具有部分知识的分布式数据库的基础

Mark Blakey

摘 要

本文提出了一种新型的带副本的分布式数据库。该类数据库在由处理站点组成的超大型网络上提供了高度分布透明性。这种分布式数据库有别于其它分布式数据库是因为在它的处理站点中只具有系统中数据对象和站点的有限而不是全部的知识。一种较为复杂的知识模型取代了传统的数据目录。文中还提出了能表征此类DDB基本特性的公理框架，从这种框架结构能推出一种拓扑网络模型来作为知识模型的基础。作者认识到了支持自主子域的实际重要性，并在其网络拓扑模型中提供了这种自主子域。给出一种命题演算以简化数据的物理定位的推理。提出的一系列启发式方法，能使查询一个数据对象位置所需的搜索工作量最小。通过描述该类DDB的主要操作过程，数据定位算法，论证了所提出的拓扑模型和启发式搜索的优点。

至今大多数分布式数据库系统只包括比较少的工作站点。然而，随着通讯支持和计算技术的日益完善，新的含有大量合作站点（如几百个）的分布式数据库应用将会出现。这方面的例子包括：新的远程通讯服务，如：联机电子目录检索或公用存取数据库；新的增值电讯服务，如：自动呼叫改道，也可以通过在每一个电话转换机中结合一个DDB的站点来实现。其它行业也可能使大规模的分布式数据库的应用得到发展，其中包括在金融机构、旅游和房地产中的应用。

对于这些大系统的成功，数据目录具有决定作用。但是，还存在一些固有的问题，比如：如何决定哪些站点应拥有目录版本，这些版本应如何完备。例如，假定全局目录存放在一个被所有站点所知的特殊站点上，这种设置主要有两点不足：(1) 由于为查找所需的数据远程访问，所有与数据库的交互操作将引起延迟和费用增加。(2) 一旦存贮的站点或站点的通讯连接遭到失败，将使得这种交互操作无效，从而降低系统的可靠性。另一种可能的设置是在每一个站点上有一目录副本。但是，这样可能在小容量的站点上加重不合理的存贮负担，且当数据需要再定位或一个站点加入（离开）系统时，所有的目录副本都需要修改，从而导致严重的网络拥塞。因此这种设置就存在网络拥塞的危险性。况且响应延迟也是人们通常所不能接受的。访问所有的目录还会发生矛盾，即用户知道某个数据对象的位置，但又不知道它是否存在（即对象存在的信息是受存取控制支配的）。

本文提出并探讨了解决这些操作问题的一种方法：对任何站点，可得到的目录信息仅限于本站点的数据对象，以及经过很好选择的其它站点子集所拥有的数据对象。关于其它网络信息，甚至关于站点的存在信息对任意站点都不是可直接得到的。在此所介绍的这种系统，我们称之为具有部分知识的分布式数据库PIDDB(Partially Informed Distributed Database)。以这样的方式分割目录能使目录分布存放并克服以上所谈到的困

难。各个站点可以拥有全局目录的不同子集。目录的某些部分在几个站点上还可以重复。PIDDB为广泛的信息检索和事务处理应用提供了概念框架结构。本文的主要目的在于建立这种框架结构，并说明其有效实现的可能性。

分布式查询处理主要是为不同站点上的局部操作在站点间传送数据对象。一旦得知所需数据对象的位置，便可生成一个有效的可采纳的处理安排。因此PIDDB查询的初级阶段就是要知道所需的任何数据对象的位置所在。这些数据对象并未被局部的目录信息所描述。在知道数据的存放位置后，无论在单个站点还是在各站点之间便可用多种技术来进行有效的查询处理[3-9]。分布计划生成算法主要采用基于半联接程序[9]的启发式爬山技术，例子包括SDD-1[10]系统中所用的查询处理器以及AHY算法。

本文的重点将放在数据定位技术上，因为这是PIDDB的特色所在。主要目标是：

1. 将一些断言和观察资料形式化使之成为一个通用的概念框架结构 。

2. 在该框架结构内建立一些特殊数据定位技术基础 。

1. PIDDB概念的正确性证明

具有部分知识的数据库概念的正确性是通过说明一个完全非知识系统的不可行性（同样与以上提到的大的、完全知识化的系统有关）来得到证明的。这样的系统可通过广播请求目录信息的要求来定位数据；这种搜索过程的复杂度是O(N)，N代表网络中的站点数目。从理论上看，这种方法似乎可采纳，但当N增大时，系统可能导致大量无意义的操作和网络拥挤。随着并发事务（T）和每个事务所引用的数据对象（O）的增大，问题又将复杂化。当T随着N增大时，对于某种N、T、O的组合，可能使响应时间的延迟程度变得不可接受。在这里介绍的模型中，站点能知道某些远程的站点和数据对象，这能大大降低数据定位操作的费用和延迟，结果便提出了所说的部分知识系统。知识模型的使用暗指n<N个站点将被访问（当然希望n≪N），在找到一条访问n个站点的搜索路径后，数据定位问题的费用和时间又会减少。然而，可能的搜索路径数目是随着N成指数增长的。因而为了在数据定位操作时取得一个费用和延迟的线性减少（比例），必须解决难处理的优化问题。本文为实现约束数据定位过程的好处，提出了一个合适的知识模型基础。下面将阐述应知道哪些站点和数据对象，这些信息如何表达以及PIDDB的特性所在。

