愛知工業大学研究報告
第26号B 平成3年

論文

Reverse Counting Method for Linear Recursive Query with Many Cyclic Extensional Predicates

多変数線形再帰型演繹データベースに対する逆数え上げ評価法

Susumu Suzuki†, Toshihide Ibaraki††, Masahichi Kishi†
鈴木 晋, 茨木 俊秀, 岸 政七

Abstract *We consider to answer a datalog program that is a generalization of the well known same generation problem in the sense that it is defined over Cartesian product of m extensional predicates r_i. Each r_i is in general assumed to be cyclic. We present a method, which is the reverse counting method with a modification of termination test to deal correctly with cyclic predicates r_i, and analyze its three costs of complexity: space-requirement, database-access-time and test-time. When compared with the magic set method, which is also applicable to the same problem, this reverse counting method is inferior in the sense of worst-case bound of test-time but competitive in worst-case costs of other two. Some simulations are also conducted to examine these costs on randomly generated r_i. According to the simulation results, however, the reverse counting method is superior to the magic set method by orders of magnitude in the costs of space-requirement and database-access-time, and is in the same order in the cost of test-time.*

1 Introduction

We consider to answer the following Datalog program:

$$\begin{aligned} p(x_1,\ldots,x_m) &\ :- \ r_0(x_1,\ldots,x_m). \\ p(x_1,\ldots,x_m) &\ :- \ r_1(x_1,x_1'),\ldots,r_m(x_m,x_m'), \\ & \qquad p(x_1',\ldots,x_m'). \end{aligned}$$

which is a generalization of the well known same generation problem in the sense that the second rule contains many extensional predicates $r_1, r_2, \ldots, r_m$ defined over disjoint sets of variables $x_1, \ldots, x_m'$. These r_i are in general assumed to be cyclic. p is a recursive predicate. r_0 is an extensional predicate to initialize p. A typical query given to this program is:

$$p(a_1, a_2, \ldots, a_h, x_{h+1}, \ldots, x_m)\ ?$$

where $a_1, \ldots, a_h$ are constants. This program reduces to the same generation program when $m = 2$. An example of this program is shown in Section 2.

For the same generation program, various methods such as the counting method[6], the reverse counting method[6], the magic set method[6] and others[4] are proposed, and their worst case complexities are analyzed[1, 2, 3, 4, 5]. Here we are specially concerned with the methods which are applicable to more than two predicates $r_1, r_2, \ldots, r_m$ that are cyclic. It appears difficult to generalize the counting method to this problem setting, but the magic set method(denoted by MS method) can be generalized in a straight forward manner. For the case of $m = 2$, the reverse counting method is more efficient than the magic set method, because the former computes unary relations while the latter does binary rela-

† 愛知工業大学 情報通信工学科 (豊田市)
†† 京都大学 工学部 数理工学科 (京都市)

tions[5,6]. However, the reverse counting method is not safe when r_1 and r_2 are cyclic. In Section 3, we present a method that is the reverse counting method with a modification of termination test to deal correctly with cyclic predicates r_i. This method(Reverse Counting method with Termination test, denoted by RCwT method) can handle the case of general m.

In Section 4, we analyze the worst case bounds of RCwT method of three costs: database-access-time (to carry out join operations), test-time (to check whether given tuples already exist) and space-requirement. In Section 5, Some simulation tests are conducted to examine actual performance of RCwT method and MS method on randomly generated ri. Finally, we give our conclusion in Section 6.

1 Program Example

An example of a Datalog program in Section 1 is:

$$ex\_sg(x_1, \ldots, x_{m-1}, y)$$
$$:-\ \ m\_eq(x_1, \ldots, x_{m-1}, y).$$
$$ex\_sg(x_1, \ldots, x_{m-1}, y)$$
$$:-\ \ par(x_1, x_1'), \ldots, par(x_{m-1}, x_{m-1}'),$$
$$2\_eq(y, y'), ex\_sg(x_1', \ldots, x_{m-1}', y').$$
$$ex\_sg(a_1, \ldots, a_{m-2}, x_{m-1}, y)\ ?$$

