递归维概念及其操作的研究*)

袁 霖 李战怀 徐秋元
(西北工业大学计算机系 西安 710072)

The Concept of Recursive Dimension and Corresponding Operations

YUAN Lin LI Zhan-Huai XU Qiu-Yuan
(Department of Computer, Northwestern Polytechnical University, Xi'an 710072)

Abstract Dimensions are an essential part in multi-dimensional data model (MDDM) for OLAP. Hierarchical structure of dimension allows the user to study measures in different levels of details. This paper focuses on a special kind of dimension with an unbalanced-hierarchy designed to support self-referencing dimension table in data warehouses, called recursive dimension, which is largely ignored by research area. So we extend MDDM to represent it and corresponding operations. Finally we show how it can be implemented in object-relational DBMS environment.

Keywords Recursive dimension, Multi-dimensional data model, OLAP

1. 引言

数据仓库技术研究如何将分散在企业内部多个异构数据源的操作型数据，经过抽取、清洗转换、集成，以一致的方式进行存储。其实现多基于关系技术，数据被分别存放在维表和事实表之中，以星型或雪花模式存储。而 OLAP(联机分析处理)[1]技术则将数据仓库中的数据以多维方式进行组织并提供多维操作，无论 ROLAP(关系实现)还是 MOLAP(多维实现)都是基于多维数据模型的。

在多维数据模型中，数据被分为两类——作为被分析对象的度量和用于描述度量的若干个维。度量被视为多维空间中的点。维自身具有层次结构，通过模式和实例来描述，图 1、2 分别是商店维的模式和实例。维模式是由若干个粒度等级(Level)所构成，每一个等级相当于一个实体集。维实例是由每一个等级中所包含的维成员的集合以及属于不同等级的维成员之间的映射组成，这种映射是满射的。我们称具有上述结构特点的维为常规维。

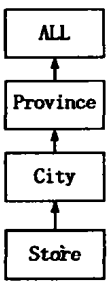


图 1 商店维模式

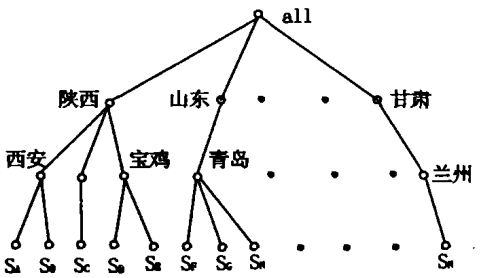


图 2 商店维实例

本文的研究对象是一类具有特殊结构的维，被设计用于支持数据仓库中的自引用维表，如表 1 所示。对于常规维来说，不同的等级被映射到数据仓库维表中的不同属性列上，有多少个等级，在数据仓库的维表中就有多少个不同的属性列与之对应。然而对于递归维，情况有所不同，所有维成员间的关系都是通过相应维表中的两个列定义的，其中的一列提供了维成员自身的唯一性标识，另一列提供了其父成员的标识。如表 1 所示的雇员维表，雇员间的关系是通过“雇员 ID”和“上级雇员 ID”递归定义的，维表中只有一个雇员没有上级成员标识，他是企业的最高领导人。这种自引用维表将产生一个非平衡层次，具有公共的根结点，所有分枝并非中止于同一等级上，如图 3 所示。递归维与常规维的另一个区别在于递归维中所有维成员来源于同一实体集。

表 1 数据仓库中的雇员维表

雇员 ID	姓名	上级雇员 ID
1	张平	Null
2	李云霞	1
3	刘洁	1
4	史忠发	1
5	李敏	2
6	李滔	2

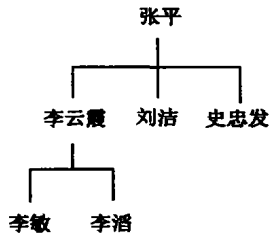


图 3 雇员组织结构

2. 递归维对于多维数据表示和操作所提出的特殊需求

2.1 非叶节点带数据

对于图 4 的雇员组织结构图，在人力资源部对员工的工资进行分析时，对于每一个非叶成员如“李云霞”，除了用于标

*)本文研究得到国家自然科学基金(60073055)、西北工业大学博士论文创新基金资助。袁 霖 博士研究生，主要研究领域为对象关系数据库，数据仓库与 OLAP 技术。李战怀 博士，教授，主要研究领域为数据库理论与技术。徐秋元 教授，主要研究领域为数据库理论与技术。