2. 概念框架结构

PIDDB的概念框架结构的提出是通过采用参考体系结构、确定公理框架结构以及提出查询求精的概念而进行的。

全文假定采用[13][8]中的关系数据模型；关系x（或任何小写字母）表示为R_x。不失一般性，我们假定所有用户级的数据对象是关系。

2.1 参考体系结构

分布式数据库的参考体系结构是通过在集中式（单一站点）数据库系统ANSI/SPARC模型[14,15]上增加一些附加的描述信息分布的模式而形成的。在集中式概念模式上引进了四个新模式，如图1所示。

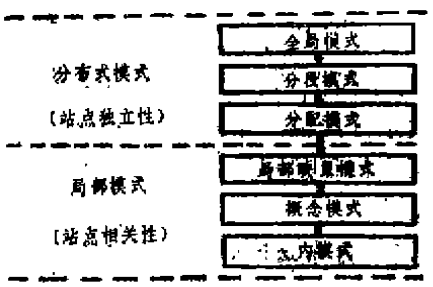


图1：分布式数据库参考体系结构

在它们不涉及局部数据模式、存贮结构或访问策略等细节意义上讲，全局、分段和分配模式具有站点独立性。但局部映象模式却具

有站点相关性。该模式定义了全局与局部数据模式或与数据库管理系统（DBMS）的转换。例如：数据对象可能局部上组织成CODASYL[16]集，但在全局上组织成关系。这样的系统是异质的，而不象同质系统那样所有的站点都使用同一 DBMS（及相同的数据模式）。

全局模式（GS）提供了一个不随时间变化的全局关系R_x的抽象描述。它的作用类似于ANSI/SPARC的概念模式，因为它只与概念记录的内容和语义有关。这些记录的逻辑或物理分布在全局模式中则是透明的。

分段模式（FS）将全局关系分割为逻辑段。当某一关系的子集具有公共特性使得能方便地将它作为原子实体来分配时，这种分割尤为有用。一个特定关系的分段模式记为F（R_x）。关系R_x的段α（一个整数）用R_x^α（如R_y^3）来表示。

关系分配模式（RAS）将逻辑段分配到有关的处理站点上。考虑一些特定的段而不是全部关系的分配模式往往很方便。通常我们认为一些适用于段分配模式的断言和约束同样也适用于关系分配模式。因而，不严格的术语"分配模式"（AS）以后均指FAS。RAS和FAS分别用$A(R_x)$和$A(R_x^\alpha)$来标记。二者由公式$A(R_x)=\cup_\alpha A(R_x^\alpha)$相联系。这意味着所有$R_x$的组成部分被映象至某些段$R_x^\alpha$上。我们希望这种映象能减少段之间的搭接。否则，如果它们在逻辑段之间被部分复制，将难于对数据对象的分配模型化。

[1]中提供了数据对象包含于全局、分段和关系分配模式中的信息定义，这些定义为数据定位和其它操作过程的产生提供了恰当的条件。

2.2 分布透明性

我们希望在全局层以下的所有模式对用户和应用程序都具有透明性。于是，这样的系统对用户好似集中式一般，并提供高度分布透明性。与本文直接有关的分布透明性包括以下两方面：（1）分段的设计；（2）段的定位和重复。

表 1：PIDDB的公理框架结构

类	名称	断言
模式	A1	一个关系的全局模式（GS）和分段模式（FS）是稳定的（在集中式数据库概念模式是稳定的这一意义上讲）
	A2	站点可以保存称为局部全局模式（LGS）的全局模式的子集。查询只能引用LGS中定义的关系。所有在LGS中定义的关系必须从占有GS变体的站点上才是可访问的。
	A3	只有当关系在LGS中时，站点才知道该关系的分段模式。
	A4	当副本被创建或消灭时，关系的分配模式（AS）应随之变化。
	A5	所有站点应拥有相互一致的全局模式版本。
知识	A6	每一模式、关系或段，如果是局部可存取的（被某站点拥有），则它是为该站点所知的，或者它的存贮位置也是已知的。众多的站点同样可知道其他站点的存在。
	A7	站点若仅知道某关系的FS时，该关系就是局部已知（或局部未知）的。
	A8	如果站点既不知道某关系的FS也不知道其AS，则该关系是完全未知的。反之，当站点同时知道其AS和FS时，该关系就是完全已知的。
	A9	一个站点只有当它拥有某关系的FS时，也才有必要知道该关系的AS。

