面向对象 数据库 数据模型 (14)

51-55,34

计算机科学 1994 Vol. 21№. 5

面向对象数据库系统：诺言·现实·前景(2)

Kim, W 张成洪 Won Kim

TP311.13

4. OODB 的前景

今天，RDB 的缺陷和 OODB 的承诺都很清楚了。然而，OODB 在数据库市场还没有多大影响。究其原因，一是当前大多数 OODB 作为数据库系统还不成熟，缺乏许多 RDB 中已有的主要数据库设施；二是不能与 RDB 完全兼容(即不支持 ANSI SQL 的超集)。

工业界和市场界所达成的共识是，面向对象技术的确能给数据库技术带来飞跃，但是必须具备至少三个主要条件之后才能实现其诺言。

第一，吸收面向对象数据模型的新数据库系统必须是与 RDB 兼容的成熟数据库系统(即其数据库语言必须是 SQL 的超集)。

第二，必须为这样的数据库系统提供应用开发工具和数据库访问工具，正如对 RDB 的使用，这些工具很重要一样。工具包括图形应用(表格)生成器、数据库的图形浏览器/编辑器/设计器、图形报告生成器、数据库管理工具以及可能的其它工具。

第三，需要一种过渡途径(桥梁)，允许这样的系统和当前已安装的 RDB 共存，根据不同目的数据库安装可以使用 RDB 和新系统，并且逐步地从当前产品过渡到新产品。

如何才能建立一个与 RDB 完全兼容的面向对象数据库系统？一个 RDB 如何过渡到这样一个新的数据库系统呢？UniSQL 公司有一个商品化的数据库系统 UuiSQL/X，支持一个带有完全面向对象扩充的 ANSI SQL 超集。UniSQL 公司也为 UniSQL/X 的使用提供图形数据库访问工具和应用生成工具。另外，UniSQL 公司提供一个商品化的联邦(多)数据库系统 UniSQL/M，允许 UniSQL/X 与 RDB 共存，同时给用户一种只有单个数据库的幻觉。我将用 UniSQL/X 和 UniSQL/M 来说明这一节的主要概念。

关系和面向对象技术的联合是后关系数据库技术的最主要基础。ORACLE 公司最近宣称计划开发一个 SQL 的面向对象扩充。ANSI SQL3 标准委员会当前正在设计对 SQL2 的面向对象扩充，SQL3 的目标与指导开发 UniSQL/X 数据库语言的目标完全相同，SQL3 大约要 3-4 年时间。另外，HP 的 OpenOOB 支持一种叫做 OSQL 的数据库程序设计语言，它以 SQL 和函数数据模型(而不是关系数据模型)的结合为基础。*Texas Instruments* 也提议并初步实现了一个称为 ZQL[C++]的数据库程序设计语言，用类 SQL 的查询机制扩充了 C++。一些 OODB 卖主也准备开发"类 SQL"语言，通常标上 ObjectSQL，包括了定义和查询面向对象数据库的设施，并作为他们已有的 OODB 的附加成份。这代表了他们在生产策略上的一个大的方向转移，就在几年前，这些卖主还仅仅打算在他们的 OODB 与一些 RDB 之间提供一些信关

4.1 RDB 和 OODB 的联合

联合方式

概括地说，有三种可能的方法将 OODB 和 RDB 结合在一起：信关、RDB 模型(engine)上加 OO 层和统一模型。如果用信关方法，一个 OODB 的请求被简单地加以转换并移交到单个的 RDB，从 RDB 中返回的结果再发送给发出原始请求的用户。对于 RDB 来说信关是作为它的一个普通用户。当前实现的信关对 OODB 的请求附加了各种限制；或者只接受读请求、只单个请求(而不是作为一个事务的请求序列)，或者只是简单请求(即不是 RDB 所能处理的所有类型的查询)。虽然信关方法允许应用程序可以使用从 OODB 和 RDB 中检索的数据，但它不是联合关系和面向对象方法的真正办法。由于翻译请求和返回数据的代价昂贵，且与 RDB 通讯的开销很大，其性能是不可接受的。另外，由于应用程序人员或用户必须知道有两个不同的数据库存在，其可用性难以接受。