识其所有直接和间接下属的工资总和外，她本人也有自己的工资，我们称这种现象为“非叶节点带有度量数据”，而非导出数据。在多维模型中，如果同一成员既要标识基础度量数据，又要标识导出的聚集数据，必然要引起语义冲突。

2.2 自定义汇总运算符

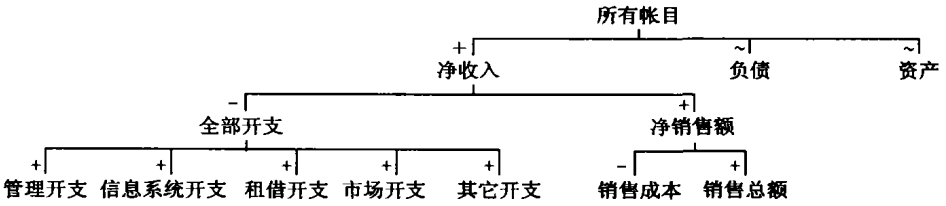


图 4 帐目结构图

图 4 表示某企业财务上的帐目结构，帐目从底层到高层的汇总方式是通过汇总运算符（+、-、×、/、~）按照从左到右的顺序计算得到的，其中‘~’表示不参与汇总。

3. 相关研究

递归维这一概念是由工业界首先提出的，并在其 OLAP 产品中给予了不同程度的支持，在 Microsoft Analysis Services[14]、Heperion Essbase[15]中称之为递归维。然而对于这些 OLAP 产品的内部实现，我们不得而知。

在学术界，“递归维”这一概念却被人们忽视了，我们对现有的多维数据模型进行了调查，其中文[2～7]只能表示常规维及其相应的操作；文[8～9]可以表示某种异构类型的维，其维模式是由多个 level 实体构成的一个非平衡层次，虽然维实例也是一个非平衡层次，但是维成员并非来自同一实体；文[10～12]研究了另一种特殊结构的维，与常规维类似，其维模式由多个 level 实体构成，不同的是维实例可以在某些 level 中出现“漏洞”，导致叶节点到根节点的距离可以不等，具有这种结构特点的维在一些文献[12]或 OLAP 产品[13]中被称为“ragged dimension”。综上所述，现有的多维数据模型都不能表达递归维及相关的多维操作。

为保证聚集语义在 OLAP 环境中的正确性，文[13]对具有“ragged and unbalanced”特点的维层次结构进行设计时应满足的约束条件进行了研究。但这篇文章不是从多维数据模型的角度论述这一概念，更未涉及多维数据的表示和操作。这是迄今为止我们所知的唯一一篇涉及递归维的研究报告。

本文的贡献在于通过对现有的多维数据模型进行扩展的以支持递归维及相关的多维操作，并给出了一个基于对象关系数据库的实现。

4. 一个扩展的多维数据模型

4.1 递归维模式和实例

图 5、6 分别是雇员维（图 3）和帐目维（图 4）的模式示意图，图中——○表示维成员属性。两个模式的共同点在于它们都有一个具有自反联系的等级，如“雇员管理角色”和“帐目”等级，我们称之为“Recursive Level”，通过{tree}或{order_tree}约束使递归等级中的维成员构成一颗树或有序树。同时两种模式也有它们各自的特点：(1)雇员维模式中包含一个用于表示每个雇员个体信息的等级，我们称之为“Primary Level”，它可用于标识来自基础数据源的度量数据，如雇员个体的工资，用于解决“非叶节点带数据”的问题，我们用布尔变量 B_{PL} 表示模式中是否包含这一等级（“Primary Level”与“Recursive Level”之间是一对一的关系，可将二者一并看作一个“非原子”的 Level 实体。独立的等级间应该是一对多的关系）；(2)在帐目维模式中，存在用于表示自定义汇总运算符的成员属性 OP，我们用布尔变量 B_{OP} 表示模式中是否存在这一属性。我们将这两种递归维模式的各自特点综合，得到图 7 所示的通用递归维模式。