PIDDB的一个基本特性是不需要用户的

（表1续）

原子性	A10	FS的知识具有原子性。如果一个站点知道关系R_i的任意段R_{ij}的定义，则它必然知道这个关系每一段的定义。
	A11	AS的知识没有原子性。站点允许占有一些段而不需知道该关系的任何其它段的分配。
	A12	一个站点只有同时具有某关系分配模式的相应部分时，才可能拥有该关系的一个段。

干预便能完成存贮站点的自动选择和“导航”。否则，希望用户将数据对象与它们的存贮位置相联系是困难和不合理的；在那些有更多动态分配的大系统中，这也是不可能的。在数据修改期间要在PIDDB系统中保持副本间的全局一致也是困难的，因为这种系统不能自动决定一个数据所有副本的当前位置。能够维持全局一致性的大系统必须同时能够对用户提供所有数据位置和副本的透明性。

2.3 PIDDB的公理框架结构

分布式数据库系统（DDBS）中PIDDB类的公理框架结构如表1所示（见上表）。

这种框架结构被划分为三类。模式类产生了参考体系结构在图1 PIDDB系统中的应用。知识断言扩充了这一应用，还定义了有助于求取数据对象位置的术语（这些概念作为知识演算在附录中有进一步的介绍）。原子性断言标识了PIDDB系统中一些固有的限定。

这些公理大部分是自说明的。关于公理A_{10}需要进一步解释。我们用以下两点来证明引入这种限定的合理性：(1)存放那些局部不存在的段FS所要求的额外数据空间与本地段的数据空间大小相比是可以忽略的；(2)如果不应用A_{10}，数据定位操作将极其复杂。（见[2]）。

2.4 查询求精

关于数据对象可提供不同层次的知识，按照查询所涉及到的分段情形，查询又可分为完全不求精（不求精）、部分求精和完全求精（求精）三种：

- 不求精：查询的所有关系是完全未知的（公理A_3）。
- 部分求精：至少有一个要搜索的关系是在*LGS*中，并且其分段模式是已知的（公理A_7）。
- 求精：查询的所有关系是完全已知的（公理A_8）。

通过弄清这些定义和相应模式的知识可以确定这三种情形，因此查询又按其求精情形分为全局、分段或分配查询。

数据定位操作的目标是将用户的全局查询求精成为一相应的分配查询。初始站点主要通过使用其局部可存取的信息查询进行求精。没有求精的残余部分则被移至其余站点继续这一处理。因此，随着逐步简化的版本在站点之间的重置，查询就被逐步求精。当每一个要求查询处理的数据对象的查询完全精化时，数据定位操作就终止。为导出数据定位过程，有必要给出与每一求精过程有关的信息的详细定义。这些在[1]中给出。

3. 拓扑模型

逻辑网络拓扑定义了信息的框架结构，其中任何数据定位过程必须执行。该框架结构指定站点应拥有哪些有关网络其余部分的信息以及信息该如何组织（即操作元数据库）。如果没有这种框架结构，唯一可能的算法就只能随机地测试网络，直到查询完全求精。以下提供的框架结构将适合于建立实际中有广泛应用的网络拓扑结构（如：星形、环形、层状）的模型，PIDDB类中这种定义是隐含的，没有站点知道整个元数据库。站点所知道的部分组成为站点的局部知识视图（LKV）。

该拓扑结构将网络划分为由邻接站点组

成的集合，称为N-集。每一个N-集至少包含一个站点并且所有的站点至少属于一个N-集。属于多个N-集的站点定义了这些集合间的搭接点 (overlapping articulation points)，以这种方式搭接的N-集组定义了一个邻域 (neighbourhood)。为了能初始化或执行查询，站点不要求存贮任何数据对象。

网络可能被划分为很多不相交的邻域。但这些邻域必须以某种方式相联系以便能决定远程数据的位置或存在。我们引进N集的H-集（超集）以提供这种联系。H-集定义了数据对象名与拥有这些对象副本的N-集间的映象。由于H-集主要使用不求精的查询处理，故这种映象是通过全局而非分配模式的名来标识数据对象的。H-集可以包含任意的N-集和邻域的集合。H-集可以嵌套，但其它形式的搭接是不允许的。嵌套结构尤其适合于大型网络，其中的站点群由不同的机构管理。每一H-集相应于一个自治的管理域。这样，复杂的网络要求一个全局H-集来包括整个网络，以保证不出现不相交的知识部分。

每一N-集由一确定的H-集所直接包含或所有。在H-集中，N-集既可能自由，也可能受限。如果H_i包含了N_k，并且没有别的包含于H_i的H-集包含N_k，那么N-集N_k在H-集H_i内是自由的。反之，如果H_i包含N_k，在H_i中N_k是受限的。