如果用 OO 层方法(例如 HP 和 OpenODB)，用户用一个 OODB 数据库语言(OpenODB 中的 ObjectSQL)与系统打交道，OO 层把面向对象的数据

库语言翻译成 RDB 数据库语言，以便与底层的 RDB 打交道。翻译的开销可能很大，况且这种方式本来就会损害性能。例如，OO 层将利用 RDB 可用的界面把对象映射成关系的元组，生成对象的 OID，把它们传递到 RDB 并作为元组的一个属性；它还将利用 RDB 界面把在一个对象中找到的 OID 映射到存储在 RDB 中的对应对象上。一个 RDB 由两层组成：数据管理器层和存储管理器层。前者处理 SQL 语句，后者将数据映射到数据库中。OO 层可以和数据管理器打交道（即通过 SQL 语句与 RDB 对话），也可以和存储管理器层打交道（即通过低级过程调用与 RDB 对话）。数据管理器界面比存储级界面慢得多。（OpenODB 在其 OO 层和底层 RDB 之间使用数据管理器界面）。由于这种方法并不打算修改底层 RDB 以更好地适应 OO 层的需要，所以当需要支持复杂的数据库设施时，会招致严重的性能和操作问题。例如，如果在类层次中有大量类必须加锁（比如为了支持动态模式进化），OO 层就必须要么要求一次锁一个类（招致性能的不良后果和死锁危险），因为 RDB 没有提供自动加锁一个类层次（一般指在一个命令中）的功能；要么对底层 RDB 的一次调用就得对整个数据库加锁（即禁止任何其他用户访问数据库的任何部分）。无论哪种选择都不好。另外，如果 OO 层准备支持在应用的事务提交（完成）时修改内存对象并自动对数据库刷新修改过的对象，那么单个对象必须用 RDB 界面一次一个地插入回数据库。

OO 层方法的合理之处是能够把 OO 层放置在各种现有的 RDB 之上；这个灵活性是以性能为代价换来的。OO 层方法是使多种数据库在应用程序看来就好像是单个数据库一样的数据库系统的基础。这样的数据库系统一般称为“多数据库系统”。OO 层方法可以用作多数据库系统的基础，应用程序可以从 OODB 和 RDB 中检索数据。我注意到 OpenODB 当前不是一个多数据库系统，其 OO 层只能连接一个 RDB。后面我将更详细地讨论多数据库系统。

联合的方法使 OO 层和 RDB 融合成单个层，同时要对 RDB 的存储管理器层和数据管理器层进行所有必要的改变。数据库系统必须完全支持数据库语言所允许的所有功能，包括动态模式进化、自动查询优化、自动查询处理、存取方法（含 B+ 树索引、可扩充的散列，外部排序）、并发控制、从软件和硬件崩溃中恢复、事务管理、以及授予和回收权限。联合之后的数据模型的丰富性增加了实现的难度。

统一数据模型

一个关系数据库由一个关系（表）集合组成，而一个关系又由行（元组）和列组成。关系中的一个行/列项可以有单个值，并且这个值属于一个系统定义的数据类型对应的集合（例如整数、字符串、浮点数、日期、时间、货币）。用户可以对这些值施加更进一步的限制，称为完整性约束（例如一个雇员年龄的整数值可以限制在 18 到 65 之间）。用户可以在一个关系上进行非过程性查询，只检索关系中那些其各列的值满足用户说明的条件的元组。另外，用户可以发出一个对关系中用户指定列的值进行比较的连接查询来把两个或多个关系关联起来。

```
1. CREATE TABLE Employee
     (Name CHAR(20),Job CHAR(20),Salary FLOAT,Hobby CHAR(20),Manager CHAR(20));

2. CREATE TABLE Employee
     (Name CHAR(20),Job CHAR(20),Salary FLOAT,HOBBY Activity,Manager Employee);

       CREATE TABLE Activity
         (Name CHAR(20),Numplayers INTEGER,Origin CHAR(20));

3. CREATE TABLE Employee
     (Name CHAR(20),Job CHAR(20),Salary FLOAT,HOBBY Activity,Manager Employee)
     PROCEDURE RetirementBenefits FLOAT;

4. CREATE TABLE Employee
     (Job CHAR(20),Salary FLOAT,HOBBY Activity,Manager Employee)
     PROCEDURE RetirementBenefits FLOAT
     AS CHILD OF Person;

       CREATE TABLD Person
         (Name CHAR(20),SSN CHAR(9),Age INTEGER);
```