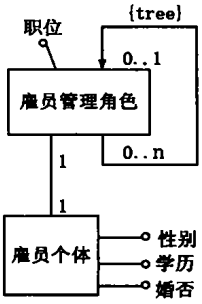


图 5 雇员维模式

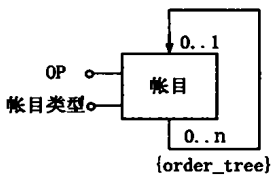


图 6 帐目维模式

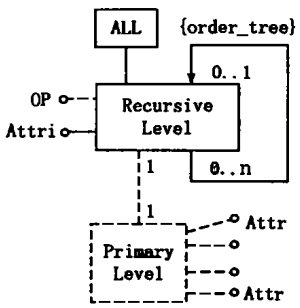


图 7 递归维模式

我们将递归维中用来标识来自基础数据源的度量数据的维成员的集合称为 L_{Bottom} 等级，当 $B_{PL}=true$ 时，L_{Bottom} 代表“Primary Level”；当 $B_{PL}=false$ 时，代表“Recursive Level”中叶节点成员的集合。L_{Bottom} 的引入只是为了描述的方便，本身也不是一个独立的实体。

定义 4.1.1（汇总运算符类型） T_{OP} 为汇总运算符类型，$dom(T_{OP})=\{+,-,*,/,\sim\}$ 是二元运算符的集合，其中+、-、*、/的语义与算术运算符相同，~表示它后面的操作数不参与运算。与算数运算符不同的是，所有汇总运算符的优先级

是相同的。

定义 4.1.2(递归维模式) $DS_f=(Dname, B_{PL}, B_{OP}, Lset, <_R, Aset, AL)$，$Dnam$ 是维名，B_{PL}和 B_{OP}是两个布尔型变量，$Lset=\{ALL, L_{Recursive}, L_{Bottom}\}$，$L_{Bottom}$是一个抽象 level，定义如下：if $B_{PL}=true$ then $dom(L_{Bottom})=dom(L_{Primary})$ else $dom(L_{Bottom})=\{e \mid e\in dom(L_{Recursive}) \wedge IsLeaf(e)=true\}$，其中，函数 $Isleaf(e)$用于判断 e 是否为 $L_{Recursive}$中的叶成员。$<_R$ 是定义在 $L_{Primary}$上的自反关系，$dom(ALL)=all$。Aset 是成员属性[17]的有限集合，$AL: Aset \to Lset$ 是一函数。当 $B_{OP}=true$ 时，$OP\in Aset$ 且 $AL(T_{OP})=L_{Recursive}$。

定义 4.1.3(递归维) 递归维 $D_f=(DS_f, Fset)$。(1)所有属于 $L_{Recursive}$的维成员构成一棵有序树，具有共同直接父成员 $father\in dom(L_{Recursive})$的所有属于 $dom(L_{Recursive})$的子成员构成一个有序链表，$F_{Children}(father)=\{child_i\in dom(L_{Recursive}) \mid i=1,\cdots,n\}$且 $F_{next}(child_n)=child_{n+1}$；(2)设 $root$ 是根节点成员，all 是 $root$ 的别名，表示为 $\rho(root)=all$；(3)当 $Bool_{PL}=true$ 时，令 $F_{PR}: dom(L_{Primary}) \to dom(L_{Recursive})$，$F_{RP}$是 F_{PR}的反函数；(4)$Fset$ 是函数映射的集合，$Fset=\{F_{Children}, F_{next}, F_{PR}, \rho\}$。

4.2 递归维的函数操作

定义 4.2.1(descendants 函数) $descendants(e,n,m)$，其中 $e\in dom(L_{Recursive})$，如果 n、m 是非负整数，$n\leqslant m$。则函数的输出是以 e 为根节点的子树中位于从 e 向下第 n 层到第 m 层中属于 $dom(L_{Recursive})$的维成员的集合。如果 m 的值为 *，则函数的输出是以 e 为根节点的子树中位于从 e 向下第 n 层到叶节点成员的集合。