站点中存有大量其邻接站点的信息，含有其它一些N-集的有限的信息，而不存在任何其它站点的信息。每一邻域由其通讯参数、使用费用和状态所描述。通讯参数主要包括网络地址和期望的连接花费及延迟。使用费用主要包括每一查询的费用或可能的连接时间或数据访问所需的空间。状态信息主要包括邻接站点是在线或脱线以及它的动态利用情况（如：平均负载）。

图2说明了一个拓扑结构的实例。它由19个站点，11个N-集和6个H-集组成。实心圆代表站点；椭圆表示N-集；H-集由阴影处表示。处于同一嵌套级（对于H_g）的H-集由相同的阴影表示。H-集H_4中给出了通过搭接点相联的扩展邻域的例子。

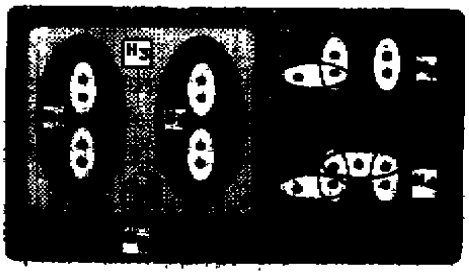


图2：网络实例

〔1〕中讨论了集合边界的确定准则。〔1，2〕中已经提出有关N-集和H-集的信息定义的形式化模式以及该模式用于对数据定位和更新算法的推导。

3.1 N-集的网间连接器(Gateway N_sets)

包含自由N-集的每一个H-集都包括一个可区分的N-集N_g的网间连接器。通过连接器，所有与其它管理域（即远程H-集）的通讯都可被制导。局部网间连接器标记为N_{gl}；远程网间连接器标记为N_{gr}；$N_{gl}\in H_l$，$N_{gr}\in H_r$

引入网间连接器有几点理由：

1. 可大大减少H-集间的通讯量。网间连接器还可在它的局部H-集内分配远程的修改源。
2. 在域间可建立可靠的通讯。（即在N-集网间连接器间可使用正确的网络协议）。
3. 可以有效选择网间连接站点的特性（如：速度、存贮、通道等）来避免通讯瓶颈。

N_g内的所有站点具有一知识核心 (knowledge kernel)，其信息包括：

- 局部（直接包含）的H-集H_l，
- 知道H_l的远程H-集的集合，
- 全局H-集H_g，
- 具有局部数据对象影子的远程H-集的集合（影子的概念见3.2），
- 还包含不具有自由N-集的H-集。

3.2 副本类型

在PrDDB中的多个站点上，数据和元数

据对象可以有副本。副本分为主动和被动两类，分别称为副本和影子。所有的主动副本在拥有的H-集H_o中被占有和维护。在更新后的一个固定时间内，主动副本是可修改的，并趋向一致的值。影子由远程的H-集拥有，它是主动副本的静态"快照"。H_o知道影子驻留于何处。一旦主动副本修改时，H_o能自动对"快照"进行刷新。因此，远离H_o的H-集中的站点能通过访问一个局部影子来减少检索的费用和延迟。

站点可含有远程H-集H_r的影子描述。H_r的版本（远离H_l并为H_l拥有）标记为H_r^l，它与H_r有两方面的区别：〈1〉与H_r不相干的信息不会形成影子。（例如：由于所有的远程通讯是直接通过N-集网络连接器的，故在H_o中数据对象到它们包含的N-集之间不要求有特别的映象）〈2〉在形成影子的H-集内，值的额外信息可以扩充H_r^l，（如：用它们的局部N-集来标识影子），由N_{gr}中接收到的版本可以在N_{gl}中生成H_r^l。

所有知道数据对象（副本或影子）的站点被授与显式或隐式的修改权。H_o中必须有知道副本的站点，且这些站点有显式修改权，它们可以调用修改过程来修正所有副本，并发放刷新后的阴影。知道影子的站点具有隐式修改权，这意味着这些站点可以对H_o提出修改要求。然而H_o不一定实现该要求。因此，H_o中极有必要保持主动副本间严格的一致性。

3.3 界限层次树（Enclosure Hierarchy Trees）

H-集之间的关系及有关的嵌套可用界限层次树(EHT)加以描述。EHT的每一个结点代表了一特殊的H-集并表明H-集所包含或拥有的每一颗子树。EHT中没有表示出N-集。包含H_g的EHT张成整个网络，称为全局界限层次树（GEHT）。图2中的网络实例由GEHT表示，如图3所示。

