设计异构型分布式数据库系统的几个问题

金志权 周晓方 孙钟秀

（南京大学计算机系）

ISSUES ON THE DESIGN OF HETEROGENEOUS DDBSs

Jin Zhiquan, Zhou Xiaofang and Sun Zhongxiu

(*Department of Computer Science, Nanjing University*)

ABSTRACT

The development of computer software system will be the combination of various techniques. The hardware system may consist of a bunch of heterogeneous sub-systems. Based on homogeneous and heterogeneous DDBSs LSZ and LSZ2 we developed, the paper discusses the system architecture, link method, parallelism as well as the combination of AI and DB technique for heterogeneous system.

摘 要

计算机系统的软件系统发展将是各种技术的结合和渗透。硬件系统将由大批异构型的分布式的分系统构成。本文以我们研制的异构型、同构型分布式数据库系统(DDBS) LSZ 和LSZ2 为背景，讨论异构型系统的结构、连接、平行性以及AI 和DB 技术的结合等问题。

1989 年9 月25 日收到，1989 年11 月25 日定稿，本课题是国家自然科学基金资助项目。

§1. 引　　言

计算机系统的软件系统发展将是各种技术的渗透和结合。典型的有AI 和DB 技术的结合渗透；分布式技术用于诸如知识库系统、图形系统等；软件工程和智能系统的渗透等。未来计算机系统的硬件系统将由大批异构型分布的分系统构成[1]，而其中的分系统可能是各种类型的计算机，包括多处理机的计算机。显然，在研制和开发当前各种软件系统时，为朝着上述目标累积各种经验是有益的。

本文通过LSZ 和LSZ2 两个异构型，同构型DDBS 的设计和实现[2, 3] 讨论上述有关问题。主要讨论4 个问题：1. 基于小型/ 微型计算机的DDBS 的系统结构。2. 上述结构下的平行性。3. 异构型和同构型。4. AI 和DB 结合。

§2. 系统结构

文献[2]、[3] 介绍了LSZ 和LSZ2 采用簇作为DDB 的场点，每个簇由一批微型/ 小型计算机(称为结点)，和一个共享服务器(磁盘) 组成(见图2)。根据实际应用背景和异构型特性，LSZ(以下代表LSZ 和LSZ2) 的诸簇可以是异构的，而簇中诸计算机基本上是同构的。与分布式系统的一般簇结构不同，在LSZ 的簇中有一台共享服务器或一台称之为DB 结点的计算机存放场点DB(nodal DB)。这种结构把过去DDBS 中的一个场点从一台计算机(见图1. (1)) 扩展成一台具有多个处理机、多个内存、多个磁盘和一个共享盘的虚拟DB 机器(见图1. (2))，其中(2) 中的进程可能是一个处理机。现有的各种网络提供的网络功能(如Ethernet 上的Sun3 的NFS, Omninet 上的Constellation 等) 支持簇中其它结点(称之为从结点) 象访问本机磁盘一样存取共享盘或它机上的硬盘。因此，上述簇结构特别适于小型微型机的DDBMS 结构。

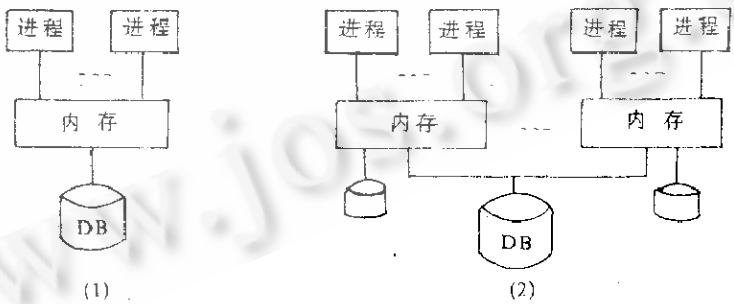


图 1

无疑，为了与此结构相适应，簇中必须有一个簇管理调度程序。根据DDBMS 的特点，我们把这个调度程序的功能加入事务管理程序TM 中[11]，进而把TM 分为两大部分：CTM (Cluster TM, 一场点只有一个) 和PTM(Process TM, 可有多个)，即 $TM=CTM+\sum_{i=1}^{k} PTM_i$，其中$k$ 为一DB 场点当前处理的(子) 事务数。CTM 管理簇中处