定义 4.2.2(tree 函数) tree(e)，其中 $e\in dom(L_{recursive})$，返回递归维中以 e 为根结点的子树。

4.3 度量及多维数据集

定义 4.3.1(度量模式) 度量模式是一个 3 元组 $MS=(Mname, MT, O)$，其中 $Fname$ 是度量的名称，MT=(idT, MVT) 是度量单元(cell)的类型，idT 是标识类型，MVT 是度量值的类型。O 是可用于 MVT 上的聚集函数的集合。

定义 4.3.2(度量域) MS 是度量模式，每一个度量单元是一个以 MT 为型的对象，存在双射函数 $rep: dom(MT) \leftrightarrow dom(idT)$，度量域 $dom(M)\subseteq dom(MT)$。

定义 4.3.3(多维数据集模式) 多维数据集的模式是一个三元组 MDS=(DSset, G, MS)，MS 是度量模式，DSset 是维模式的集合 $DSset=\{DS_i \mid 1\leqslant i\leqslant n\}$，$G=\{L_i \mid 1\leqslant i\leqslant n\}$用来表示多维数据集的粒度，其中 $L_i\in level(DS_i)$。对于递归维 $level(DS_f)=\{L_{Bottom}, L_{Recursive}, ALL\}$。

定义 4.3.4(多维数据集) 多维数据集是一个五元组 $MD=(MDname, MDS, Dset, M, F_{ML})$，MDS 是多维数据集的模式，M 是以 MS 为模式的度量，Dset 是以 DSset 为模式的维的集合；对于 n 维多维数据集，$MD=\{\langle e_1, e_2, \cdots, e_n, m\rangle \mid m\in dom(M) \wedge e_i\in dom(L_i), i=1,\cdots,n\}$，$ML_i=\{\langle m, e_i\rangle \mid \exists \langle e_1, e_2, \cdots, e_i, \cdots, e_n, m\rangle \in MD\}$，$F_{ML}=\{ml_i, 1\leqslant i\leqslant n\}$。

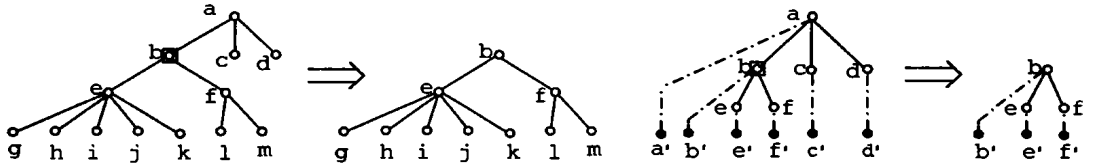
4.4 对多维操作的扩展

假设被操作的 n 维数据集为：

$MD=(MDname, MDS, Dset, M, F_{ML})$，$MDS=(DSset, G, MS)$，$Dset=\{D_1, \cdots, D_i, \cdots, D_n\}$，结果多维数据集为 $MD'=(MDname', MDS', Dset', M', F_{ML'})$，$MDS'=(DSset', G', MS')$。

4.4.1 作用在递归维上的对多维数据集的限定操作

作用在常规维上的限定操作在文[17]中有详细的论述。作用在递归维上的限定操作与之类似，可表示为 $\sigma[P](MD)=MD'$，P 是作用在 MD 上的限定谓词，用文法产生式表示为：$P\to P_{D_i.L_j} \mid P_{D_i.L_j.A_K}$，$P\to P\wedge P$。$P_{D_i.L_j}$、$P_{D_i.L_j.A_K}$是分别作用在维等级、成员属性上的谓词。需要说明的是，限定操作只是消除那些不满足限定谓词的多维数据，并不改变度量值。图 8 是基于维成员的限定操作的两个例子，分别针对两种不同结构的递归维。



(1) $\sigma[L_{Recursive}=b]$, $B_{PL}=false$　　(2) $\sigma[L_{Recursive}=b]$, $B_{PL}=true$

