关于多重复制定义的优化传播算法*

者 敬+, 孙玉芳

(中国科学院 软件研究所 中文信息中心,北京 100080)

Optimized Propagation Algorithms for Multi-Replication Definition

ZHE Jing+, SUN Yu-Fang

(Chinese Character Information Center, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: 86-10-62544129, Fax: 86-10-62645414, E-mail: jzhe@sonata.iscas.ac.cn

http://www.iscas.ac.cn

Received 2002-02-26; Accepted 2002-07-02

Zhe J, Sun YF. Optimized propagation algorithms for multi-replication definition. *Journal of Software*, 2003,14(2):230~236.

Abstract: Multi-replication definition (MRD) is a new trend of database replication, but it will increase the propagation costs. In this paper, three optimized propagation algorithms for MRD are presented, D-M, ILS and LIS. D-M Algorithm gets the minimum individual propagation cost by dividing replication objects and merging propagation objects. Based on it, ILS makes the total costs to the least in strict Chain Topology scenarios, and LIS gets optimized total costs in other common situations by decomposing the propagation task. Their correctness and efficiencies are validated both theoretical and experimentally.

Key words: database replication; algorithm; propagation; multi-replication definition

摘 要: 多重复制定义(MRD)虽然是数据库复制发展的一个新趋势,但是它会引起传播开销的增大.提出了关于MRD的3个优化传播算法:D-M,ILS和LIS算法.其中D-M算法通过拆分复制对象和合并传播对象获得最小的单次传播开销.在此基础上,ILS算法在链式结构下使总传播开销最小.而LIS算法在更普遍条件下通过分解传播任务得到优化的总传播开销.理论和实验分别验证了它们的正确性和有效性.

关键词: 数据库复制;算法;传播;多重复制定义

中图法分类号: TP311 **文献标识码**: A

数据库复制是分布式环境中提高系统可用性和可靠性的关键技术.近年来,随着数据库复制在更多重要领域(数据仓库、移动环境、电子商务、嵌入式系统等)的飞速发展,它时刻面临新的挑战.

同一数据源复制到不同节点多个副本的现象十分常见.在复制的早期,各副本内容几乎完全一样.这在当时可以理解,因为那时复制的需求以及处理流程都比较简单[1].不过现在情况已发生变化,出于灵活性和自治性的考虑,可以在同一物理的数据对象上定义不同的复制对象,而每一复制对象又独立拥有自己的多个副本[2].这种现象我们称为"多重复制定义"(multi-replication definition,简称MRD).

* Supported by the National Natural Science Foundation of China under Grant No.19831020 (国家自然科学基金)

第一作者简介: 者敬(1971－),男,陕西汉中人,博士,主要研究领域为大型网络与数据库工程.

与此同时,数据库复制所依赖的网络环境也突破了原有的限制,包括公共 WAN、Internet 和无线网络等慢速介质如今也经常使用,这时的通信容量往往成为制约复制性能的瓶颈.而在 MRD 时,由于副本间存在交迭(水平的或垂直的)数据是不可避免的,传播开销势必会因重复数据的传输而增大.

为降低传播开销,许多复制系统通常使用传统的压缩、预取(pre-fetch)等方法,而从消除重复传输内容入手进行的研究成果尚未见到.Cabinet Replicator 是我们开发的异构数据库复制原型系统,为适应数据库复制的新发展趋势,原型强调了对 MRD 的支持,并从消除重复传输数据入手研究了此情况下的传播开销优化问题.作为研究结果,本文提出了 3 个优化传播算法:D-M,ILS 和 LIS.

1 MRD 描述

文献[3]提到的"单个对象独立复制"(replication-per-object)模型是最适合描述 MRD 的复制模型,本文在它的基础上进行了适当的修改和扩充.

在某一时刻,初始节点 S_0 有 m 个复制对象 $O_1,O_2,\ldots,O_m$,要分别传播到其他 n 个节点 $S_1,S_2,\ldots,S_n$,总传播任务可用一个由(对象,节点)对的序列构成的集合表示.

$$\{(O_i,S_j)\},1\le i\le m,1\le j\le n. \tag{1}$$

如果对每个 O_i 定义一个目标节点集合 $D_i\subseteq\{S_1,S_2,\ldots,S_n\}$,也可表示为

$$\{(O_1,D_1),(O_2,D_2),\ldots,(O_m,D_m)\}. \tag{2}$$

如果对每个 S_j 定义一个接收对象集合 $R_j\subseteq\{O_1,O_2,\ldots,O_n\}$,还可转换为

$$\{(R_1,S_1),(R_2,S_2),\ldots,(R_n,S_n)\}. \tag{3}$$

定义节点连接矩阵 $M=\{(M_{ij})\}$,它可用 Dijkstra 算法[4]获得.M_{ij} 是从 S_i 到 S_j 的最小传播步数,$M_{ij}=1$ 表示 S_i 和 S_j 相邻;$M_{ij}=\infty$表示不能从 S_i 传播到 S_j;其他则表示从 S_i 到 S_j 需通过至少 $M_{ij}-1$ 个中间节点的中转.