理机、场点DB 的并发存取、与外场点的联系等。它也根据需要创建PTM 处理各个事务。PTM 是本簇启动的事务的协调者或外场点启动事务的代理者，负责管理事务和子事务的执行，CTM 作为一个常驻进程存放在DB 结点上，PTM 则可在簇中任一结点上。在PC 簇中，PTM 对应一台PC 机。在Sun 簇中，PTM 对应Sun 机中一个进程。这种结构的优点是：1. 簇中结点的增减是十分方便的。只要向CTM 登记，一个簇可以动态地扩大和缩小。2. 增加了由微机组成的场点的处理能力，提供了平行处理的环境(见 §3)。

LSZ 提供三种不同类型的簇结构(见图2)，以适应异构性。第一类PC 簇结构适应于单用户的微型机。簇中PC 机形成一个处理机库，由CTM 调度。每台PC 机有DDBMS。各机各自独立平行地接收处理用户的数据请求(事务)。读写DB 时受CTM 协调。场点DB 放在共享服务器上。DB 结点是由共享服务器和一台驻留CTM 的专用PC 机组成。第二类Sun 簇结构适用于多用户的工作站。这类簇的CTM 管理簇中计算机，并对其进程进行调度。如，它可在任一台从结点上创建一个进程(如PTM)，场点DB 放在DB 结点上，从结点也可通过Mount 命令(Sun OS 提供) 把DB 结点的LSZ 子目录安装在本结点上。上两类簇的CTM 功能相同，从进程角度看结构也一样，但实现模式不同。第三类Dual 簇结构是适用于下列两种情况：1. 一台机器不能直接连接到LAN 上，这时通过它和PC 机的RS232 接口互连，间接地把它连到LSZ。2. 希望在这类机器上尽量少做工作。此时可把LSZ DDBMS 放在PC 机上，而Dual 机相当于一个具有局部处理能力的DB 存储设备。PC 机把单场点的数据操作作为写操作向RS232 输出，而Dual 机用读操作从该RS232 读入所需执行的操作。

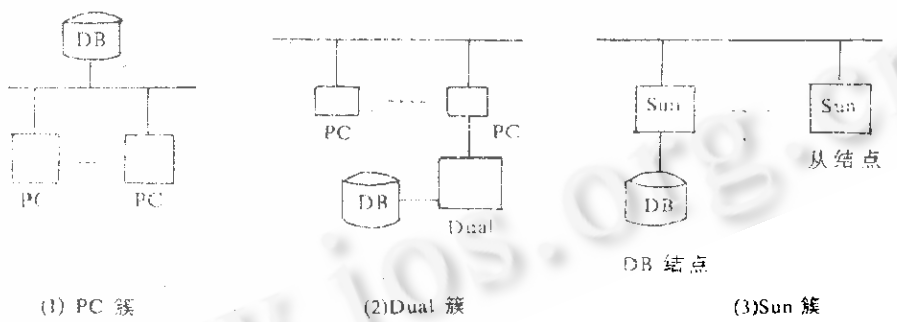


图2 三类簇

由此可见，LSZ 的系统结构是适应于第一节所述的，将由大批异构型分布的分系统构成的未来计算机系统的硬件系统。

§3. DDBS 的平行性

过去的经验表明，当多个用户同时使用一台超级微型机的DB 时，查询的响应时间是很慢的，这是由于所有查询处理、读写DB 操作，实际上都是串行的。LSZ 运行在微型机环境，系统的目标是最大限度地提高微机的使用率，尽可能增加事务处理

的平行性。为此，LSZ 采取簇作为DDB 场点。LSZ 的簇和TM 结构导致三类平行性：用户级、事务级和局部级(子事务级)。

用户级平行性是指，当一个场点的多个用户同时进入LSZ 时，系统平行地分析处理这些用户的请求，作查询优化，对事务进行分解，直至将子事务分布到有关场点以及等待各场点的回答(即2PC 协议)。因LSZ 用户可在簇中任一台机上进入系统，系统在该台机器上创建一个由PTM 控制的user 进程，由一种通信机制(PC 簇用send, receive; Sun 簇用RPC) 沟通PTM 与CTM 的联系，于是，运行在不同机器上的User 进程平行地处理各自事务。由此可见，用户级平行性可通过设计适于平行处理的系统结构(簇) 及其相应结构的(TM 的结构) 软件设计来实现。