图 8 递归维上基于维成员的限定操作

4.4.2 基于递归维的汇总函数

为了定义基于递归维的多维上翻操作，我们先定义两个维汇总函数，类似于传统的聚集函数，不同的是聚集函数与维结构无关、接受一个值的集合、输出一个标量值，而汇总函数与递归维的结构相关、输入是一个关系的集合，输出的也是一个关系的集合，定义如下：

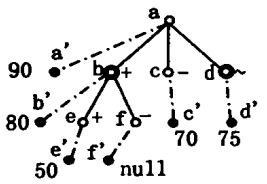


图 9 汇总运算(B_{PL}=true)

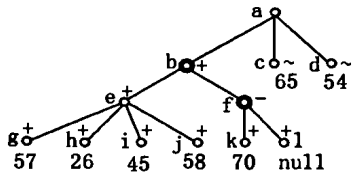


图 10 汇总运算(B_{PL}=false)

定义 4.4.1(基于汇总运算符的汇总函数) $AggOP[OP, E']ML_{Bottom}=M'E'$，OP 是 $L_{Recursive}$上的属性，其类型为 T_{OP}；$E'\subseteq dom(L_{Recursive})$；我们将每一个 $e\in dom(L_{Bottom})$都与一个值 m 关联，用 $e.m$ 表示，满足 $m\in dom(M)\vee m=null$。令 $ML_{Bottom}=\{\langle m, e\rangle \mid e\in dom(M)\wedge e.m\neq null\}$。

对于所有，$e\in dom(L_{Recursive})$，$e.m$ 定义如下：

(1) 当 $B_{PL}=false$，则 $e.m$ 递归定义如下：if $e\in dom(L_{leaf})$, $e.m=ML_{Bottom}(e)$ else $e.m=0(e.child[1].OP)\ e.child[1].m\ (e.child[2].OP)\ e.child[2].m\cdots(e.child[n].OP)\ e.child[n].m$

表达式中的运算符被置于()之中，对于 $e.child[i]=null$ 时，按如下方式处理：当 $e.child[i].m$ 的运算符为+、−、~时，$e.child[i].m=0$；否则 $e.child[i].m=1$。

(2) 对于 $B_{PL}=true$，则 $e.m$ 递归定义如下：if $e\in dom(L_{leaf})$, $e.m=ML_{Bottom}(F_{RP}(e))$ else $e.m=F_{RP}(e)(e.child$

[1].OP) $e.child[1].m\cdots(e.child[n].OP)\ e.child[n].m$

对于 $e.child[i]=null$ 时的处理方式同上。

输出结果为：$M'E'=\{\langle m',e'\rangle \mid e\in E\wedge e'.m'\neq null\}$。

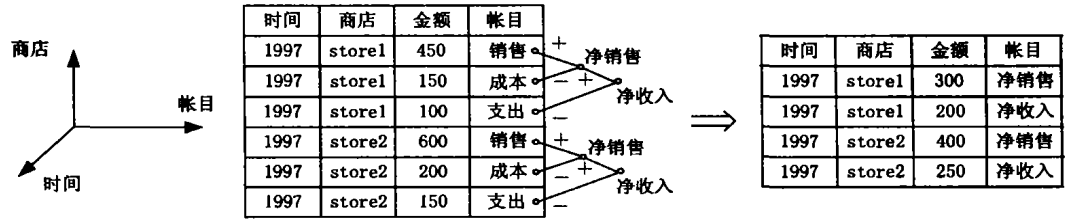
定义 4.4.2(基于聚集函数的汇总函数) $AggFun[E']ML_{Bottom}=M'E'$，其中 E、ML、$M'E'$ 同定义 4.2.2；聚集函数 AggFun 可以是 sum、count、max、min、avg 也可以是用户自定义的。$e.m$ 定义如下：

(1)对于 $B_{PL}=false, e.m=Fun(\{m' \mid \exists\ e'\in tree(e)\wedge e'\in dom(L_{Bottom})\wedge m'=e'.m\})$

(2)对于 $B_{PL}=true, e.m=Fun(\{m' \mid \exists\ e_i\in tree(e)\wedge e'\in dom(L_{Bottom})\wedge m'=F_{RP}(e').m\})$