2 D-M 算法

2.1 算法结构

首先考虑节点间的单次传播开销,为此引入"传播对象"概念.对任意 O_i 和 O_j,如果 $O_i\cap O_j\neq\varnothing$,则生成一个新传播对象 P_k,令 $P_k=O_i\cap O_j$,同时把 O_i 和 O_j 也转换为相应的传播对象 P_i 和 P_j,$P_i=O_i-P_k$,$P_j=O_j-P_k$;如此反复进行,直到对$\forall P_i\forall P_j,i\neq j,P_i\cap P_j=\varnothing$.为保证能够还原接收的传播对象,为 P_i 另加一个指示所属复制对象的集合 F_i,$\forall P_j\subseteq O_i\ (j\in F_i)$.如此,式(2)进一步转换为

$$\{(P_1,D_1,F_1),(P_2,D_2,F_2),\ldots,(P_k,D_k,F_k)\},\forall P_i\forall P_j,P_i\cap P_j=\varnothing,1\le i,j\le k,i\neq j,k\ge m. \tag{4}$$

根据式(4)的具体做法是,发送时把存在交迭数据的复制对象拆分成彼此无交迭的传播对象,而接收时再把传播对象合并为原来的复制对象.两个步骤分别由 Divide 和 Merge 过程负责,因此称其为 D-M 算法.

Divide 过程和 Merge 过程的伪语言描述如下:

```
PROCEDURE Divide
BEGIN
    P1:= O1; F1:={1}; k:= 2;
    FOR i:=2 TO m DO
        Pk := Oi; Dk:= Di; Fi:={i}
        FOR j:=1 TO i−1 DO
            Ptemp:=Pi∩Pj;
            IF Ptemp= ∅ THEN LOOP;
            Pi:=Pi−Ptemp; Pj:=Pj−Ptemp;
            Pk:=Pi; k:= k+1;
            Pk:=Ptemp; Dk:=Di∪Dj;
```

```
            F_k:= F_i ∪ F_j; k:=k+1;
        NEXT
    NEXT
    SEND {(P,D,F)};
END
PROCEDURE Merge (S_p)
BEGIN
    RECEIVE {(P,D,F)};
    FOR i:=1 TO m DO O_i:=∅
    FOR i:=1 TO Count ({P}) DO
        IF S_p∉D_i THEN LOOP
        FOR j:=1 TO n DO
            IF j∈F_i THEN O_j:= O_j ∪ P_i
        NEXT
        D_i:=D_i－{S_p};
        IF D_i=∅ THEN P_i:= ∅;
    NEXT
END
```

2.2 算法证明

下面用数学归纳法证明 D-M 算法是正确的,而且单次传播开销最小.

证明:(1) 当 m=2,即只有复制对象时,O_1 和 O_2:

如果 $O_1 \cap O_2=\varnothing$,则 $P_1=O_1,F_1=\{1\},P_2=O_2,F_2=\{2\}$.此时 $\{(O_1,D_1),(O_2,D_2)\}$与$\{(P_1,D_1,F_1),(P_2,D_2,F_2)\}$等价,Divide 过程和 Merge 过程显然都正确.由于 F_1 和 F_2 的传输成本可以忽略不记,传播开销不会更小.

反之 $O_1 \cap O_2 \neq \varnothing$,则 Divide 过程用交迭数据生成新的传播对象 $P_3=O_1 \cap O_2$,令 $D_3=D_1 \cup D_2$(表示 P_3 要传播到 O_1 和 O_2 的所有目标节点),$F_3=\{1,2\}$(表示 $P_3 \subseteq O_1 \wedge P_3 \subseteq O_2$),同时令 $P_1=O_1-P_3,P_2=O_2-P_3$,而 $F_1=\{1\},F_2=\{2\}$.此时 $P_1 \cup P_2 \cup P_3=O_1 \cup O_2$,数据是完备的;$P_1,P_2$ 和 P_3 互相的交集为∅,单次传播开销最小.

