类似高速缓存技术），但实际上，用户要为这些机制付出双倍的代价。因此，从成本上考虑，数据库互操作性应避免资源重复。响应时间是评价软件产品性能的重要指标。数据库互操作平台不仅应对单个用户的复杂查询提供快速响应，而且还能支持多用户的并行查询，允许同时响应对后端数据库的多个访问请求。所以，从性能上考虑，数据库互操作性必须支持多线索操作。

从开放性的角度来讲，数据库互操作平台必须支持广泛使用的客户机平台及其应用间的通信机制。这些平台包括 Microsoft Windows、Macintosh、UNIX/Motif 以及 OS/2 的 Presentation Manager 等。数据库互操作性还应支持丰富的终端用户数据操纵功能，不只是为用户提供向远程数据库提交查询条件的能力，还必须包括交叉列表数据、筛选数据、对结果数据进行分析以及提取新数据等复杂的数据处理能力。

当然，能为用户提供足够的后端数据库信息，以帮助用户了解其操作所引起的后端数据库环境的变化，也是非常有用的。另外，还应为用户提供有关"哪些数据库对他是可获得的"这样一些信息。

4 数据库互操作性与其它数据访问机制的比较

数据库互操作性不同于远程数据访问和分布式数据库管理系统的概念，要解决的不仅是对不同硬件环境下同类数据库的访问，更重要的是能提供对任意数据库资源（即使数据模型截然不同）的完全透明的操作，所强调的是通用性和开放性。而远程数据访问和分布式数据库系统尽管也支持分布于不同网络节点上的数据库的访问，但它们都有一定的针对性和局限性，并不能提供数据库互操作性所能达到的完全透明、开放的环境。

远程数据访问为用户提供一个简单可行的界面工具，允许方便地获取特定的数据库中的数据。目前最流行的是构造查询的图形化工具，用户通过对数据库、表、字段、算术和逻辑操作的选项表的选择来定义查询条件，从而构造查询。通常，远程数据访问要求用户对数据库所存放的位置、与其他数据库间的关系等，都必须有清楚的了解，然后根据用户请求中所提供的这些信息，实施对远程数据库的操作。因此，这种访问对用户是不透明的，用户仍需知道有关远程数据库的诸多信息。同时，远程数据访问只局限于对数据的单向操作，一般只提供了一定方式的查询功能。所以，如果从数据库互操作性的角度来看，远程数据访问实际上只提供对特定数据库最基本的远程访问支持，其出发点也只是作为一种有限的解决用户"燃眉之急"的暂时之策，而非长久之计。当某一单位迫切需要对某种数据库进行访问时，就临时采用一种支持该数据库访问的用户应用程序，实现对远程数据的查询操作。基于以上讨论，远程数据访问不具有数据库互操作所必备的一些重要特征，实际上提供的只是一种基本的对远程数据实施查询操作的机制。

分布式数据库是指逻辑上相关的数据库物理地分布在一个含有某种数据库管理系统的计算机网络上。分布式数据库的定义强调了两点：分布性和逻辑整体性[12]。后者表明，尽管数据库在物理上是分布的，但在逻辑上是互相联系的，是一个整体。而我们所说的数据库互操作，是指对分散在计算机网络节点上存在或不存在内在联系的任意数据库进行访

问，更具有普遍性。分布式数据库还强调场地自治性以及自治场地间的协作性[12]。也就是说，每个场地是独立的数据库系统，可以完成局部数据库应用，具有高度的独立性；同时又相互协作组成一个整体，用户对任何一个场地的数据库的使用，都如同对本地数据库的操作。由于分布透明性，用户不必关心数据的逻辑分片，不必关心物理位置分布的细节，也不必关心局部场地上数据库支持哪一种数据模型。分布式数据库的独立性和分布透明性满足了数据库互操作所追求的透明访问和数据库的自治性。但是，我们也清楚地知道，分布式数据库系统与一般的数据库系统一样，需要专门的分布式数据库管理系统(DDBMS)来对分布的数据进行管理和维护。而数据库互操作性并不是一种数据库系统，也不需要相应的DBMS，它提供的是对不同数据库系统进行数据访问的统一途径。另外，建立一个分布式数据库系统通常采用数据库集成的方法，将局部场地的数据库模式都映射成全局模式。如果参与的各数据库来自同一厂商，这种方式容易做到。但如果参与的各数据库来自不同厂商，建立全局模式相当困难[3,9]，数据库互操作性则是使用户面对多个异种数据库系统，并提供对这些异种数据库的透明访问。