事务级平行性是一个事务被分解成多个子事务，在多个场点执行的DDBMS 固有的平行性。这是由数据分布和关系分割引起的，以增加通信开销为代价。

在LSZ, 当一场点的CTM 接到其它场点的子事务请求后，它将根据工作负载和机器资源，在本结点或从结点产生一agent 进程，由其PTM 控制该子事务的执行，因而LSZ 提供了多个子事务在一场点内平行执行的可能性。

局部级平行性，当一个场点执行K 元连接时，速度明显下降，特别是对大关系，因此提高多元连接的执行速度是有意义的。LSZ 采用平行执行的方法来实现，对不同的局部DBMS 采用不同的处理策略。设$\theta = R_1 \infty R_2 \cdots \infty R_K$ 是一个单场点的k 元连接查询。$R_{i_1}\ R_{i_2} \cdots R_{i_m}$ 是R_i 的一个划分，且满足下列条件：

$$R_i = R_{i1} \cup R_{i2} \cdots \cup R_{im}$$

$$R_{ij} \cap R_{ik} = \phi \qquad \text{当 } j \neq k$$

令

$$\theta_1 = R_1 \infty \cdots R_{i1} \cdots \infty R_k$$

$$\vdots$$

$$\theta_m = R_1 \infty \cdots \infty R_{im} \cdots \infty R_k$$

显然$\theta = \theta_1 \cup \theta_2 \cdots \theta_m$。在Sun 簇，$R_i$ 的划分可以简单地在R_i 的一个属性上增加一元限制来实现。例如，设有查询Q 为：

```
select    *
from      S, SC, C
where     S. sno=SC. sno and C. cno=SC. cno
```

其中S、C、SC 分别代表学生、课程和选课关系。对上述查询中的S 加一元限制，如学号sno, 则Q 能变换成下列$Q_1 \cdots Q_m$ 的并。

$$Q_1\begin{cases} \text{select} & * & /^*89\text{届学生}^*/ \\ \text{from} & S, SC, C & \\ \text{where} & S.sno = SC.sno \ \ and & C.cno = SC.cno \ \ and \\ & S.sno \ \ between & 890001 \ \ and \ 900000 \end{cases}$$

$$Q_2\begin{cases} select & * & /^*88\text{届学生}^*/ \\ from & S, SC, C & \\ where & S.sno = Sc.sno \ \ and & C.cno = Sc.cno \ \ and \\ & S.sno \ \ between & 880001 \ \ and \ 890000 \end{cases}$$

LSZ 把$Q_1, Q_2 \cdots$送给簇中几台Sun 3 同时执行。为了使多台机器工作负载接近，关系的有关属性的选择度(各值在属性中出现的频率)，必须作为语义信息保存，它们也用于语义优化。

由于d BASE 不同于Unify(LSZ 中的局部DBMS)，LSZ 一方面把LSZ 的全局数据语言GRDL 转换成dBASE，另一方面也作K 元连接的优化。为了避免产生中间关系，LSZ 采用一次连接法[4] 处理多元连接。一次连接法处理上述查询Q 时，若选取R_1为划分关系的话，则只需把它分成m 个等分。然后先对$R_2 \cdots R_k$ 以连接属性为关键词建立索引文件$I_2, I_3 \cdots I_k$，最后若干台PC 机同时用一次连接法各做一个k 元连接$Q_i (i = 1, 2, \cdots m)$。

无论事务级平行，还是局部级平行，都涉及到CTM 如何在簇中寻找空间结点。这包括三个问题: 机器空间状态的度量问题、机器的调度策略以及状态信息的获得方法。

机器的空间状态度量是指一台机器尚有多少能力可被利用。这用下式度量$R = (1 - E) \times C$ 其中C 是机器总能力，E 是该机的已利用率。

CTM 的功能决定了在这种簇结构中，机器的调度策略是集中式的。由CTM 的DB 结点收集状态，并作出选空间结点的决定。但从DDBS 的全局观点看，调度策略既不是集中式，也不是分布式，而是簇(分组) 调度策略。

