多物化视图并行增量保持三阶段模式*

王腾蛟[1] 王海洋[2] 洪晓光[2] 董继润[2]

[1](北京大学计算机科学技术系 北京 100871)
[2](山东大学计算机科学系 济南 250100)

摘要 文章提出了一种基于并行流水线处理方式的多物化视图(materialized views)增量保持方法,即在先进行筛选的条件下,利用归类拓扑排序,将所有物化视图进行分类,使每一类中的视图之间没有嵌套定义关系.最后对每一类中的所有视图利用基于信号量控制机制的流水线模式并行处理,以达到对多物化视图实现增量保持的目的.

关键词 物化视图,拓扑排序,流水线,数据仓库,分布式数据库.

中图法分类号 TP311

分布式数据库和数据仓库中的物化视图是将对基本数据库的某次查询结果或基本视图物理地存储在本地工作站上,因而当以后再做有关的查询时,可以不必访问基本数据库而在物化视图中直接得到结果.目前,这方面的研究[1,2]主要集中在单物化视图的增量保持讨论上,而应用中大量多物化视图的增量保持问题需要我们加以处理,尤其是当多个物化视图之间存在联系时,这类问题显得更为重要.故此,本文提出了一种新的多物化视图增量保持三阶段模式.

1 物化视图的增量保持

首先,我们利用文献[2]中的概念和术语来描述单物化视图的增量保持方法.$R_1, R_2, \ldots, R_n$代表基本数据库中的关系;C是物化视图的选择-投影-连接(SPJ)定义;INS(R_u, T)表示将元组集合T插入到关系R_u中;DEL(R_u, C_d)表示将关系R_u中满足条件C_d的元组删除;A代表出现在视图V中的属性集;A^+表示由A唯一确定的属性集;$\alpha(C_d)$代表C_d中使用的变量集合;$\alpha(C)$代表C中使用的变量集合.

定义1(Z-延伸视图,V^z). V^z是由视图V经过如下变换得到的V的延伸:将视图V追加属性集$Z=(\alpha(C_d)\cup\alpha(C))-A^+$,相应地,每个元组$t\in V$追加$|Z|$个不同的变量$z_k, 1\leqslant k\leqslant |Z|\times|V|$. z_k称做Z-变量.

定义2(关系R_i的Z-投影:R_i^z). 给定由C定义的视图V,关系R_i的Z-投影:$R_i^z=\Pi_{G_i}(V^z)$,其中$G_i=Z=(\alpha(C_d)\cup\alpha(C)\cup A^+)\cap\alpha(R_i), 1\leqslant i\leqslant n$.

定义3(投影-连接映射(PJM)). Z-延伸视图V^z的投影-连接映射定义为PJM$(V^z)=R_1^z\times R_2^z\times\ldots\times R_m^z$.每一个元组$\mu\in$PJM$(V^z)$,一定属于下面中的一类,$C(\mu)$是使用$\mu$计算的视图定义.

对μ中Z-变量进行某种实例化后,$C(\mu)$是可满足的,而且

(1) 将μ投影到不同属性上得到的元组在物化视图V中;或者,

(2) 将μ投影到不同属性上得到的元组不在物化视图V中.

对μ中Z-变量无论进行何种实例化后,$C(\mu)$都不会满足.这样的μ可在PJM中删除.

属于上述(1)类型的所有元组称做$pos(V)$,即正确信息;属于上述(2)类型的所有元组称做$neg(V)$,即否定

* 本文研究得到山东省自然科学基金资助.
作者王腾蛟,1973年生,博士生,主要研究领域为数据库管理系统.王海洋,1965年生,教授,主要研究领域为数据库,管理信息系统.洪晓光,1964年生,副教授,主要研究领域为并行/分布式数据库.董继润,1935年生,教授,主要研究领域为数据库,管理信息系统.

本文通讯联系人:王腾蛟,北京 100871,北京大学27楼326

本文1998-10-06收到原稿,1998-12-21收到修改稿

信息，且用 n_i 来编号.