S_p 的 Merge 过程接收$\{(P_1,D_1,F_1),(P_2,D_2,F_2),(P_3,D_3,F_3)\}$,当 $O_1 \cap O_2 \neq \varnothing$时有 4 种情况:

(a) $S_p \notin D_1 \wedge S_p \notin D_2$,有 $S_p \notin D_3$,说明 S_p 不是 P_1,P_2 和 P_3 的目标节点,不处理它们;

(b) $S_p \in D_1 \wedge S_p \notin D_2$,因为 $D_1 \subseteq D_3$,所以 $S_p \in D_3$;S_p 令 $O_1=P_1 \cup P_3$,并从 D_1 和 D_3 中去除 S_p,如果 D_1 或 D_3 为∅,说明 P_1 或 P_3 没有别的目标节点,于是“吞咽”之;

(c) $S_p \notin D_1 \wedge S_p \in D_2$,有 $S_p \in D_3$;处理同(b),忽略;

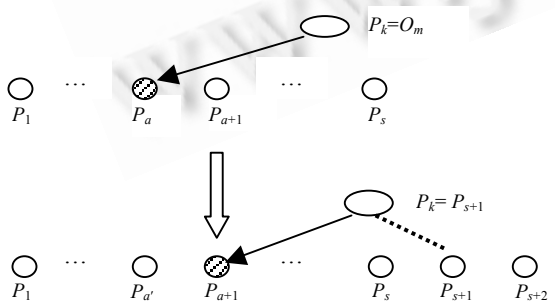

Fig.1 When P_k and P_a are overlapped

图 1 P_k 和 P_a 有交迭时的处理方法

(d) $S_p \in D_1 \wedge S_p \in D_2$,有 $S_p \in D_3$;S_p 令 $O_1 = P_1 \cup P_3$,$O_2=P_2 \cup P_3$,并按相同方式处理 D_1,D_2 和 D_3.

所以当 m=2 时,关于 D-M 算法的结论成立.

(2) 假设 $m=k$ 时 D-M 算法是正确的,且单次传播开销最小,则来看 $m=k$+1 时的情况.

此时 $\{(O_1,D_1),(O_2,D_2),\ldots,(O_m,D_m)\}$ 与 $\{(P_1,D_1,F_1),(P_2,D_2,F_2),\ldots,(P_s,D_s,F_s)\}$等价,如图 1 上半部分所示.

对于新复制对象 O_m,Divide 过程先令 $P_k=O_m$,再把它依次与 $P_1,P_2,\ldots,P_s$ 作比较.一旦发现 P_k 与 $P_a(1 \le a \le s)$ 有数据交迭,则按 m=2 时的方法对它们进行拆分,得到 $(P_{a'},D_a,\{a\}),(P_{s+1},D_k,\{k\})$和$(P_{s+2},D_a \cup D_k,\{a,k\})$.此时有

两个重要推论:(a) $\forall x,P_{a'}\cap P_x=\varnothing\wedge P_{s+2}\cap P_x=\varnothing$;(b) $\forall y\le a,P_{s+1}\cap P_y=\varnothing$,这两个推论保证了无须再回头从 P_1 开始进行处理.从 P_k 中除去交迭部分后(即 $P_k=P_{s+1}$),再把它与余下的 $P_{a+1},\ldots,P_s$ 按同样方法依次比较,如图 1 下半部分所示.最后得到的传播对象序列为 $\{(P_1,D_1,F_1),(P_2,D_2,F_2),\ldots,(P_t,D_t,F_t)\}$,依然有 $P_1\cup P_2\cup\ldots\cup P_t=O_1\cup O_2\cup\ldots\cup P_k$,以及 $\forall P_i\forall P_j,P_i\cap P_j=\varnothing$,数据仍然完备,且单次传输开销最小.

节点 S_p 的 Merge 过程对所有 P_i 按以下情况处理:

(a) $p\notin D_i$,S_p 不是 P_i 的目标节点,无须进行处理;

(b) $p\in D_i$,表明 P_i 包含于至少一个复制对象,根据 F_i 把 P_i 合并到所有相关的复制对象 O_j 中去,并按 m=2 时的方法根据 D_i 决定是否对 P_i 作吞咽处理,这样能够将复制对象还原(否则 Divide 过程不成立).