CTM 获取其成员结点的状态，有两种方法: 查询式和报告式[5, 6]。查询式是CTM 定期发送请求各成员状态的信件。报告式是成员结点一旦状态发生一定量的变化，便向CTM 汇报其状态。若簇中结点数为k，则一次信息收集，查询式要$2k$ 或$k + 1$ 封信；报告式最多要k 封信。

综上所述，在LSZ 有四种主要进程: CTM，user, agent, parallelor (k 元连接平行执行者)。在PC 簇，上述每种进程可运行在簇中任一台PC 机上。在Sun 簇，DB 结点除了运行CTM 外，还可能运行user, agent 进程。在从结点上可有user、agent、parallelor 进程。

§4. 异构型和同构型

[2] 指出，同构型系统是异构型系统的特例。按理讲，有了异构型就不需同构型，但考虑到系统效率以及与同构型的原集中库系统的兼容性，一般不用异构型代替同构型，而是另建独立的同构型系统。另一方面，同构型系统是否能根据环境需要扩充成异构型。这也是一个值得研究的问题。下面就我们的研制经验讨论这个问题。

我们的研制工作是，首先研制了异构型DDBS LSZ，然后建立Sun3 上的同构型DDBS Unify*。因而在设计实现Unify* 时，有三种可能选择: 1. 用LSZ 代替Unifty*。2. 建立独立的同构型系统。3. 建立一个容易扩充异构型的同构/ 异构型系统。上面所说的用异构型代替同构型所存在的兼容性和系统效率问题，分别是设计和实现方面的问题。设计问题是起因于，异构型的设计通常是集成各局部系统的共性部分，而不考虑它们的个性。例如LSZ 的数据类型取局部系统的交集，因此比Unify 少，它不考虑Unify 在建立关系时要给出期望元组数，指出关键域等等个性，因而若不对

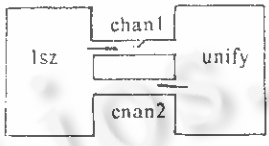
LSZ 设计作修改，则无法与Unify 兼容。实现问题是由于对异构型的实现通常强调统一处理，而不考虑特殊处理，因此效率低。例如，[2] 指出，若用LSZ 处理中间临时关系的办法(建立临时关系) 用于Unify，则查询效率将大为降低。因Unify 建立关系的速度相当慢。由此可见，用LSZ 代替Unify* 是不适合的。同样，建立一个独立的同构型系统又不利于我们的计算机环境，并且目前对未来计算的看法是，异构型计算机以及连网以提供对远程资源的访问[1]。

根据上述考虑，我们采用第三个选择。这种选择的前题是原局部系统具有较标准的数据语言。Unify 用SQL, 适于作为这种系统的基础。于是，LSZ2 呈现在用户面前的是两种构型：划出Ethernet 网上一批Sun3, 构成效率较高的，与Unify 兼容的同构型DDBMS Unify*; 接上IBM PC、Dual 机和Omnient 网便构成异构型LSZ。实现这样的系统，在设计上应考虑数据语言及类型、与局部系统的接口。在实现上应考虑局部系统的独立性以及由此产生的恢复、临时关系的处理等问题。

本节重点讨论与局部数据库系统的接口及其独立性。文[2] 给出的LSZ DDBMS 的总体结构如图3 所示。

在一个异构型系统中，局部系统的自主性、独立性是很重要的，而开发一种简便的接口方法，把一台计算机接入异构型系统是关键。对DDBMS 来说，就是要有一种接口方法能方便地把一个新的局部DBS 加入异构型DDBS 中。实际上，这种方法对AI_DB 结合也是十分有用的，因目前大部分KBS 和AI 语言没有与DB 系统的连接能力，也是由于没用好的接口方法。