我们定义的这两个汇总函数，由于充分利用了递归维中度量数据对于父子成员之间的可导出关系，一次计算多个成员的汇总值，与一次输出一个标量值的聚集函数相比，显然是高效的。另外，基于自定义汇总运算符的汇总函数，由于汇总表达式的值依赖于计算路径上每一点的运算符，因此比聚集函数有更强的计算能力。

4.4.3 基于递归维的上翻操作 将上节中定义的两个汇总函数 AggOP 和 AggFun 应用于多维数据集的上下文环境之中，就是基于递归维的两个上翻操作符。具体方式为：首先将多维数据集中的{〈度量，递归维成员〉}按照其它各维进行分组，如图 11 所示的多维数据集中有两个分组〈1997，store1〉和〈1997，store2〉，在每一个分组中用汇总函数进行计算，即为上翻操作。对基于递归维的上翻操作，我们总假设多维数据集从 L_{Bottom} 上翻到 $L_{Recursive}$。



时间	商店	金额	帐目
1997	store1	450	销售
1997	store1	150	成本
1997	store1	100	支出
1997	store2	600	销售
1997	store2	200	成本
1997	store2	150	支出

时间	商店	金额	帐目
1997	store1	300	净销售
1997	store1	200	净收入
1997	store2	400	净销售
1997	store2	250	净收入

图 11 基于递归维的上翻操作

定义 4.4.3(基于递归维的上翻操作) 将多维数据集，基于递归维 $D_f\in Dset$ 从等级 L_{Bottom} 上翻到 $L_{Recursive}$，其它维等级保持不变，满足 $L_i={}_LL_i, i=1,\cdots,n$ 且 $i\neq f$。参数 $E'\subset dom(L_{Recursive})$，OP 是汇总运算符，AggFun 是汇总函数。根据作用在度量上的两个汇总函数 AggOP、AggFun 分别定义如下两个上翻操作：

(1)$RollupOP[E',OP]MD=MD'$

$F_{ML'}=\{(m',\langle e_1,\cdots,e'_f,\cdots,e_n\rangle) \mid \exists\ \langle e_1,\cdots,e_f,\cdots,e_n\rangle$
$(\{m',e'_f\})=AggOP[OP,E'](\{(m,e_f)\}))\wedge\langle e_1,\cdots,e_f,\cdots,e_n\rangle$
$=F_{ML}(m)\wedge m\in dom(M)\wedge ML_{Bottom}(m)=e_f\wedge e'_f\in E'\}$

$dom(M')=\{m' \mid (m',e')\in F_{ML'}\}$。

(2)$RollupFun[E',AggFun]MD=MD'$

$F_{ML'}=\{(m',\langle e_1,\cdots,e'_f,\cdots,e_b\rangle) \mid \exists\ \langle e_1,\cdots,e_f,\cdots,e_n\rangle$
$(\{m',e'_f\})=AggFun[E'](\{(m,e_f)\}))\wedge\langle e_1,\cdots,e_f,\cdots,e_n\rangle=F_{ML}(m)\wedge m\in dom(M)\wedge ML_{Bottom}(m)$
$=e_f\wedge e'_f\in E'\}$

$dom(M')=\{m' \mid (m',e')\in F_{ML'}\}$。

5. 递归维的实现

5.1 递归维结构的对象关系表示

递归维在关系数据库中常用递归关系表表示，SQL2 对于递归关系的查询都要通过表自身的连接，查询效率低，而对象关系系统所特有对象标识引用可以方便地实现导航式的操作。对于图 12 所示的递归维实例(1)，在存储时为保证每条记录的长度相同，将其以(2)所示的方式重新组织，箭头表示对象标识引用，不同线形表示不同的类型引用，其存储模式如(3)所示。