不可能任何一个站点都知道GEHT。而每一站点都拥有一局部界限层次树（LEHT），只用来描述局部所知的拓扑结构部分（即LEHT定义了LKV（局部知识视图））。LEHT中的结点代表H-集的信息，并且，LEHT内的结点相对位置与GEHT内所确定的相对H-集嵌套是一致的。否则，单独的站点将不能简化已知的H-集间的关系。LEHT中表示的信息主要用于决定哪些远程H-集能够有助于查询求精（见§4），这种信息在网络拓扑结构发生变化而要更换GEHT时也是需要的。然而，能提示修改的站点并不能推断出变化的范围。

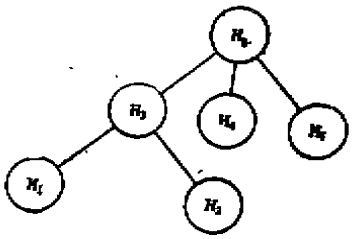


图3：GEHT实例（图2的网络结构）

我们标识了八类LEHT结点以表示四类不同的知识。每一类知识含有一互补集，它由LEHT的一个知识结点和一个"逆"知识结点所组成。类型X（H_r）的知识结点（X是以下描述的P、R、L或S四者之一）定义了H_r的各种不同信息及信息所驻留的位置。"逆"知识结点X^{-1}（H_r）标识了那些拥有X类知识的站点，这些结点由X（H_r）结点所标识的站点拥有。X^{-1}（H_r）信息主要用于分配刷新后的快照，这些快照可以是用户感知的，也可以是影子站点所涉及到的元数据对象。我们建议提出以下八类LEHT结点来对数据定位和更新的分配进行必要的分析：

P（H_i）：H-集H_i的描述是局部存在的，H_i的知识具有原子性；所有属于S_i中LGS的段得到描述，其中S_i位于H_i内（i为l或r）。

P^{-1}（H_r）：已知远程H-集H_r含有一局部H-集H_l的描述。因而，有关对H_l的具有全局意义的修改能直接传给N_{gr}。（如影响H_l中可用段的集合所作的一些改变）

R（H_r）：局部知道远程H-集H_r的存在，H_r通过N_{gl}和N_{gr}是可隐式访问的。

$R^{-1}(H_r)$：已知远程H-集H_r知道局部H-集H_l的存在，因而有关对H_l进行的具有全局意义的修改将直接传与N_{gr}。

L（H_i）：已知H_l中某一N-集具有一H-集H_i（i为l或r）的描述。这些局部影子允许数据定位算法能很容易地确定H_l能否对查询的任意部分进行求精。当H_l是一远程H-集H_r时，L结点可以在H_l中确定重置至H_r的得益。这有助于避免无益的远程事务处理，并能减少数据定位的全部费用和时间延迟。H_r只意识到一些远程H-集拥有其局部H-集的影子描述，而影子的特殊作用是完全局限于包含它的H-集的。

L^{-1}（H_r）：含有远程H-集H_r描述的站点知道H_l中任何局部站点，这些局部站点含有引用这种描述的L结点。

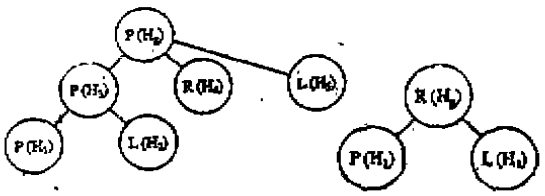
S(H_r)：拥有段R_r^a影子拷贝的那些站点必须也同时拥有一个S结点以便能请求并接受有关那个影子的修改。这一结点类型标识了具有版本H_l的局部N-集，H_l是占有R_r^a的H-集H_r的一个版本。H_l不必直接被影子站点所知，它是足以用来确定H_o（R_r^a）的H_l的最小子集。

S^{-1}（H_r）：那些段被影子化的H-集知道哪些远程H-集对哪些段应产生影子。因此，N_{gl}能够保证刷新的影子拷贝被分配在要求进行局部修改的地方。与H_l有关的修改也能直接作用到N_{gr}。H-集的子集H_r'可由具有S^{-1}结点的网间连接站点产生并分布。

信息的定义与每一种LEHT结点都相关，表2给出了它们所在的位置以及最终得到的知识。模式定义中标有横线的量表明引用的是数据对象的名称而非值。本节所使用的知识操作符（φ、ϕ、k和K）在附录中有定义。例如：表中指出L（H_r）结点是保留在含有远程H-集的影子副本的H-集内的。模式定义说明了局部N-集N_k内的一些站点知道那些含有远程H-集H_r某种版本的一些站点；所有权约束说明了LEHT结点必须由局部N-集N_l内的某些站点S_a所有；知识约束则定义了根据所占有的结点而获得的信息；这种情况下，H_l内拥有L（H_r）LEHT结点的站点就可知道远程H-集H_r的存在。

虽然，P^{-1}和R^{-1}结点的模式定义和知识约束是相同的，但它们的语义和应用情况却有明显的区别。在模式定义中集合K并不是ϕ是因为"只能知道另一H-集的网间连接站点"这一约束所限。