定义 4(结合信息：$L(V)$). $L(V)$是由 Z-延伸视图 V^z 的元组得到的正确信息与由 $neg(V)$中的元组得到的否定信息的结合. 即由定义 C 和视图 V 得到的信息的汇总.

引理 1. 给定非空视图 V，它的 Z-延伸 V^z 和 PJM(V^z)，对 R_1 应用删除条件 C_D 后，元组 $t^z \in V^z$ 仍留在视图中当且仅当 $L(V) \Rightarrow \exists H:[R_1(\Pi_{R_1}(t^z)) \wedge \ldots \wedge R_n(\Pi_{R_n}(t^z)) \wedge C(t^z) \wedge \neg C_D(\Pi_{R_1}(t^z))]$，其中 H 为元组 t^z 中出现的 Z-变量集合，R_i 是指结果出现在视图中的某个基本关系.

引理 2. 给定非空视图 V，它的 Z-延伸 V^z 和 PJM(V^z)，对 R_1 应用删除条件 C_D 后，元组 $t^z \in V^z$ 应在视图中删去当且仅当 $L(V) \Rightarrow \forall H:\neg [R_1(\Pi_{R_1}(t^z)) \wedge \ldots \wedge R_n(\Pi_{R_n}(t^z)) \wedge C(t^z) \wedge \neg C_D(\Pi_{R_1}(t^z))]$，其中 H 为元组 t^z 中出现的 Z-变量集合，R_i 是指结果出现在视图中的某个基本关系.

定理 1. 有条件自治可计算的必要条件是，如果一个删除操作是有条件自治可计算的[12]，则每一个元组 $t \in V$ 一定满足引理 1 或引理 2 中的条件.

插入与删除基本思路相同，还要考虑基本数据库中没有表现在物化视图中的元组.

2　多物化视图环境中的增量保持三阶段模式

三阶段模式包括筛选、分类管理和并行流水线处理. 其中保持单个视图时利用了第 1 节中提到的处理方法，并在流水线模式中利用 $L(V)$优化管理机制进行优化处理. 假设本地工作站上有物化视图 M 个，记做 V_1，V_2，…，V_M. 对数据库的修改 U 包含 δ_1，δ_2，…，δ_n.

2.1　筛选最小子集

筛选最小子集，即在 V_1，V_2，…，V_M 中删除本次修改 U 明显不会影响到的视图.

```
min_subset={V1,V2,...,VM}
unchange=∅
updated=λ(δ1)∪λ(δ2)∪...∪λ(δn)          /*λ(δi)表示被 δn 修改的关系*/
for i=1 to M loop
    if λ(C(Vi))∈updated then            /*λ(C(Vi))表示视图 Vi 定义中使用的关系*/
      unchange=unchange∪{Vi}
    end if
end loop
min_subset=min_subset-unchange
```

2.2　归类拓扑排序

假设由 2.1 节得到的最小子集中的物化视图有 m 个($m \leqslant M$)，我们重新将其标记为 V_1，V_2，…，V_m. 这 m 个物化视图中可能还存在定义嵌套关系，因此有必要对最小子集中的视图进行分类，主要算法描述如下.

输入：有向图 $G=\langle V,E\rangle$.

输出：若干有序分类集合 class.

```
top:=null;
for i=1 to n do                  /*建立一个无先驱顶点(内次为0)的栈.*/
    if i.count=0 then stack:=top; top:=i;
endfor
    j:=1;
    do while top<>null   /*将栈内的顶点归为一类输出，再建立新的栈，直到新栈为空*/
class[j]:=∅;
    do while top<>null
      del_point:=top; top.count:='*'; top:=top.stack;
```

```
        class[j]:=class[j]∪{del_point};
        ptr:=del_point.link;
        do while ptr<>null
          K:=ptr.vertex; K.count:=K.count-1; ptr:=ptr.link;
        enddo
        j=j+1;
      enddo
      top:=null;
      for i=1 to n do
        if i.count=0 then i.stack:=top; top:=i;
      endfor
    enddo
```

对有 n 个顶点和 e 条边的有向图，搜索入度为 0 的时间为 $O(n)$，若有向图无环，入度减 1 的操作在外层 do while top<>null 语句中共执行 e 次，所以总的时间复杂度为 $O(n*e)$.