where ex_sg is a recursive predicate and par, m_eq and 2_eq are extensional predicates. $m\_eq(x_1, \ldots, x_{m-1}, y)$ means that $x_1 =, \ldots, = x_{m-1} = y$, and $2\_eq(y, y')$ does that $y = y'$. $par(x_i, x_i')$ means that x_i' is a parent of x_i. The intention of $ex\_sg(x_1, \ldots, x_{m-1}, y)$ is that $x_1, \ldots, x_{m-1}$ are cousins having a common ancestor y, i.e., there are lines of descendant from y to $x_1, \ldots, x_{m-1}$ covering the same number of generations. par can be cyclic. The query $ex\_sg(a_1, \ldots, a_{m-2}, x_{m-1}, y)$ requests to find all pairs of the cousin x_{m-1} and the common ancestor y of particular individuals $a_1, \ldots, a_{m-2}$.

2 Reverse Counting Method with Termination Test

We present RCwT method in Fig.1. According to the idea of the magic set[6], each r_i is restricted to a part relevant to a constant a_i before RCwT method starts. The reverse counting set $RCS_j(i_1)$ is a set of descendants x_j" that are i_1 generations down from x_{1j} , where $(x_{11}, x_{12}, \ldots, x_{1m}) \in R_0$. $RCS_j(i_1 + i_2)$ is a set of descendants x_j" that are i_2 generations down from x_{2j}, where $(x_{21}, x_{22}, \ldots, x_{2m}) \in R_0$. And so on. The method works as follows:
(1)in step2, computes all $RCS_j(i)$ to satisfy a condition: $\{(x_1, \ldots, x_m) \mid p(x_1, \ldots, x_m)\} \subset \cup_k\{(x_1, \ldots, x_m) \mid x_j \in RCS_j(k)\}$, (2) answers to a query by using these $RCS_j(k)$ in step3.

RCwT method has a termination test: $\{(x_1, \ldots, x_m) \mid x_j \in WS_j\} \subseteq \cup_{k=0}^{i-1}\{(x_1', \ldots, x_m') \mid x_j' \in RCS_j(k)\}$ (line 9) , so that it can terminate even when all r_j are cyclic. To reduce running-time, termination tests are also executed only when the number of generations down from each $(x_1, x_2, \ldots, x_m) \in R_0$ is power of 2(line 9).

1: step1: /* initialize */
2 : $R_0 := \{(x_1, \ldots, x_m) \mid r_0(x_1, \ldots, x_m)\}$;
3: step2: /* compute all reverse counting sets $RCS_j(i)$ */
4: i := 1;
5: while $R_0 \neq \phi$ do begin
6 : $R_0 := R_0 - \{(x_1, \ldots, x_m)\}$; /* $\exists (x_1, \ldots, x_m) \in R_0$ */
7: for j:= 1 to m do $WS_j := \{x_j\}$;
8: ij := 1;
9: while i=1 or notpower(ij, 2) or

$$\{(x_1, \ldots, x_m) \mid x_j \in WS_j\}$$
$$\not\subseteq \cup_{k=0}^{i-1}\{(x_1', \ldots, x_m') \mid x_j' \in RCS_j(k)\}$$

/* notpower(ij, 2) means that ij is not power of 2 */
10: do begin
11: for j := 1 to m do begin
12 : $RCS_j(i) := WS_j$;
13 : $WS_j := \{x_j \mid r_j(x_j, x_j'), x_j' \in RCS_j(i)\}$;
14: end
15: i := i + 1; ij := ij + 1;
16: end
17: end
18: step3: /* find all answers */
19: Answer :=

$$\cup\{(x_{h+1}, , , x_m) \mid x_{h+1} \in RCS_{h+1}(k), \ldots, x_m \in RCS_{m(}k)\};$$

k such that $a_1 \in RCS_1(k), \ldots, a_h \in RCS_h(k)$

Fig.1. RCwT method

Fig.2 shows an example of RCwT method's process for { $r_1(a_1, a_2)$, $r_1(a_2, a_1)$, $r_2(b_1, b_2)$, $r_2(b_2, b_1)$, $r_3(c_1, c_2)$, $r_3(c_2, c_1)$, $r_3(c_2, c_2)$, $r_0(a_1, b_1, c_1)$, $r_0(a_1, b_2, c_2)$ }.