5 数据库互操作性面临的问题

数据库互操作性的灵魂是透明访问，透明的关键是屏弊差异。网络环境下数据库间的差异主要表现在两方面：一是运行数据库系统的软硬件环境的不同，如硬件体系结构、系统软件(如操作系统)和通信系统；二是数据库系统本身的不同，包括DBMS所引起的差异和数据语义所引起的差异(见图1)

为屏蔽这些差异，实现数据库的互操作性，研究者们所面临的问题很多，概括起来，主要涉及以下几个方面：

1)解决数据库运行环境的差异所带来的问题。数据库互操作的实现通常以客户/服务器的体系结构为基础。位于客户端的用户通过一个应用编程界面(APL)发出远程数据请求。接着，将描述被请求数据的语言格式转换成目标数据库系统的格式(不同DBMS厂商使用的数据操纵语言DML可能不尽相同)。然后，把按客户机操作系统格式生成的请求转化成服务器方操作系统所使用的格式。最后还可能要进行不同网络协议的转换，将数据请求传送到服务器方。数据库服务器响应请求后，再按上述过程的逆过程将结果返回用户。除了DML格式转换之外，以上过程实际上是异种机互联互操作要解决的问题[14]。研究与开发者们为此作了大量的工作，但离问题的解决仍存在一定的距离。数据库互操作性的研究者可以借鉴他们的部分成果，但仍有许多问题需要解决。

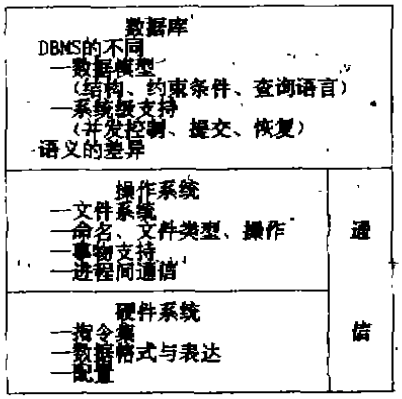

图1

2)解决DBMS的差异所带来的问题：不同的DBMS所引起的差异表现在数据模型的不同和系统调用级的不同。

每个DBMS都有一个数据模型，用于定义数据结构和约束条件。在表达(结构与约束条件)和语言两个方面都可能导致差异。

·结构不同：不同的数据模型提供不同的结构原语。比如，在关系模型中用关系(表)为信息建模，而在CODASYL模型中则用记录类型来建模。如果表达的是相同的信息内容，则处理结构上的不同还较容易。例如，在一种模式中地址被表示成一个实体，而在另一模式中被表示成一个组合式属性。但是，如果表达的信息内容不同，则很难处理这种差别。

·约束条件不同：两个数据模型可能支持不同的约束条件。例如，在CODASYL模式中的系(set)型，可以部分地在关系模式中作为一个参考完整性约束条件来建模。然而，CODASYL支持的insertion和retention约束条件却无法单独由参考完整性约束条件获得。在关系系统中必须用触发器来获得这种语义。

·查询语言不同：不同的语言被用来处理用不同的数据模型表达的数据，即使当两个DBMS支持相同的数据模型，它们的查询语言仍可能存在差

别[16]。

除了数据模型的不同，DBMS 在系统级的差异也是研究者们必须面对和解决的问题。系统级的差异涉及：事务管理的原语和技术（包括并发控制、提交协议和恢复）、硬件和系统软件的需求以及通信能力。