在LSZ, 我们用Unix 的管道(pipe) 作为与局部DBMS 连接的标准接口，从整体上看，DBMS 总是从标准输入设备接收命令和查询，向标准输出设备写结果。这样，LSZ 与局部DBMS 的联系可看成它与局部DBMS 的标准输入/ 出口的接口问题，只要在它们之间建立两个管道，其中一个定义为局部DBMS 的标准输入，另一个为标准输出，就能使局部DBMS 从LSZ 读入要执行的命令，而把结果交还给LSZ。下面是连接形式和构成这种连接的程序段。




```
init_ipe ( )
{          int chan 1 [2], chan 2 [2], pd;
           if (pipe (chan 1) < 0) return (0);  if (pipe (chan 2) < 0) return (0);
           if (( pd=fork ( ))==0)
           {     dup2 (chan 1 [0], 0);
                 dup2 (chan 2 [1], 1);
                 close ( chan 1 [1]);  close (chan 2 [0] );
                 execl ["/bin/sh", "sh", "−C", "$ HOME/bin/unify", 0)
      }
      if (pd==−1) return (0);
      close (chan 1 [0]);  close (chan 2 [1]);
      unity_rpd =chan 2 [0];  unifty_wpd =chan 1 [1];
```

```
        unify_wait ( ); return (1);
}
```

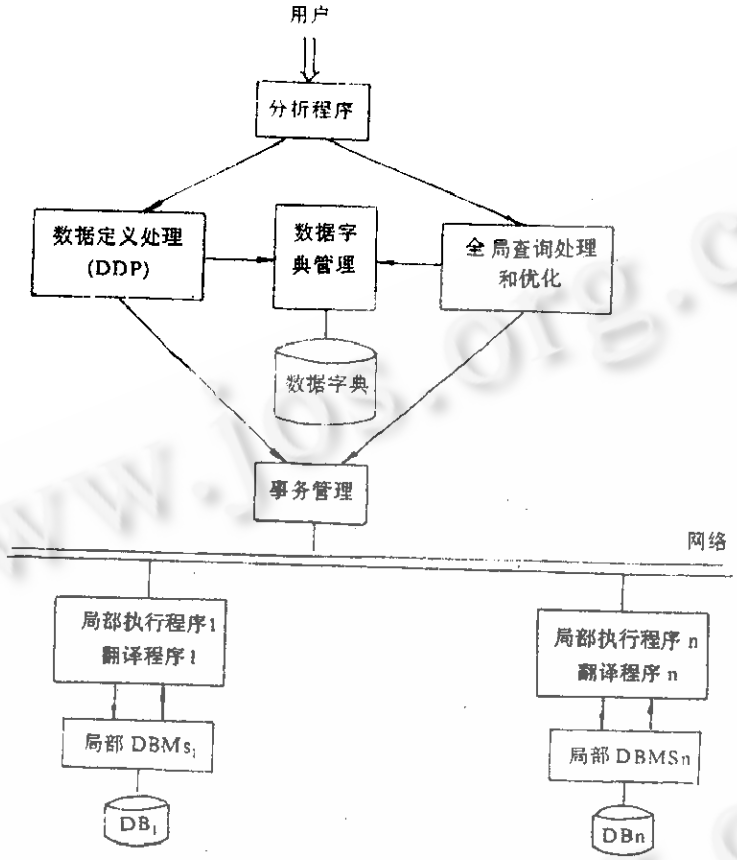


图 3

pipe 的特点[10] 保证了它们之间的读写能协调同步工作。实际上, §2 介绍的Dual 簇中PC 机和Dual 机的连接也属这种方法。

LSZ 提供单一的全局数据语言GRDL [2]:

<GRDL 语句> ::=< 数据定义语句> | < 数据操纵语句> | < 控制语句>

< 数据定义语句> ::=< 关系定义> | < 关系撤消> | < 关系加入> | < 关系收回> | < 子关系定义> | < 子关系撤消> | < 窗口定义> | < 窗口撤消> | ……

< 数据操纵语句> ::= 同SQL

< 控制语句> ::= 同SQL

经查询处理和优化后, GRDL 被转换成中间形式(有利于异构型系统与局部系统的接口), 再由TM 把各子事务分布到有关场点。每个场点有一个局部执行程序负责执行表示成执行图的子事务, 并由翻译程序把中间形式命令转换成局部DBMS 接受的命令形式[2]。例如, 在Sun 簇, 它把关系定义语句的中间形式转换成按菜单方式