Li	ij	$R1$	$R2$	$R3$	test	status
1	1	$\{a_1\}$	$\{b_1\}$	$\{c_1\}$		(NEW)
2	2	$\{a_2\}$	$\{b_2\}$	$\{c_2\}$	TEST	NEW
3	3	$\{a_1\}$	$\{b_1\}$	$\{c_1, c_2\}$		(NEW)
4	4	$\{a_2\}$	$\{b_2\}$	$\{c_1, c_2\}$	TEST	NEW
5	5	$\{a_1\}$	$\{b_1\}$	$\{c_1, c_2\}$		(OLD)
6	6	$\{a_2\}$	$\{b_2\}$	$\{c_1, c_2\}$		(OLD)
7	7	$\{a_1\}$	$\{b_1\}$	$\{c_1, c_2\}$		(OLD)
	8	$\{a_2\}$	$\{b_2\}$	$\{c_1, c_2\}$	TEST	OLD
8	1	$\{a_1\}$	$\{b_2\}$	$\{c_2\}$	TEST	NEW
9	2	$\{a_2\}$	$\{b_1\}$	$\{c_1, c_2\}$	TEST	NEW
10	3	$\{a_1\}$	$\{b_2\}$	$\{c_1, c_2\}$		(NEW)
	4	$\{a_2\}$	$\{b_1\}$	$\{c_1, c_2\}$	TEST	OLD

Li means level i.
$R1, R2$ and $R3$ mean $RCS_1(i), RCS_2(i)$ and $RCS_3(i)$, respectively.
TEST means execution of termination test.
NEW means existence of new tuples, OLD no existence.

Fig.2. An example of RCwT method

3 Worst-case Costs

We compare the methods using three costs:

(1) database-access-time

Running time to carry out join operations. It is the size of the intermediate results of join operations, i.e. the size of $\{x_j\}$ in line 13 in Fig.1 before duplication elimination.

(2) test-time

Running time other than database access. It is namely time to check whether generated tuples already exist. Considering the time to process a tuple, test-time of RCwT method is the product of the number of tuples generated at level i and i, i.e. $|\{(x_1, \ldots, x_m)\}| \times i$ in line 9 in Fig.1. Test-time of MS method is the number of tuples generated.

(3) space-requirement

The size of work space. It is the summation of $|RCS_j(i)|$.

The worst-case costs of RCwT method are shown in Tab.1. Those of MS method are also shown in Tab. 2 to compare these two methods.

database-access-time
$O(G \times \Sigma_{i=1}^{m} E_i) \leq O(\Pi_{i=1}^{m} N_i \times \Sigma_{i=1}^{m} E_i)$
test-time
$O(F \times G \times \Pi_{i=1}^{m} N_i \times \log_2 \frac{G}{F})$ $\leq O(F \times (\Pi_{i=1}^{m} N_i)^2 \times \log_2 \frac{\Pi_{i=1}^{m} N_i}{F})$
space-requirement
$O(G \times \Sigma_{i=1}^{m} N_i) \leq O(\Pi_{i=1}^{m} N_i \times \Sigma_{i=1}^{m} N_i)$

Tab.1. Worst-case costs of RCwT method

database-access-time
$O(\Pi_{i=1}^{m} N_i \times \Sigma_{i=1}^{m} \frac{E_i}{N_i})$
test-time
$O(\Pi_{i=1}^{m} E_i)$
space-requirement
$O(m \times \Pi_{i=1}^{m} N_i)$