因此当 $m=k+1$ 时,关于 D-M 算法的结论也成立.

综合(1)和(2),可以证明 D-M 算法是正确的,而且其传播量最小. □

3 ILS 算法

D-M 算法只保证节点间的单次传播开销最小,而要使总传播开销最小就必须考虑传播路由.在一些特定场合,要求按照严格的链式结构来传播[5],即路由是单一线型的而且不存在并行分支.

这时,非相邻节点间的传播必须通过中间节点的转发来实现,如图 2 所示.无论传播路由如何选择,从 S_0 到 S_1 的传播流量总是 $O_1\cup O_2\cup\ldots\cup O_m$,而且每经过一个中间节点流量就减小一部分.由此不难发现获得其中的关键:越是前面的节点,传播流量通过时的净减少量就越大.分析 D-M 算法的 Merge 过程,它一旦发现某个 $D_i=\varnothing$就"吞咽"掉对应的 P_i,而吞咽的数据量就是减少的传播量,因此应选择吞咽量最大的相邻节点作为下一节点.

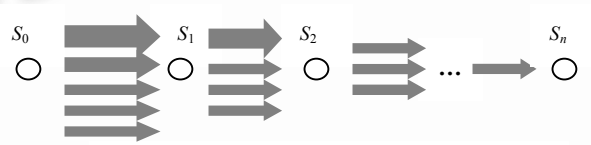

Fig.2 Chain topology propagation

图 2 链式拓扑结构传播

如何确定全部路由有两种方式,一种是采用图遍历方式,S_0 在发送数据前就一次性确定传播路由;另一种采用动态规划方式,各节点都独立地只选择下一节点.前者的优点在于能得到一个精确、无振荡的结果,但在单个对象独立复制模型下,它的自治性和灵活性都比较差,而且在中间节点本身也有传播任务时该路由不是全局最优解.而后者在单个独立复制模型下的适应性极好,也易于实现,而且考虑到了中间节点有传播任务的情况.另外,由于前者吞咽量第二大的节点不一定就是第二中间节点,其时间复杂度最大可达 $O(n!)$,而后者的时间复杂度在每个节点最大为 $O(n)$,全局不超过 $O(n^2)$.

由此提出严格链式结构下的 ILS(independent largest swallowing)算法,这里有两个新表示符:Ils_num 是计算各节点可吞咽量的数组,V 是传播数据已经过的节点集合(为了避免出现兜圈现象).根据最优性原理(principle of optimality),ILS 算法的正确性显而易见,证明从略.

```
ALGORITHM ILS (Sx)
BEGIN
    Ils_max:= 0; S_next:= S1;
    V := V∪ {Sx};
    FOR i:=1 TO n DO Ils_num[i] :=0;
    FOR i:=1 TO Count ({P}) DO
        IF Count (Di)≠1 THEN LOOP;
        j:= Element Value of Di;
        IF Mjx≠1 OR Sj∈V THEN LOOP;
        Ils_num[j]:= Ils_num[j]+cost (Pi);
        IF Ils_num[j]>Ils_max THEN
            Ils_max:= Ils_num[j];
```

```
            S_next :=S_i;
        ENDIF
    NEXT
    SEND {(P,D,F)} TO S_next;
END.
ALGORITHM LIS (S_x)
BEGIN
  FOR i:=1 TO n DO
    m:= Site_list[i]; Lis_Max:= 0; b:= m;
    IF M_xm=1 THEN
        T_m:= {(R_m,S_m)}; Lis_RO[m]:= R_m;
        LOOP;
    ENDIF
    FOR j:=1 TO k DO
        IF cost(Lis_RO[j]∩R_m)>Lis_max THEN
            Lis_max:= cost(Lis_RO[j]∩R_m);b:= j;
        ENDIF
    NEXT
    T_m:= T_m∪{(R_m,S_m)};
    Lis_RO[b]:= Lis_RO[b]∪R_b;
  NEXT
END.
```

4 LIS 算法

如果没有其他强制条件,其实无论从传播开销还是从时间延迟上看,对相邻节点以中转方式来传播都是不明智的做法.为了减少不必要的中转开销,可根据节点连通情况把 S_x 的整个传播工作等价地分解为多个子任务.这里仍依照上节用动态规划只选择首中间节点的方法,分解原则按式(3)表示为:

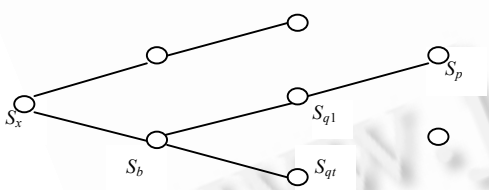

Fig.3 Select a head mid-site for S_p

图 3 为 S_p 选择一个首中间节点

(1) 星式分解:对每个相邻节点 S_b,生成一个传播子任务 T_b= {(R_b,S_b)},M_{xb}=1;

(2) 链式分解:对每个非相邻节点 S_p,选择合适的相邻 S_b 作为首中间节点,并把(R_p,S_p)加入到 T_b,T_b=T_b ∪ {(R_b,S_b)},$1<M_{bp}<\infty$, $p\neq b$.

由于只考虑选择首中间节点,所以欲使 T_b 的传播开销最小,只需使从 S_x 到 S_p 的传播开销最小.假设此时已有 $S_{q1},\ldots,S_{qt}$ 选择 S_b 为传播路由的首中转节点,如图 3 所示.此时 T_b={(R_b,S_b),(R_{q1},S_{q1}),…,(R_{qt},S_{qt})},令 $R_{b'}$= R_b∪R_{q1}∪…∪R_{qt},从 S_x 到 S_p 的传播开销为

$$\text{cost}(R_b\cup R_{q1}\cup\ldots\cup R_{qt}\cup R_p)=\text{cost}(R_{b'}\cup R_p)=\text{cost}(R_{b'})+\text{cost}(R_b)-\text{cost}(R_{b'}\cap R_p).$$

不难看出,应选择使 R_b'与 R_p 交集最大(或并集最小)的 S_b 作为首中间节点,因此称其为 LIS(largest inter-section)算法.新的表示符号有:Lis_RO 是 k 个子任务各自包含的全部接收对象的数组,Site_list 是已知的根据最小传播步数而确定的节点处理顺序链表.由上述分析可知,LIS 算法是正确的,证明从略.

5 实现与测试

异构数据库复制原型 Cabinet Replicator 在具体实现算法时应用了一些小技巧.其中两个重要的技巧是:

(1) 集合运算替代.集合运算在算法经常用到,在通常情况下它的运算成本会很大,而在原型中是用“标识位运算”来代替它的.具体地讲,对元素数量不多或比较固定的集合(节点、交迭的列)用二进制长整数来表示;而对元素变化很大的集合(交迭的行)则事先采取对元素分组加排队、用可变长二进制字符串表示的方法.在假定集合都是预先生成的前提下,“集合运算”成本基本上保持为常量.

(2) 集合排序.D-M 和 LIS 算法都涉及到集合的运算次序,为此在具体运算前先根据集合的元素多少进行排序,这样不仅能提高性能,还能在一定程度上简化算法流程.

Cabinet Replicator 的运行环境包括了若干平台,这里的测试只选择其中一种,但其结论在其他平台也同样有效.具体包括:CPU 为 Pentium Ⅱ 450MHz,15G 硬盘,128M 内存,操作系统为 Red-flag Linux(服务器版 2.4.15),DBMS 为 IBM DB2 (企业通用版 7.0).测试时尽量降低其他进程对系统资源的占用.

先测试 D-M 算法的额外传播开销比例.首先保持其他参数不变,当Σcost(PO)/Σcost(RO)为不同值(10%~90%)时,测试额外开销在实际的总传播开销中所占比例.图 4 是在有 20 个节点、1 000 行数据、每行数据项平均 100 字节时的测试结果.从中可以看出,两者接近于倒数对应关系,这符合理论分析的结论.

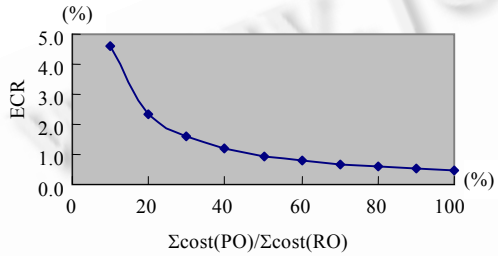

Fig.4 Extra cost ratio of D-M (1)

图 4 D-M 算法的额外开销比(1)

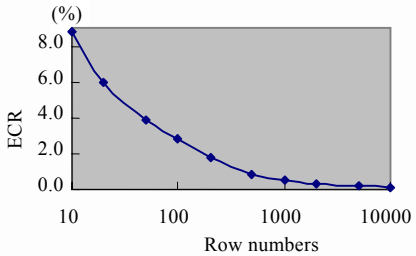

Fig.5 Extra cost ratio of D-M (2)

图 5 D-M 算法的额外开销比(2)

分析 D-M 算法可看以出,数据项行数对传播开销影响较大,因此随后保持其他条件不变和Σcost(PO)/Σcost(RO)为 50%,当数据项行数变化(10~10 000)时,测试额外开销在实际的总传播开销中所占比例,结果如图 5 所示.可以看出,额外开销比与数据项行数接近于负指数对应关系,这也符合理论分析的结论.