工作的一系列键盘操作，包括敲打选择项号、回车换行、按菜单在屏幕确定位置填写关系名、属性名、类型等。这一系列键盘操作将通过上述的标准接口送给Unify 执行。

这个接口方法实现了我们构造LSZ 的原则，不改造原集中DBMS, 而把它们加入LSZ。同样，这种连接也可用AI 和DB 的连接。

§5. AI 和DB 技术结合

未来计算系统要求AI 和DB 技术与其它技术一道工作[1]。AI 和DB 的结合可以是两种系统通过一个接口互相连接的松耦合式结合[7], 也可以是设计实现一种系统时加入另一种系统的技术的紧耦式结合。如DB 技术用于AI 系统，或反之，从适用于DB 系统的特定AI 特性的需要看，AI 的知识表示、推理和搜索能力以及基于AI 的用户接口功能是最基本、主要的技术。为了适应DB 的这种发展，我们在LSZ 中，对后者作了简单的尝试，这就是[8] 介绍的语义优化。下面先给出语义优化中语义等价的形式描述[9]。

1). 一阶逻辑中的几个概念

一阶理论：在谓词演算基础上，再定义一些新的公理(称为非逻辑公理), 所得的形式系统称为一阶理论。

理论的解释：对理论中所有函数和谓词进行定义，给所有变量赋值，称作该理论的一个解释。

理论的模型：当且仅当一个解释使理论中所有公理为真时，称该解释是这个理论的模型。

外延数据库(EDB) 由事实组成，这些事实就是原来DB 中的元组。内涵数据库(IDB) 由规则组成，这些规则就是语义规则，即DB 完整性约束条件，它用于表示DB 允许的状态和状态间的转换。IDB 中每个语义规则可看作是一个非逻辑公理，EDB 允许的状态使这些非逻辑公理为真，所以EDB 是以IDB 为非逻辑公理的一阶理论的模型。

2). 语义等价

查询可用一阶逻辑的合式公式表示，因此语义等价可定义为：

查询Q_1 和查询Q_2 语义等价$\Longleftrightarrow \vdash_T Q_1 \longleftrightarrow Q_2$, 其中$T$ 是以IDB 为非逻辑公理的一阶理论。

常规的查询优化是在关系代数一级上进行的。而语义优化是利用语义信息将查询转换成语义上等价的、处理开销较小的一查询。设Q_1 和Q_2 为

$$Q_1 = P(A_1, \cdots, A_k), \quad Q_2 = R(A_1, \cdots, A_k)$$

其中$A_i (i = 1, 2, \cdots, k)$ 是查询涉及的关系的属性，A_i 定义在值域D_i 上，$D = D_1 \times D_2 \cdots \times D_k$, I 是$(A_1, \cdots, A_k)$ 的符合语法规则的允许解释的集合，显然$I \subseteq D$, 于是，

Q_1和Q_2语义等价 $\Longleftrightarrow$ 对每个$(A_1, \cdots, A_k) \in I$, Q_1和Q_2同真假

Q_1和Q_2逻辑代价等价 $\Longleftrightarrow$ 对每个$(A_1,\cdots,A_k)\in D, Q_1$和$Q_2$同真假

两个查询若逻辑代数等价，则一定语义等价。反之不然。在DDB 中，关系的分割条件实际上是对各关系片的某一属性的取值范围进行限制，它相当于完整性约束条件。

基于上述思想，LSZ 对查询先作语义优化。语义优化程序包括语义信息(组织成语义规则库)，一组启发式规则以及基于这组规则的变换机制[8]。具体实现采用变换方法，用变换规则表示知识，其形式为$A\Rightarrow B$。规则左边A变换成右边B，而不改变其语义。例如语义信息:

S. dept = "CS" $\longleftrightarrow$ S. sno $\in$ [350000, 390000]
SC. cno $\in$ [100, 200] $\rightarrow$ S.dept ='CS'
S. age $\in$ [10, 50]

分别表示成:

$$*S.dept='CS' \Longrightarrow S.sno\in[350000,390000]$$
$$S.sno\in[350000,390000]\Longrightarrow S.dept='CS' \quad (1)$$

$$*S.Sno=SC.sno\ \ and\ \ SC.cno\in[100,200]\Longrightarrow$$
$$S.sno=SC.sno\ \ and\ \ S.dept='CS' \quad (2)$$

$$*S.any\Longrightarrow S.any\ \ and\ \ S.age\in[10,50] \quad (3)$$

设查询Q表示成$Q:\{TL|QUAL\}$，则上述变换规则可用于Q的$QUAL$部分变换，下列变换规则可应用$\{TL|QUAL\}$变换。

$$*\{S.any_1|ANY_2\ and\ S.dept='CS'\}\Longrightarrow\{S_1.any_1|ANY2\} \quad (4)$$

$$*\{S.any_1|ANY_2\ and\ S.dept='Math'\}\Longrightarrow\{S_2.any_1|ANY_2\} \quad (5)$$