3）解决语义差异所带来的问题。由于网络上各数据库应用系统的开发往往是独立的，对相同或相关数据的意义或解释的不一致是不可避免的。这便会发生语义上的差异。下面的例子可以说明这个问题。

考虑数据库 DB_1 中关系 RESTAURANT 的属性 MEAL_COST，它表示餐馆中每个客人一次进餐的平均费用，但不含服务费和税。而数据库 DB_2 中关系 BOARDING 的同一属性 MEAL_COST 所表示的每个客人一次进餐的平均费用却包含了服务费和税。设两个属性具有相同的语法性质。将 DB_1. RESTAURANT. MEAL_COST 和 DB_2. BOARDING. MEAL_COST 进行比较是没有意义的，因为它们的语义有差异。

检查语义差异相当困难。通常，DBMS 模式没有提供对数据进行一致解释的足够语义。数据模式的不同也增加了识别和解决语义差异的困难程度。

除了以上提到的几个重要问题之外，在具体实现时还可能遇到这样那样的问题，如数据完整性等，研究与开发者们需要付出巨大的努力。

6 结束语

数据库互操作性所要达到的目标很有现实意义，吸引了越来越多的研究者。然而，所面临的问题对研究者们又是一种严峻的挑战。人们正在从各个方面采取各种途径，为最终实现完全可互操作的数据库系统而努力着。由于篇幅所限，关于数据库互操作性的实现方法在另一篇文章中讨论[18]。（参考文献共15篇略）

（上接第47页）

出的混合调度协议中，冲突关系可在前后两个方向上"生长"，即表现为 SSG 可双向生长。假设一个事务 T 已到达 ILP 且拥有一个写前锁，然后一个新事务 τ 到达，并在同一数据项上获得了一个读锁，当 τ 到达 ILP 时，关系 τ→T 建立了。现在即使 T 提交终止了，只要 τ 还活跃着（未到达 FLP），另一事务 τ' 仍可能到来被放于 τ 之前，从而也就在 T 之前。通过这种方式，将来的事务有可能在逻辑上处于过去。虽然已存在的二版本封锁协议如2V2PL 协议也允许一个事务读"前值"，但那些协议没摆脱2PL 的限制，写事务仍要等待读事务的提交，从而无法在两个方向上"生长"。之所以有这么个区别，本质上是由于本协议与2PL、SGT 协议在保证串行化的机制上不同。在本协议中，串行化是通过混合调度实现的，综合了不同机制的特点，由此可生成非2PL 的经历。例如下述非2PL 经历便是本协议允许的：

经历：$R_1(X)\ R_2(Y)\ W_1(Y)\ R_3(Z)\ W_2(Z)$

串行顺序：$T_3 \rightarrow T_2 \rightarrow T_1$。

七、结论

本文给出的混合调度协议尽管比纯协议（2PL 及 SGT）复杂得多，但它还是值得研究的。串行性是经历的一种句法特性，判断一个经历的可串行性是一个 NP-完全问题。所以现有的调度算法均只能产生可串行化经历的一个子集，对此子集的判定是有效的。无论2PL、SGT 及本协议均是如此。但本协议在调度中更宽容些（相对允许更大的并行性），且完全有可能通过对读集、写集提供的信息作进一步的分析来识别允许更多的可串行化经历，进一步提高并行性。

参考文献

[1] Philip A. Bernstein, Vassos Hadzilaco, Nathan Goodman, Concurrency Control And Recovery in Database Systems

[2] P. Dasgupta and Z. M. Kedem, The Five Color Concurrency Control Protocol: Non-Two-Phase Locking in General Databases, ACM TODS, Vol. 15, No. 2, June 1990

[3] 瞿兆荣，分布式数据库，华东计算技术研究所，1989年

[4] K. Vidyasankar, A Non-Two-Phase Locking Protocol for Global Concurrency Control in *Distributed Heterogeneous Database Systems*, IEEE transactions on knowledge and data engineering, Vol. 3, No. 2, June 1991