图 3　对关系模型的成功扩充

UniSQL/X 在三方面推广并扩充了这个简单的数据模型，每一方面都反映了一个主要的面向对象概念。面向对象的系统或程序设计语言的一个基本观念是一个对象的值也是一个对象。UniSQL/X 的第一个扩充就反映了这点，允许关系的列的值是一个任意的用户定义的关系的元组，而不仅仅是系统定义的数据类型(数、字符串等)的一个元素。这意味着用户可以说明一个任意的用户定义的关系作为一个关系的列的域。图 3 中的第一个 CREATE、TABLE 语句例示了关系模型中一个 Employee 关系的说明，其 Hobby 和 Manager 列的值被限制成字符串。图 3 中第二个 CREATE TABLE 反映了对一个关系的列的数据类型扩充，Hobby 列的值不再需要限制到字符串，现在可以是一个用户定义的关系 Activity 的元组。类似地，表 Employee 的 Manager 属性的数据类型甚至可以是 Employee 关系本身。

允许关系的列可以是另一关系的元组(即任意类型的数据)直接导致嵌套关系；也就是说，一个关系的行/列项的值现在可以是另一关系的元组，而且其值又可以是另一关系的元组，如此循环往复。在图 1 中我们已经看到这个简单的概念扩充在数据检索时如何产生重大的性能改善，这也使得数据库系统有潜力支持这样一些应用，如多媒体系统(管理图像、声音、图形、正文数据、以及含有这样数据的复杂文档)，科学数据处理系统(操纵向量、矩阵等)，工程和设计系统(处理复杂嵌套对象)，等等。这为当今程序设计语言和数据库系统中所支持的数据类型铺平了道路。

UniSQL/X 的第二个扩充是面向对象的"封装"概念，即把数据和操作数据的程序(过程)组合在一起，允许用户把过程加到一个关系上并使过程对每个元组的列值进行操作。图 3 中的第三个 CREATE TABLE 语句表明了 PROCEDURE 子句用来说明一个过程 RetirementBenefits，为任意给定的雇员计算退休金并返回一个浮点数值。隐含地，系统为每个关系提供了读和修改每个列值的过程。

一个关系现在封装了其元组的状态和行为；状态是列值的集合，行为是操作列值的过程的集合。用户可以编写任何过程并把它连到关系上去操作关系的任何一个或一些元组的值。过程的应用差不多是没有限制的。

UniSQL/X 的第三个扩充是面向对象的"继承层次"概念。UniSQL/X 允许用户把数据库中的所有关系组织成一个层次，使得在一组关系 P 和 C 之间，如果 C 会得到(继承)所有在 P 中定义而没在 C 中定义的列和过程，则 P 是 C 的父亲。另外，它允许一个表有多个父关系，可以从中获得列和过程。子关系能从一些父关系中继承列和过程(这叫多继承)。关系层次是一个有单个系统定义的有根有向无圈图(而不是一颗树)，还有，IS-A(泛化和特化)关系在子关系和其父关系之间成立。在图 3 的第四个 CREATE TABLE 中，Employee 关系定义为另一个用户定义的关系 Person 的子关系。Employee 关系自动继承 Person 关系的三个列，即 Employee 关系将有 Name、SSN 和 Age 列，即使在其定义中并未说明它们。