再测试 D-M 算法的运行时间.分析 Divide 过程和 Merge 过程,它们的时间复杂度与复制对象数均是 $O(m^2)$ 的关系.图 6 是它们在最坏情况下的运行时间随复制对象数变化(10~100)的测试结果,可以看出与理论分析结论是一致的.

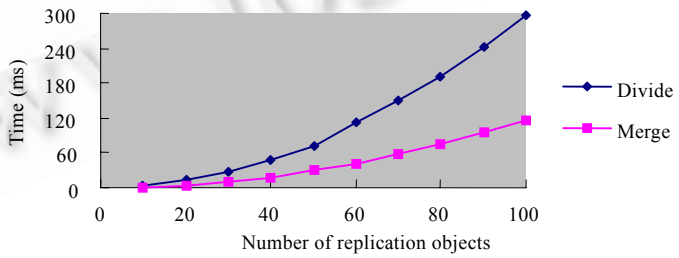

Fig.6 Spending time of Divide/Merge

图 6 Divide/Merge 过程的运行时间

ILS 和 LIS 算法几乎不产生额外传播开销,因此只测试它们的运行时间.经分析可以看出,ILS 的时间复杂度与传播对象数的关系是 $O(m)$,而 LIS 的时间复杂度与节点数的关系是 $O(n)$.但在测试时发现,ILS 的运行时间基本上与传播对象数量变化(10~500)无关,如图 7 所示.而 LIS 的运行时间与节点数(10~150)之间是极小斜率的线性关系,如图 8 所示.这充分表明原型使用的集合运算替代方法是成功的.

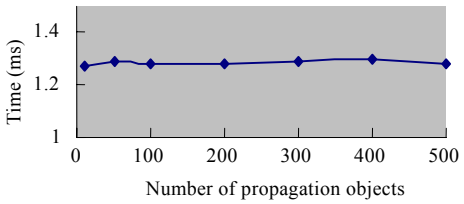

Fig.7 Spending time of ILS

图 7 ILS 算法的运行时间

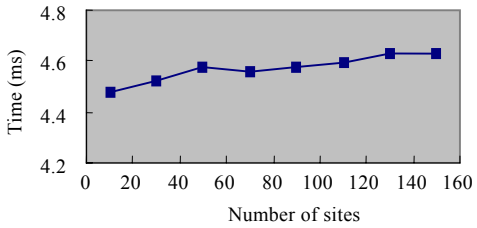

Fig.8 Spending time of LIS

图 8 LIS 算法的运行时间

6 结 论

针对 MRD 这一发展趋势,本文提出 3 个优化传播算法:D-M,ILS 和 LIS.D-M 算法通过拆分复制对象和合并传播对象得到最小的单次传播开销;而其余两个算法根据动态规划原则得到优化的总传播开销,其中 ILS 算法用于严格链式结构,LIS 算法在其他条件下采取传播任务分解的方法.理论和实验分别验证了它们的正确性和有效性.

References:

[1] Ceri S, Houstsma M, Keller A, Samarati, P. A classification of update methods for replicated databases. Technical Report CS-TR-91-1392, Stanford University, 1991.

[2] Hull, R. Managing semantic heterogeneity in databases: a theoretical perspective. In: Abiteboul S, ed. Proceedings of the 6th ACM Symposium on Principles of Database Systems (PODS). New York: ACM Press, 1997. 51~61.

[3] Nicola, M, Jarke, M. Performance modeling of distributed and replicated databases. IEEE Transactions on Knowledge & Data Engineering, 2000,12(4):645~672.

[4] Rosen, KH. Discrete Mathematics and Its Applications. 4th edition. New York: McGraw-Hill Press, 1998.

[5] Hwang, SY, Lee KS, Chin, YH. Data replication in a distributed system: a performance study. In: Wagner R, Thoma H, eds. Proceedings of the 7th International Conference on Database and Expert System Applications. New York: Springer-Verlag, 1996. 708~717.