Tab.2. Worst-case costs of MS method

where N_i and E_i are the number of nodes and edges, respectively, in the graph SG_i(node set SVi, edge set SE_i) representing r_i, and F is the number of tuples in r_0. Let $MG(MV, ME)$ be a product graph defined by
$MV = \{(x_1, \ldots, x_m) \mid x_j \in SV_j\}$ and
$ME = \{((x'_1, \ldots, x'_m), (x_1, \ldots, x_m)) \mid r_j(x_j, x'_j)\}$.
For each tuple $\bar{t_j} (= (t_{j1}, \ldots, t_{jm}))$ in r_0, let $Sub\_MG_j(MV_j, ME_j)$ be the subgraph of MG spanned by $MV_j \subseteq MV$, where $MV_1 = \{\bar{x} \in MV \mid \bar{x}(= (x_1, \ldots, x_m))$ is reachable from $\bar{t_1}\}$ and $MV_j = \{\bar{x} \in MV \mid \bar{x}$ is reachable from $\bar{t_j}\} - (MV_1 \cup \ldots \cup MV_{j-1})$. Then, G is defined by $G = I_1 + I_2 + \ldots + I_F$, where $I_j = 1 + \max_{\bar{x} \in MV_j}\{$ the length of the shortest path from $\bar{t_j}$ to $\bar{x}$ in $Sub\_MG_j(MV_j, ME_j)\}$.

PROOF) We will prove Tab.1. Let MAXI be the level, i.e. i in Fig.1 and Fig.2, when the method terminates. Then,

$$\text{database-access-time} \le MAXI \times \Sigma_{i=1}^{m} E_i)$$
$$\text{test-time} \le \log_2(\frac{MAXI}{F})^F \times MAXI \times \Pi_{i=1}^{m} N_i$$
$$\text{space-requirement} \le MAXI \times \Sigma_{i=1}^{m} N_i$$

where, $\log_2(\frac{MAXI}{F})^F$ is the upper bound of the number of executions of termination tests. For each $\bar{t_j}$ in r_0, let I'_j be the number of executions of the loop consisting of lines 10-16 in spite of the fact that all tuples $\{(x_1, \ldots, x_m) \mid x_j \in RCS_j(i)\}$ at the same level have already been generated. Then,

$$MAXI = (I_1 + I'_1) + (I_2 + I'_2) + \ldots + (I_F + I'_F)$$

Also,

$$G = I_1 + \ldots + I_F$$
$$I'_j \le I_j$$
$$G \le \Pi_{i=1}^{m} N_i$$

These prove Tab.1.

Tab.1 indicates that G greatly influences the performance of RCwT method. That is, (i) in the case of $G \ll \Pi_{i=1}^{m} N_i$, RCwT method is more efficient than MS method in database-access-time and space-requirement, (ii) while, in the case of $G \doteq \Pi_{i=1}^{m} N_i$, RCwT method is inferior to MS method in test-time.

4 Average Costs

Some simulation tests are conducted on randomly generated r_i. The parameters are that $m = 2 \sim 5, N(= N_i) = 7 \sim 100, E/N(= E_i/N_i) = 1.0 \sim 4.0, F = 20$. These simulation tests shows that $G = O(\log_{E/N} N^m)$ in the case of $E/N >= 1.5$. Tab.3 shows the costs when the above G is substituted into the G of Tab.1. It is also estimated in test-time in Tab.2 that (the number of executions of termination test $F \times \log_2 \frac{G}{F}) \doteq G$.

database-access-time
$m^2 \times \log_{E/N} N \times E$
test-time
$m^2 \times (\log_{E/N} N)^2 \times N^m$
space-requirement
$m^2 \times \log_{E/N} N \times N$