关系层次较之大量独立(不相关)关系的简单组合的传统关系模型，有两个优点：一是用户可以创建一个新关系作为一个或多个已有关系的子关系，新关系继承(重用)已有关系及它们的祖先关系中说明的所有列和过程。二是系统能在一组关系之间确保 IS-A 关系。RDB 要求用户管理和保证这个关系。

现在，让我们改变关系的术语如下。改"关系"为"类"，"关系的元组"为"类的实例"，"列"为"属性"，"过程"为"方法"，"关系层次"为"类层次"，"子关系"为"子类"，"父关系"为"超类"。上述的 UniSQL/X 数据模型是一个面向对象数据模型！一个面向对象数据模型能通过扩充关系模型来获得。如果数据模型是通过对传统关系数据模型进行上述三个扩充来获得的，则术语"面向对象数据模型"、"扩充的关系数据模型"以及"统一的关系和面向对象数据模型(简称为统一的)"变成了同义词。但是，一个扩充的关系模型(系统)如果没有包括所有的三个扩充，就不是一个面向对象模型(系统)，另外，值得注意的是，基于这样一个模型的数据库系统由于其关系基础，可以通过改进过去二十年间已经开发的关系数据库技术的所有理论基础而建立。

尽管三个扩充中的每一个单独看起来似乎都不太重要，但相对于应用数据建模的容易程度和/或随后在查询性能上的提高来说，不论是单独的还是合起来的扩充，其影响都是重大的。嵌套关系的扩充不再需要 RDB 用户所使用的笨重的变通办法。过程和关系层次的扩充在应用数据建模和应用程序设计方面提供了一些重要的新的可能性。另外，嵌套关系和关系层次的扩充反映了 OOPL 的强有力的数据类型设施。

查询和数据操纵

当然，只定义一个允许用户表示复杂数据要求

的数据模型是不够的。一旦数据库模式用数据定义设施定义好之后，数据库中就可能放置大量的用户定义的对象。当用户能有效地检索和修改数据库的一小部分时，数据库系统的能力才会体现出来。考虑到这点，一个数据库系统须提供查询和数据操纵（插入、修改、删除）设施。

UniSQL/X 查询语言不象一般的"类 SQL"对象查询语言，而是 ANSI SQL 的一个超集，因此如果从语法上把扩充去掉，就退化为 ANSI SQL。我这里说的"类 SQL"语言是指，或者是 SQL 的一个子集，或者不支持 SQL 的同样语义的数据库语言。作为 SQL 的一个子集的类 SQL 语言是指那种在 WHERE 子句中不支持嵌套子查询或者在 SELECT 子句中不支持聚集功能等的语言；也指那种不包括诸如定义和使用视图的设施、或动态地改变数据库模式的设施、或在一个类的属性上说明 UNIQUE 和 NULL 约束的设施、或授予和回收权限的设施等等的语言。不支持 SQL 的相同语义的类 SQL 数据库语言指的是，那种诸如处理与 SQL 不一样的 NULL 值、或毫无理由地在接受用户的读和更新请求后拒绝提交事务、或引入在 SQL 中没有的限制（例如当尚有对象属于一个类时不允许 DROP CLASS 命令删除这个类，而 SQL 中的 DROP TABLE 命令是不管有无元组都可删除一个表及其所有元组）等等的语言。

如果一组类象传统关系数据库中的关系那样来定义，UniSQL/X 查询语言的用户就可以发出所有的 ANSI SQL 查询，包括连接和嵌套子查询、成组和排序结果的查询、视图上的查询。让我们考虑两个利用图 4 的简单例子。在图中，类 Employee 定义为类 Person 的一个子类，类 Activity 是类 Employee 的属性 Hobby 的域。第一个查询找出所有挣钱超过 50000 且年龄超过 30 岁的雇员，再根据工种输出所有这样的雇员的平均工资。第二个查询是一个连接查询，找出那些挣钱超过他们的经理的所有雇员的名字。

```
SELECT Job,Avg(Salary)
FROM Employee
WHERE Salary>50000 AND
      Age>30
GROUP BY Job;
```

```
SELECT Employee.Name
FROM Employee
WHERE Employee.Salary > Employee.Manager.
      Salary;
```