图4给出了图2的网络结构中的一些LEHT实例以及图3的GEHT。



（a）关于$S_a \in H_l$的LEHT　　（b）关于$S_a \in H_r$的LEHT

图4，LEHT实例（图2的网络结构）

3.4 局部知识视图的约束

这里提出了一系列定义每个站点应拥有的最小知识集的局部知识视图约束。这些约束条件不仅使得其它H-集里的数据对象被定位，而且能使副本间保持一致性，还能刷新阴影区的数据对象。这些要求将通过以下两个完全性约束施于LEHT集合而得到满足：其一：所有GEHT结点（H-集）必须至少由一个LEHT P结点所描述；其二：GEHT的相对结构在所有站点都是可推导的。为使这些要求得到满足，提出以下的知识约束：

1. 所有N_{gl}中的站点拥有知识核心（见§3.1）
2. 对于每个H_g的子集，网间连接的N-集站点拥有P、L或R LEHT结点（即这一约束定义了关于H_g知识核心所包含的内容）。
3. 所有不在N_{gl}中的站点知道N_{gl}（即知道N_{gl}中每一站点的网络地址）。
4. 每个邻域中至少有一站点拥有直接包含它的H-集H_l的描述。
5. 每个邻域中，至少有一个站点知道，
   a. 至少是直接祖先H-集，和/或者
   b. 所有直接的子孙H-集。

从信息论观点来看，在可以导出一组不大严格的要求这一意义上讲，这些约束条件是弱的。所提出的这组要求重复了操作简单

表 2：LEHT结点定义

类型	模式	约束 所有权	约束 智识
$P(H_1)$	$\langle \underline{H_1}, H_1 \rangle$	$\exists S_n \in N_1 : \phi^n P(H_1)$	$\exists S_n \in N_1 : \phi^n H_1$
$P(H_r)$	$\langle \underline{H_r}, H_r' \rangle$	$\exists S_n \in N_1 : \phi^n P(H_r)$ $\forall S_n \in N_{g1} : k^n \Phi(P(H_r)) \wedge k^n N_{gr}$	$\exists S_n \in N_1 : \phi^n H_r'$
$P^{-1}(H_r)$	$\langle \underline{H_r}, K(H_1) \rangle : K \in N_{gr}$	$\forall S_n \in N_{g1} : \phi^n P^{-1}(H_r)$	$\forall S_n \in N_{g1} : k^n K(H_1)$
$R(H_r)$	$\langle \underline{H_r} \rangle$	$\exists S_n \in N_1 : \phi^n R(H_r)$ $\forall S_n \in N_{g1} : k^n \Phi(R(H_r))$ $\wedge k^n N_{gr}$	$\exists S_n \in N_1 : k^n H_r$
$R^{-1}(H_r)$	$\langle \underline{H_r}, K(H_1) \rangle : K \in N_{gr}$	$\forall S_n \in N_{g1} : \phi^n R^{-1}(H_r)$	$\forall S_n \in N_{g1} : k^n K(H_1)$
$L(H_1)$	$\langle \underline{H_1}, \Phi(H_1) \rangle : \Phi \in N_k \in H_1$	$\exists S_n \in N_1 : \phi^n L(H_1)$	$\exists S_n \in N_1 : k^n H_1$
$L(H_r)$	$\langle \underline{H_r}, \Phi(H_r') \rangle : \Phi \in N_k \in H_1$	$\exists S_n \in N_1 : \phi^n L(H_r)$	$\exists S_n \in N_1 : k^n H_r$
$L^{-1}(H_r)$	$\langle \underline{H_r}, \Phi(L(H_r)) \rangle : \Phi \in H_1$	$\exists S_n \in N_k \in H_1 : \phi^n L^{-1}(H_r)$ $\wedge \phi^n P(H_r)$	$\exists S_n \in N_k \in H_1 : k^n K(H_1)$
$S(H_r)$	$\langle \underline{H_r}, \Phi(H_s) \rangle :$ $\Phi \in N_k \in H_1 \wedge$ $R_x^a \in H_s \wedge H_s \subseteq H_r'$	$\exists S_n \in N_1 : \phi^n S(H_r)$ $\forall S_n \in N_{g1} : \phi^n S(H_r) \wedge k^n N_{gr}$ $\forall S_n \in \Phi(H_s) : \phi^n P(H_s)$	$\exists S_n \in N_1 : k^n H_s$
$S^{-1}(H_r)$	$\langle \underline{H_r}, K(H_1), \{R_x^a : \exists S_n \in H_r : \phi^n R_x^a \wedge \underline{H_1} = H_0(R_x)\} \rangle :$ $K \in N_{gr}$	$\forall S_n \in N_{g1} : \phi^n S^{-1}(H_r)$	$\forall S_n \in N_{g1} : k^n K(H_s)$