其中，S被分割成$S_1(dept='CS')$, $S_2(dept='Math')\cdots$等分片。

关系的属性赋于一个优先级，被索引的属性有较高的优先级。这样，若存在语义信息[8]$R_5: index(S.age)$时(表示S在属性age 上建立了索引)，则有上面变换规则(3)。实际上[8]，启发式规则HR_6(利用索引加速局部查询处理) 已推广成启发式规则HR_8(把QUAL 变换为有较高优先级属性的$QUAL^1$)。

由于启发式规则的复杂性和有效性，我们选用过程作为表示法。所有启发式规则写成C语言的函数。查询变换机制是利用启发式规则指导选用哪些变换规则(语义信息) 来作变换的一个迭代过程。

上面是我们在设计实现LSZ 和LSZ2 时，对异构型系统的结构、平行性。与局部系统的连接以及AI 和DB 技术的结合等问题的考虑。实用的异构型系统采用的技术总是从较简单的向较高级的发展，而AI 技术用于DB 系统有着广阔的前景。

致谢: 作者对柳诚飞、绳程弟、陈佩佩、徐革群、顾建明、杜兴、曹春祥以及参加LSZ 和LSZ2 工作的其他同志表示感谢。

参考文献

[1] M. L. Brodie, "未来的智能信息系统: AI 与DB 技术的结合", Readings in Artificial Intelligence and Databases 1988. 中译文在计算机科学, 1989 年第3 期。
[2] 金志权等, "LSZ: 一个异构型分布式数据库系统", 计算机研究与发展, 1989 年第10 期。
[3] 金志权等, "LSZ2 同构/异构型分布式数据库系统的设计与实现", 1989 年全国第八届DB 会议论文集。
[4] 顾建明等, "LSZ 系统中GRDL 到dBASE 的实现及其局部查询优化策略", 小型微型计算机系统, 1988 年11 期。
[5] M. M . Theimer, K. A. Lantz, "Finding Idle Machines in a Workstationbased Distributed System", 8th Inter. Conf. on Distributed Computing Systems, 1988.
[6] M. J. Litzkow et al., "Condor—A Hunter of Idle Workstations", 同上。
[7] B. Napheys and D. Herkimer, "A Look at Loosely-Coupled Prolog/Database Systems", 2nd Inter. Conf. on Expert Database Systems, 1988。
[8] Liu Chengfei et al., "Query Optimization Strategies in Heterogeneous DDBS LSZ", 1988 ISMM Int. Conf. on Mini and Microcomputers.
[9] 柳诚飞, "异构型分布式数据库系统与分布式查询", 南京大学博士论文, 1988 年。
[10] 陈佩佩、纯程弟, "LSZ 系统中GRDL 到Unify 的实现方法", 小型微型计算机系统, 1988 年11 期。
[11] 周晓方、金志权, 柳诚飞, "LSZTM: 一个分布式数据库系统的执程管理", 微型计算机, 1989 年第3 期。
[12] 石树刚等, "局部网分布式数据库的语义查询优化", 微型计算机, 1988 年第4 期。

第三届全国机器学习研讨会(CML'91)
征文通知

中国人工智能学会机器学习学会、中国计算机学会人工智能与模式识别专业委员会定于1991年7月在黑龙江省镜泊湖举行第三届全国机器学习研讨会, 会议由哈尔滨工业大学计算机系洪家荣教授承办。现将有关事项通知如下:

征文范围: 机器学习的一般理论; 学习的认知机制; 示例学习; 解释学习; 类比学习; 观察与发现学习; 遗传算法; 神经网络; 知识获取与知识表示; 机器学习的应用。

论文要求: 1、未在其他刊物或会议上发表过2、文章不超过6000 字3、在正文前附200 字的中英文摘要及关键词4、来稿请注明作者姓名和通信地址。

截稿日期: 1991 年3 月20 日

投稿地址及联系人: 哈尔滨工业大学机器人研究所 刘宁同志

邮政编码 150006 电话 228383-4044 电挂 3300