UniSQL/X 查询语言也允许构成一些统一的数据模型所需要的额外查询类型（即在关系模型下不适用的查询）。统一的数据模型更丰富，因此它产生了在 RDB 中没有的查询表达式。具体地讲，它允许路径查询，即嵌套类上的查询；包含方法作为搜索条件的成分的查询；返回嵌套对象的查询；类层次中的一组类上的查询。

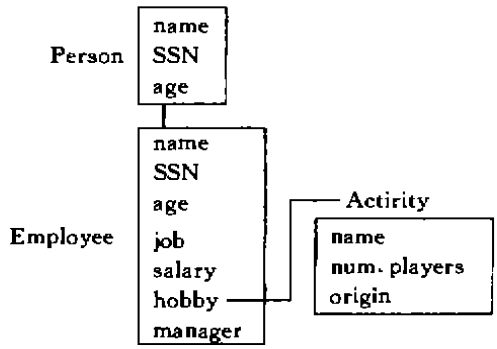

图 4　一个示例数据库模式

（细实线代表嵌套属性；粗实线代表继承路径）

在一个类层次上的查询的一个例子是从一个类和它的所有子类中检索实例。在下面的查询中，关键字 ALL 使得查询是在类 Person 和它的子类 Employee 上计值。

```
SELECT Name,SSN
FROM ALL Person
WHERE age>50;
```

再看图 4，检索嵌套对象的路径查询的一个例子是"找出那些挣钱超过 $50,000 且爱好乒乓球的所有雇员及他们的雇主的名字"，这个查询是在用类 Employee 和 Activity 定义的嵌套对象上计值。查询的构成是把谓词（Name='tennis'）和类 Activity 联系起来，把谓词'Salary>50000'和类 Employee 联系起来。查询从嵌套的 Employee 对象中返回满足查询条件的 Employee 的所有属性。

```
SELECT *
FROM Employee
WHERE Salary>50000 AND
      Hobby.Name="Tennis";
```

谓词（Hobby.Name="tennis"）中的圆点符号扩充了标准的谓词表达式，以表示在任意数据类型的使用过程中的属性嵌套。

支持对象导航

和一些设计来使 OOPL 对象持久化的 OODB 一样，UniSQL/X 提供工作空间管理设施来自动管理内存中的大量对象（称做工作空间或对象缓冲池）。具体地讲，UniSQL/X 在数据库格式和内存格式之间自动转换对象的存储格式，当对象从数据库

装进内存时自动地将存储在对象中的 OID 转换成内存指针，当修改对象的事务完成之后自动将内存中修改过的对象刷新(写)到数据库中去。

在 UniSQL/X 中的这些工作空间管理设施使得数据库应用程序可以通过内存指针追踪来导航驻留内存的对象，可以将单个对象上的变化传播到数据库中。RDB 应用必须借助于显式查询(或连接两个关系或至少要搜索单个关系)来模仿从一个对象到另一个相关对象的简单导航。另外，RDB 应用还必须通过 RDB 界面(数据管理器层或者存储管理器层)一次一个地将修改过的元组传播到数据库中。当一个事务完成之后，UniSQL/X 自动将该事务产生或更新过的所有对象发送到数据库中使之永久化。UniSQL/X 应用程序不需要做任何事情来将变化传播到数据库中去。

我注意到，UniSQL/X 不象大多数也提供工作空间管理设施的 OODB，它支持完全的查询设施和完全的动态模式进化。由于在任何一个时刻，一个对象都可能既在数据库中又在工作空间中，且工作空间中的"拷贝"可能已被更新过了，所以在查询计值时，必须针对已被装入工作空间的那些对象在该工作空间中的"拷贝"，也必须针对未装入工作空间的那些对象的数据库对象。另外，如果用户进行模式修改(例如删除一个类的属性，或者为一个类增加一个属性)，工作空间中的对象"拷贝"将变得无效。UniSQL/X 在支持自动查询处理和动态模式进化时充分考虑了这些问题。