性和健壮性的一些知识。

4. 数据定位算法

提出数据定位算法的基础是用来说明LKV的典型应用。这种算法依赖于启发式搜索的层次。这种方法里，搜索的范围是逐渐增加的，直到所搜索的设要么被定位，要么发现它不存在。该方法包括以下几个层次：

- 在初始化站点S_{init}上，用LKV尝试局部的查询求精。
- 局部求精后的残余部分被分布到局部的N-集站点以得到更多的信息。
- 如果在局部N-集中不能完全求精查询，则剩余部分将通过搭接点重置入局部邻域N_{Hl}的其它N-集中。这一过程将持续直至查询被完全求精或不再有N_{Hl}来试探为止。
- 如果在N_{Hl}内查询还不能完全求精，那么将从GEHT的根H_g开始进行穷举搜索（询问LKV和试探N_{Hl}类似于对GEHT进行由叶子向上至根的试探。如果这种搜索由于可得的LEHT的不完备而失败，那么将有必要由GEHT的根开始进行自顶向下的搜索）。

下面给出了数据定位过程的主要步骤。文[2]中描述了一个更完整的算法，它将全局查询求精和基于LEHT的"导航"结合在一起。本文假定用户的查询已求精至分段级别，并且只搜索单个的段R_x^a。

1. 如果$k^n : R_x^a$，那么结束。
2. $\forall \phi^i P(H_i)$：如果$R_x^a \in H_i$（或H_r'此时$i=r$）那么结束。
3. $\forall \phi^i \perp (H_r)$，重置至$S_n \in H_i : \phi^n P(H_r)$询问$H_r'$，等待回答，如果$R_x^a$现已定位，则结束。
4. $\forall \phi^i R(H_r)$，重置至$N_{gr} \in H_r$，询问H_r，等待回答，如果R_x^a现已定位，则结束。
5. 试探局部邻域直至$k^n : R_x^a$或N_{Hl}已用完。如果R_x^a现已定位，则结束。
6. 重置至N_{gl}，由GEHT的根（即从H_g）开始穷举搜索。用户可以干预，要不就限制搜索的程

度（费用和延迟）。

此算法在任何查询的初级阶段都将被调用。因此，重要的是尽可能减少有关的处理延迟以避免造成数据定位的"瓶颈"。该问题的某些方面在[2]中有详细论述。

带副本的分布式数据库PIDDB系统现已得到关注和发展。其特色在于限制了每个站点可用于网络中各站点和数据对象子集的知识。在数据分段、定位以及副本方面，PIDDB都具有全透明性。PIDDB系统固有的特性就是适合于超大系统并能保持子域的自治性。为了使副本能在它们各自的子域中得到可靠的创建或删除，我们对站点当前或初始数据段的分配没有作出任何假设。本文提出了一种启发式网络拓扑结构，它将处理站点组成为邻接点集或超集（N-集和H-集）以保证数据定位过程易于控制。这种拓扑结构又可用树来加以描述，可以看到，此表示能方便地被转换成站点所拥有的局部知识树，我们已经说明了这些树在查询处理期间对确定段定位算法的应用。

文献[1]中也提供了这样一种结构，它允许信息提供者用其各自所有的数据对象部分来共同定义"分布式虚拟数据对象"，这些数据对象如其它影子对象那样被说明和分布为快照。所提出的这种机制按[3]中全局模式的观点来定义分布式数据对象。