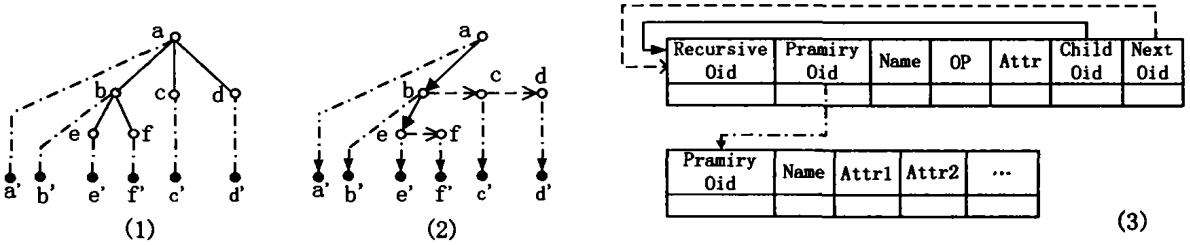


图 12 递归维在对象关系数据库中的存储模式

5.2 汇总函数的实现

(1)$AggOP[OP,E']ML_{Bottom}=M'E'$的实现：

首先将 E' 中的维成员组织到若干个不相交的子树中，然后依次对每棵子树进行后序遍历生成计算表达式 EXP。对于每一个 $e'\in E'$，在计算表达式中加入 $\uparrow e'$ 表示输出〈e'，m'〉值。

例如对于图 9 所示的递归维结构，E'={b,d}，则生成两棵计算子树，表达式为：$EXP=b'e'+f'-\uparrow bd'\uparrow d$。对于递归维结构，E'={b,f}，只有一棵计算子树，表达式为 $EXP=00g+h+i+j++0k+l+\uparrow f-\uparrow b$。

(2)$AggFun[E']ML_{Bottom}=M'E'$的实现：

与 AggOP 的实现相类似，不同的是表达式中的运算符

(下转第 20 页)

务迁移给其它工作站。

最后，n-HLBSM 具有很好的可扩展性。因为 n-HLBSM 的第 n 任务调度层实行动态调度策略，每个任务调度结点只要知道邻接任务调度结点的局部系统的信息，就能够执行负载平衡调度策略，这样就能够适应分布式系统不断扩展的需要，符合条件的任何任务调度结点都可以随时成为第 n 任务调度层的成员。可扩展性可以克服静态调度策略的固有缺点，即要求调度服务器掌握所有任务调度结点的当前负载情况和工作状态，这种要求的另一缺点是随着系统规模的扩大，静态调度算法的效率将明显下降。

结束语 近年来，负载平衡调度问题的研究成果层出不穷，人们采取不同的方法，从不同的角度进行了多方位的探讨。分布式任务分配算法大致可分为：基于图论的分配算法、整数规划方法、启发式算法等；负载平衡调度可分：集中式调度和分布式调度；负载平衡调度模型可分为静态负载平衡调度和动态负载平衡调度。本文提出的分层负载平衡调度模型 HLBSM 在易实现性、可扩展性、最优性、智能性和容错性等综合特性方面优于其它模型。文章给出了 HLBSM 系统的实现规则，今后将进一步研究实现智能 HLBSM 系统。

参 考 文 献

1 Willebeek-LeMair M H. Strategies for Dynamic Load Balancing on Highly Parallel Computers. IEEE Transactions on Parallel and Distributed System, 1993,4(9):979～993

2 Liao C-J. Tree-Based Parallel Load-Balancing Methods for Solution-Adaptive Finite Element Graphs on Distributed Memory Multicomputers. IEEE Transactions on Parallel and Distributed System, 1999,10(4):360～370

3 Hui C-C, Chanson S T. Hydrodynamic Load Balancing. IEEE Transactions on Parallel and Distributed System, 1999,10(11):1118～1137

4 Das S K. Parallel Processing of Adaptive Meshes with Load Balancing. IEEE Transactions on Parallel and Distributed System, 2001,12(12):1269～1279

5 Chow K-P, Kwok Y-K. On Load Balancing for Distributed Multiagent Computing. IEEE Transactions on Parallel and Distributed System, 2002,13(8):787～801

6 秦忠国，姜弘道. 静态负载平衡问题的表示与算法. 计算机科学，1998,25(2):95～97

7 李冬梅，施海虎. 多处理机系统的负载平衡调度算法. 软件学报，2003 (6)