还有，工作空间管理设施是对象持久化和支持用 OOPL 编写的应用程序在对象导航方面的性能要求必不可少的。虽然 UniSQL/X 并没有与任何具体的 OOPL 结盟，但它所提供的成熟工作空间管理设施意味着在 UniSQL/X 之上可以实现一个相当简单的翻译层来支持任何具体的 OOPL(例如 C++ 或 Smalltalk)。

5、与 RDB 的互操作性

对于信关方法，我讨论过把它作为联合 OODB 和 RDB 的一种(不满意的)选择，可以起另一个作用。它允许 OODB 和 RDB 共存，一个应用程序可以同时使用从一个 OODB 和一个或多个 RDB 中获得的数据。但正如我已经提到过的，当前的 OODB-RDB 信关一般只能把请求传递到一个 RDB(例如到 Sybase 或到 ORACLE)，而且不能把对 OODB 的请求和对 RDB 的请求分开来作为单个事务(即作为处理单位的一组请求)进行处理。

多数据库系统(MDBS)逻辑上是信关的充分推广。MDBS 实际上是一个控制多个信关的数据库系统。它没有自己的数据库，只是通过信关管理远程数据库，每个远程数据库有一个信关，MDBS 把多个远程数据库作为单个"虚"数据库呈现给用户。由于 MDBS 没有自己的"真实"数据库，某些数据库设施，如那些管理存取方法(创建和删除 B^+ 树索引、可扩充的散列表等)和参量化性能调整的设施，变得毫无意义。

然而，MDBS 是几乎羽毛丰满的数据库系统，它必须提供数据定义设施，可以在远程数据库的基础上定义虚数据库。数据定义设施需要包括调和(统一)不同远程数据库中数据语义上等价的不同表示的一些措施。MDBS 用户可以查询虚数据库的定义，查询和更新虚数据库(需要查询优化和查询处理机制)。多个 MDBS 用户可以同时查询、更新甚至放置对象到"虚"数据库中(需要并发控制机制)；用户可以把一组查询和更新作为虚数据库上的单个事务(需要事务管理机制)；用户将向其它用户授予和回收数据库各部分上的权限(需要权限机制)。

为了把 MDBS 的查询和更新翻译成远程数据库系统能处理的等价的查询和更新，MDBS 需要适合于远程数据库系统的信关。MDBS 中的信关通常称为"策动器"，远程数据库系统称为"局部"数据库系统，MDBS 呈现给用户的单个虚拟数据库称为"全局"数据库。另外，我们说 MDBS 把多个局部数据库"连接"成单个全局数据库。

UniSQL/M 是 UniSQL 公司的一个多数据库系统，将多个 UniSQL/X 数据库和多个关系数据库连接在一起。UniSQL/M 是 UniSQL/X 增加了存取外部的关系数据库和 UniSQL/X 数据库之后的产物；因此，它是一个象样的数据库系统，而且 UniSQL/M 用户能用 SQL/X 数据库语言查询和更新全局数据库。UniSQL/M 将全局数据库作为定义在局部 RDB 的关系和局部 UniSQL/X 数据库的类之上的一组视图来维护的。UniSQL/M 也维护那些集成为全局数据库的局部数据库的关系和类、它们的属性和数据类型、以及方法等的目录。利用目录中的信息，UniSQL/M 将查询和更新翻译成等价的查询和更新，由局部数据库系统来处理，管理那些查询和更新需要访问的数据。局部数据库的策动器将翻译过的查询和更新传递到局部数据库系统，再把结果传递回UniSQL/M进行格式转换、合并、以及任何

(下转第 34 页)

(3)式可用 Karmarkar 方法[5,6]在多项式时间内求解。

3.4 求解满足条件(Ⅰ)—(Ⅳ)的连接权矩阵 W^*。

由3.2节和3.3节的讨论,求满足(Ⅰ)—(Ⅳ)的权矩阵 W^*,等价于求下述线性规划:

$$\begin{cases} \min \sum_{i=1}^{N} w_{ji} \\ \text{subject to} \begin{cases} Sw \geqslant b \\ (S+H+D)w \geqslant \varepsilon \\ (S+D)w \geqslant c \end{cases} \end{cases} \tag{4}$$