参考文献

1. M. Blakey, Partially Informed Distributed Databases: Conceptual Framework And Knowledge Model, Tech. Rep. 80, Dept. of Comp. Science, Monash Univ., Melboume, Australia, Dec., 1986.
2. M. Blakey, Partially Informed Distributed Databases, Data Location Algorithm, Tech. Rep. 87/85, Dept. of Comp. Science, Monash Univ., Melboume, Australia, May, 1987.
3. C.J.Date, *An Introduction to Database Systems, Volume 1, 4th Edition*, Addison-Wesley, Reading, Massachusetts, 1986.
4. J. D. Ullman, *Principles of Database Systems*, Comp. Science Press, Rockville, Maryland, 1982.
5. M. Jarke and J. Koch, Query Optimization in Database Systems, *ACM Comp. Surveys* 16, 2, (Jun. 1934), 111-152.
6. W. Chu and P. Hurley, Optimal Query Processing for Distributed Database Systems, *IEEE Trans. on Computers* C-31, 9, (Sep. 1982), 835-850.
7. C.T. Yu and C. C. Chang, Distributed Query Processing, *ACM Comp. Surveys* 16, 4, (Dec. 1984), 399-433.
8. C. T. Yu, C. C. Chang, M. Templeton, D. Brill and E.Lund, Query Processing in a Fragmented Relational Distributed System: Mermaid, *IEEE Trans. on Software Eng. SE*-11, 8 (Aug. 1985), 795-809, IEEE.
9. S. Ceri and G. Pelagatti, *Distributed Databases: Principles and Systems*, McGraw-Hill, New York, 1985.
10. P. A. Bernstein, N. Goodman, E. Wong, C. L. Reeve and J. B. J. Rothnie, Query Processing in a System for Distributed Databases (SDD-1), *ACM Trans. on Database Sys.* 6, 4, (1981), 602-625.
11. A. Hevner and S. B. Yao, Query Processing in Distributed Database Systems, *IEEE Trans. on Software Eng. SE*-5, 3, (May 1979), 177-187.
12. P. M. G. Apers, A. R. Hevner and S. B. Yao, Optimization Algorithm for Distributed Queries, *IEEE Trans. on Software Eng. SE*-9, 6, (Jan. 1983), 57-68, IEEE.
13. E. F. Codd, A Relational Model of Data for Large Shared Data Banks, *Comm. ACM* 13, 6, (1970), 377-387.
14. C. Mohan and R. Popescu-Zeletin, Impact of Distributed Data Base Management on the ISO-OSI And ANSI/SPARC Frameworks, Proc. *15th Hawaii International Conf. on Systems Sciences*, Honolulu, Jan., 1982, 532-542.
15. ANSI, Reference Model for DBMS Standardization, *ACM SIGMOD Record* 15, 1, (Mar. 1986), 19-58.
16. CODASYL, *Data Base Task Group Report*, ACM, New York, Apr., 1971.

附录：知识演算

PIDDB中查询执行策略的推导和分析经常需

Ada相关的分布式数据库管理系统Ada-DDBMS

徐泽同 （中国科学院数学研究所）

摘 要

此文介绍了中国科学院科学数据库POREL小组在维也纳技术大学应用信息研究所消耗10人年，在VAX，UNIX上重新模拟实现的分布式数据库管理系统POREL；也介绍了中国科学院数学研究所Ada-DDBMS小组正在进行的VAX，VMS，DECNET上的Ada相关的分布式数据库管理系统Ada-DDBMS。

分布式数据处理及其核心分布式数据库管理系统因它能克服集中式数据处理及其核心集中式数据库管理系统的很多弱点，也因集中式数据库技术与计算机数据通讯技术的成熟，因而在过去十余年来国内外计算机科学界给予了此研究领域广泛的重视，投入了不少研究人员，花了不少钱，出了不少论文，也造了不少原型系统，而今据这方面的权威人士认为，DDBMS的研究是应走向实用化的时候了。我国地域辽阔，在离散的计算机应用初具规模之下，进而不失时机地走向大范围的联网的计算机应用，无论于何个方面的现代化的发展，意义均是十分深远的。无论是POREL小组在外的工作，还是Ada-DDBMS小组在内的工作，都是为了这一目的，我深望我们终会达此目的。

1. POREL简介

*POREL*是1977年以来西德计算机科学家E. J. Neuhold教授领导下研制的计算机非均质网上的关系型数据库管理系统。在1984年前的斯图加特时期，为研制此系统花了35个科学家人年，40个学生人年；花了450万西德马克；出了不少论文，其中博士论文七篇，硕士论文36篇；在PDP11，RSX11M上作程序近七万行，遗憾此系统过于复杂与庞

要关于处理站点所具有的推理的知识。以下提出的命题演算可以简化这种知识的表达和处理。从两种类型定义了四个新的知识操作符。

除了通常的结构化和量化的信息外，*PIDDB*中的数据对象如果没有其位置信息将不能被完整地描述。此处用布尔变量操作符，k和φ来表达和测试这种信息：

k：k^nOBJ为真，如果对象OBJ在站点S_n已知的话（意味着OBJ可以由站点S_n拥有，也可以是站点S_n知道拥有OBJ的其它站点的标识）。

φ：ϕ^nOBJ为真，如果对象OBJ由站点S_n拥有。这两个操作符由推理规则$\phi^nOBJ \rightarrow k^nOBJ$联系起来。

用这些操作符表示的断言的真实性与陈述该断言的位置无关。例如：$\phi^3 x$，^{5}可能为真，但S_4站点并不知道。OBJ既可为一数据，也可是元数据对象，还可是一个站点。如果OBJ是一个站点，那么φ操作符将不适用，这是由于在PIDDB结构中并不支持"站点间互相占有"这一概念。

第二类操作符是大写的K和Φ。它们定义了已知或拥有OBJ的站点集合。这一表示法简化了知识模式及约束的表达：

$$K(OBJ) = \{S_n : k^nOBJ\}$$

$$\Phi(OBJ) = \{S_n : \phi^nOBJ\}$$

〔冯惠华 贝来倩译自"Proc, of the 13th VLDB Conf., Brighton, 1987, 谭 穆校〕