8 林成江，李三立. 一种可适应的分布式动态负载平衡策略及其仿真. 计算机学报，1995,18(10):721～739

9 陈华平. 分布式动态负载平衡调度的一个通用模型，软件学报，1998,9(1):25～28

10 钟求喜，谢涛，陈火旺. 任务分配与调度的共同进化方法. 计算机学报，2001,24(3):308～314

11 鞠九滨，杨鲲，徐高潮. 使用资源利用率作为负载平衡系统的负载指标. 软件学报，1996,7(4):238～243

(上接第 10 页)

全部相同。

5.3 基于递归维的上翻操作的实现

设 e_f 为递归维成员，首先将多维数据集$\{\langle e_1,e_2,\cdots,e_{n-1},e_f,m\rangle\}$中的$\{\langle e_f,m\rangle\}$按照$\{\langle e_1,e_2,\cdots,e_{n-1}\rangle\}$进行分组，然后在每一个分组中对表达式 EXP 进行求值，算法[16]如下：

从左到右一次扫描表达式中的每一个符号，每遇一维成员 e，将 e.m 值压入栈顶暂存起来(对于 null 值的处理见定义 4.2.3)；每遇一个运算符，就从栈顶弹出相应的运算对象进行相应的运算，并将结果压回栈中；每遇一个 ↑e，输出$\langle e_1,e_2,\cdots,e_{n-1},e,m'\rangle$到结果集，$m'$为栈顶值。

基于递归维的上翻操作难于直接用 SQL 实现，因此我们采用'过程化的 SQL'或其它能够编写"存储过程"的语言来实现。

结论 本文将递归维概念引入多维数据模型，并对相关的多维操作进行了扩展，提高了模型对多维数据的表达能力和操作能力。本文提出了两个适用于递归维的、用于取代聚集函数的、输出为集合关系的汇总函数，用于支持基于递归维的对于多维数据集的上翻操作，提高了多维复杂计算的能力。由于对象关系数据库中所特有的对象标识引用也使得基于递归维的导航式的操作更为高效。

参 考 文 献

1 Pendse N. What is OLAP?. http://www.olapreport.com/fasmi.htm

2 Cabibbo L, Torlone R. A Logical Approach to Multidimensional Databases. In:Proc. of the EDBT 1998

3 Agrawal R, Gupta A, Sarawagi S. Modelling Multidimensional Databases. In:Proc. of the ICDE 1997

4 Li C, Wang X S. A data model for supporting on-line analytical processing. In: Proc. Conf. on Information and Knowledge Management, Nov. 1996

5 Gyssens M, Lakshmanan L V S. A Foundation for MultiDimensional Databases. In:Proc. of the VLDB 1997

6 Vassiliadis P. Modeling multidimensional databases, cubes and cube operations. In: Proc. of 10 th SSDBM 1998, Capri

7 Lehner W. Modeling Large Scale OLAP Scenarios. EDBT'98

8 Nguyen T B, Tjoa A M, Wagner R R. An Object Oriented Multidimensional Data Model for OLAP. In: Proc. of the First Intl. Conf. on Web-Age Information Management (WAIM'00), Shanghai, China, June 2000

9 Jagadish H V, Lakshmanan L V S, Srivastava D. What can hierarchies do for data warehouses? VLDB99

10 Ragged Dimension Support . http://msdn.microsoft.com

11 Pedersen T B, Jensen C S. Multidimensional Data Modeling for Complex Data. In:Proc. ICDE' 99

12 Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. Carlos A. Hurtado, Alberto O. Mendelzon, ICDT 2001. 375～389

13 Niemi T, Nummenmaa J, Thanisch P. Logical multidimensional database design for ragged and unbalanced aggregation hierarchies. In:D. Theodoratos, et al. eds. DMDW'2001

14 Analysis Service. http://msdn.microsft.com/

15 Essbase. http://www.essbase.com/

16 蒋立源，康慕宁. 编译原理. 西北工业大学出版社，1999. 197

17 袁霖，李战怀. 属性维概念及其操作的研究. 计算机科学，2003(6)