2.3 并行流水线处理模式

经过上面的处理后，对于某个 class[i]中的顶点集(视图集)，各顶点之间已没有嵌套定义关系，于是，我们提出了一种流水线处理模式来并行保持某个 class 集中的视图.

设某个 class[i]中的顶点(物化视图)有 m 个，$U=\{\delta_1,\delta_2,\ldots,\delta_n\}$ 是对数据库的修改. 我们把根据 δ_i 保持 V_j 的过程定义为 UPDATE(δ_i,V_j)，在这种模式中，每个 UPDATE 过程处理时间并不相同. 我们设计了一种基于并行算法的信号量控制机制来保证正确的运行.

V_1 的算法.

```
初始化: for each δi∈U do
            commit UPDATE(δi,V1);
            send DONE(δi,V1) message to V2;
        end for;
        send FINISH message to V2;
para begin
    receiving ACK message from V2 do;
    stop itself algorithmic execution;
para end.
```

一般视图 V_u($1<u\leqslant m$)的算法.

```
初始化: committed:=0;
para begin
    receiving DONE(δi,Vu-1) message from Vu-1 do;
        L1: if committed<i-1 then goto L1;
        commit UPDATE(δi,Vu);
        committed:=committed+1;
        if Vu≠Vm then send DONE(δi,Vu) message to Vu+1;
para end;
para begin
    receiving FINISH message from Vu-1 do;
        L2: if committed<n then goto L2;
        if Vu≠Vm then send FINISH message to Vu+1;
```

else send ACK message to V_{m-1};

para end;

para begin

receiving ACK message from V_{u+1} do;

send ACK message to V_{u-1};

stop itself algorithmic execution;

para end.

运行该算法时，若每个 UPDATE 过程处理时间相同，记为 Δt_0，流水线上完成 n 个任务的时间为 $T=n\Delta t_0-(m-1)\Delta t_0$，若不相等，则 $T=\sum_{i=1}^{m}\Delta t_i+(n-1)\Delta t_j$，$\Delta t_j$ 为最慢一段所需时间.

注意到 $L(V)$ 的计算，仅利用了当前物化视图 V，因此，若 UPDATE(δ_i,V_u) 没有使 V_u 发生变化，那么在执行 UPDATE (δ_{i+1},V_u) 时，就不必再重新计算 $L(V_u)$. 我们建立 $L(V)$ 优化管理机制来随时监测各个处理器上 UPDATE(δ_i,V_u) 的结果，从而避免了重复计算.

3 结　论

本文提出了一种多物化视图增量保持三阶段模式，这种模式能够根据多物化视图之间的内在联系进行分类处理，并有效利用了部分信息保持物化视图的方法，减少了分布式数据库及数据仓库中的通信费用和维护代价.

参考文献

1 Ashish Gupta, Blakley J A. Using partial information to update materialized view. Information Systems, 1995, 20(8): 641～662

2 Tompa F W, Blackley J A. Maintaining materialized view without accessing base data. Information Systems, 1988, 13(4): 393～406

A Three-stage Model of Incremental Maintenance of Multi-materialized Views

WANG Teng-jiao[1]　WANG Hai-yang[2]　HONG Xiao-guang[2]　DONG Ji-run[2]

[1](*Department of Computer Science and Technology　Beijing University　Beijing*　100871)

[2](*Department of Computer Science　Shandong University　Ji'nan*　250100)

Abstract　A new method is presented in this paper, which is about incremental maintenance of multi-materialized views based on parallel pipeline process. In this process, the authors classify all the materialized views first by using choosing methods, then by classified topological sorting, so that no nested definitions of relations exist among these views. Finally, in order to maintain the incremental multi-materialized views, they process in parallel all the views by utilizing pipeline model of semaphore process mechanism.

Key words　Materialized views, topological sorting, pipeline, data warehouse, distributed database.