的解 w^*。

当不等式组:

$$\begin{cases} Sw \geqslant b \\ (S+H)w \geqslant \varepsilon \\ w_{ji} \leqslant 0 \end{cases}$$

有解时,根据3.3节注,也可通过求解下述线性规划:

$$\begin{cases} \min \sum_{i=1}^{N} w_{ji} \\ \text{subject to} \begin{cases} Sw \geqslant b \\ (S+H)w \geqslant \varepsilon \end{cases} \end{cases} \tag{5}$$

来确定 W^*。

参考文献

[1]杨行俊,郑君里,人工神经元网络,高教出版社(1990)

[2]张承福,王心强等,几种联想记忆神经网络模型的分析,人工智能与模式识别,3卷1期(1990),p14-21

[3]Martin Grotschel et al., Geometric Algorithms and Combinational Optimization, Springer Verlag 1990.

[4]Bo Zhang, Ling Zhang, Fuchao Wu, Programming Based Learning Algorithm of Neural Networks with Self-feedback Connections, IEEE Trans, Pattern Anal. Machine Intell. To appear.

[5]N. Karmarkar, A New Polynomial-time Algorithm for Linear Programming, Combinatorica 4(1984), 373-395

[6]G. Strang, Introduction to Applied Mathematics, Wellesley-Cambridge Prass(1986)

(上接第55页)

必要的后处理(例如排序、成组、连接)。另外,UniSQL/M支持局部数据库之上的"分布式事务管理",这意味着一个 UniSQL/M 事务中发出的所有更新即使会产生对多个局部数据库的更新,但也能同时被提交或夭折。

当今,RDB 卖主提供不同复杂程度的信关。有些信关允许 SQL 查询被传递到层次数据库系统(即IMS)或者文件系统(如 DEC 的 RMS)。有些信关当前正被提升为能接受查询和更新、甚至支持局部数据库上的分布式事务管理。但是,这些信关中还没有一个能将 SQL 查询传递到 OODB 的;也很少有开发这样的信关的要求。

UniSQL/M 在三个主要方面不同于 RDB 卖主和 OODB 卖主当前提供的信关。

——UniSQL/M 是一个象样的数据库系统,而不仅仅是信关,它支持查询、更新、授权、以及在全局数据库(关于定义在局部数据库的表和类上的视图说明,还有局部数据库的表和类的信息目录)上的事务管理,当前的大多数信关不接受更新。

——UniSQL/M 为处理单个 UniSQL/M 事务要连到局部数据库上,并协调多个局部数据库的查询和更新;具体来说,它支持局部数据库上的分布式事务管理。当前大多数信关只是将请求传递给一个局部数据库,或者当它们支持多个局部数据库时,也不允许在单个事务中同时对多个局部数据库进行更新。

——UniSQL/M 还提供了一个比当前任何信关都更强有力的设施。尽管不够充分(由于理论限制),UniSQL/M 还是将局部 RDB 扩充成了 UniSQL/X;也就是说,UniSQL/M 能对局部关系数据库中所检索到的元组增加对象标识并允许用户为其加上方法,将它们转换成对象。用这个方法,UniSQL/M 使得 UniSQL/X 中提供的主要的面向对象设施对局部 RDB 也可用,具体包括 SQL/X 的路径查询、方法、以及 UniSQL/M 内存对象的工作空间管理。

UniSQL/M 至少在三个不同的环境下是可用的。首先,可以用于允许 UniSQL/X 和 RDB 的共存。其次,可以用于将一组 RDB(或一组 UniSQL/X)变成一个分布式数据库系统。最后,当与单个 RDB 打交道时,它可以作为 RDB 模型上的对象管理层,把 RDB 变成 UniSQL/X。(全文完)

[张成洪译自《Proceedings of the 19th VLDB Conference》,Dublin,Ireland 1993,施伯乐校]