Tab.3. Estimates of average costs of RCwT method

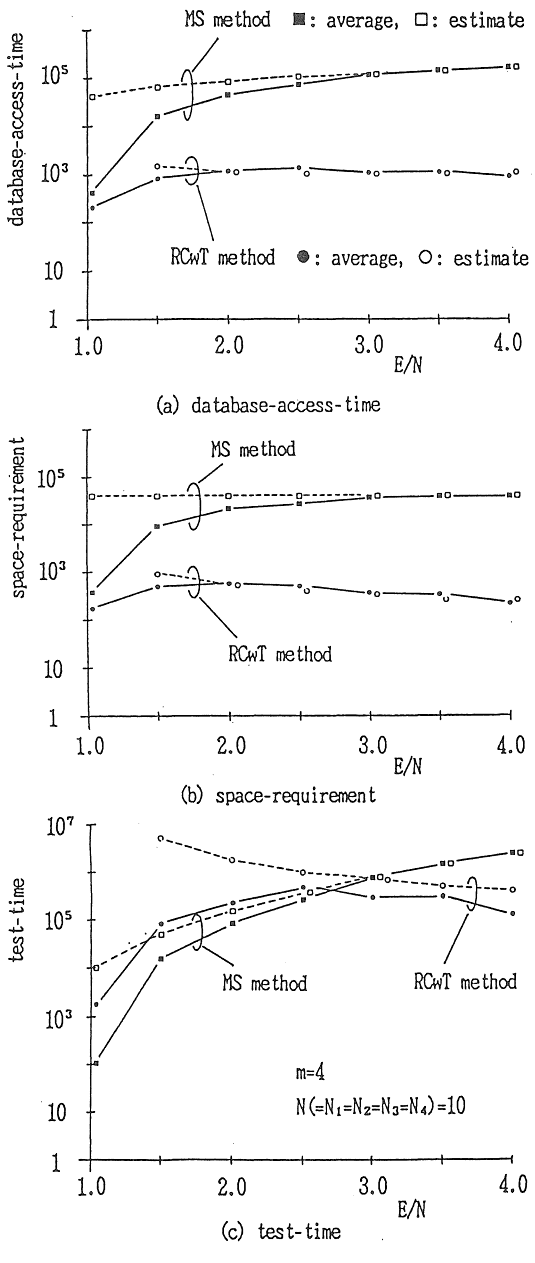

Fig. 3. Average costs on randomly generated r_j

We also demonstrate the simulation results of average costs of RCwT method and MS method in Fig.3, where $m = 4, N = 10$ and $F = 20$. The results of RCwT methods in Fig.3 are close to the estimation of Tab.3. The results of MS method in Fig.3 are also close to the worst case formulas in Tab.2.

These indicate (i)RCwT method is superior to MS method by orders of magnitude in the average costs of space-requirement and database-access-time, (ii)RCwT method is in the same order as MS method in the average cost of test-time.

5 Conclusion

We have considered to answer a linear recursive datalog program with many cyclic extensional predicates r_i. We have presented a method that is a modification of the reverse counting method, and have evaluated the performance in three costs: space-requirement, database-access-time and test-time. This reverse counting method with termination test is inferior to the magic set method in the worst case bound of test-time; however, in the average space-requirement and database-access-time on randomly generated r_i, this method is superior to the magic set method by orders of magnitude. The other costs of these two methods are almost the same.

To apply our method to real applications, it is desired to improve further the computational cost of test-time.

References

[1] A. Marchetti-Spaccamela and D. Sacca, "Worst-case Complexity Analysis of Methods for Logic Query Implementation," Proc. 6th ACM SIGACT-SIGMOD-SIGART Symp. on PODS, 1987, pp.294-301.
[2] C. Beeri, "On the Power of Magic," 6th ACM SIGACT-SIGMOD-SIGART Symp. on PODS, 1987, pp.269-283.
[3] D. Sacca and C. Zaniolo, "On the Implementation of a Simple Class of Logic Queries for Databases," Proc. 5th ACM SIGACT-SIGMOD Symp. on PODS, 1986, pp.16-23.
[4] D. Sacca and C. Zaniolo, "Magic Counting Methods," Proc. ACM SIGMOD Intern. Conf. on Management of Data, 1987, pp.49-59.
[5] F. Bancilhon and R. Ramakrishnan, "An Amateur's Introduction to Recursive Query Processing Strategies," Proc. ACM SIGMOD Intern. Conf. on Management of Data, 1986, pp.16-52.
[6] F. Bancilhon, D. Maier, Y. Sagiv and J. Ullman, "Magic Sets and Other Strange Ways to Implement Logic Programs," Proc. 5th ACM SIGACT-SIGMOD Symp. on PODS, 1986, pp.1-15.

(受理 平成 3 年 3 月 20 日)