3G−3 Join Query Optimization in Object-Oriented Databases

Li Yaxin* Hiroyuki Kitagawa† Nobuo Ohbo†
*Doctoral Program in Engineering, University of Tsukuba
†Institute of Information Science and Electronics, University of Tsukuba

1 Introduction

Object-oriented database systems (OODBSs) are very promising for supporting new data base applications. However, there remain many research issues in implementing high performance OODBSs. One important issue is query processing, in particular query optimization. In OODBSs, efficient execution of queries involving complex objects is required. In this presentation, we propose an algorithm to derive the cost optimal execution sequence for join queries involving complex objects under certain assumptions.

2 Basic Terminology

Complex objects form hierarchical structures. A table with a nest of columns and rows can be used for representing complex objects. Such a table is called a *nested relation*. We consider two types of join: *natural join* and *natural embed* [1]. Examples of natural embed and natural join, denoted by ⋈ and ϵ respectively, are shown in Figure 1.

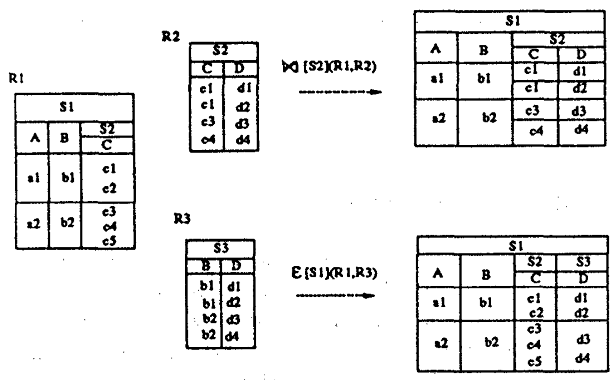

Figure 1: Join and Embed

Corresponding to a given query, we introduce a *query graph* with two edge types. One is a *join edge* representing natural join, and the other is an *embed edge* representing natural embed. A subgraph including only join edges is called a *join cluster*. An example of a query graph is shown in Figure 2.

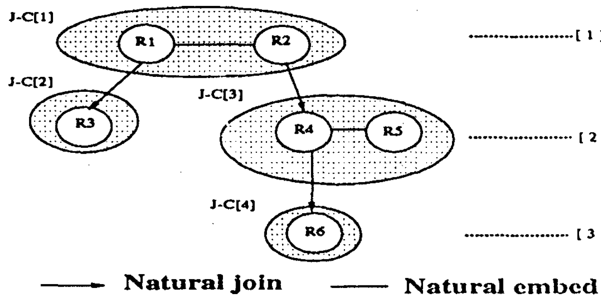

Figure 2: Query Graph

An execution of a query is represented by a *processing tree*. A processing tree is called a *binary linear processing tree* (*BLPT*) if all the joins and embeds performed are binary, and no more than one temporary relation is used as input to any join and embed.

3 Model

We propose an algorithm which derives the cost optimal BLPT for a given query graph.

3.1 Assumptions

1. The internal structure of a join cluster forms a tree.
2. The structure connecting join clusters forms a tree.
3. The bottom of a BLPT is restricted to be a relation in the root join cluster.
4. The general cost formula (i.e., n1 × g2(n2)) is applicable for all the join and embed methods.
5. In any $\bowtie[S_k](R_i, R_j)$ and $\epsilon[S_k](R_i, R_j)$, S_k is the root group of R_i.
6. Once $\epsilon[S_k](R_i, R_j)$ is performed, all the joins and embeds involved in the join clusters of R_j and its descendents are performed before any of the other join and embed operations.

3.2 Cost Equation

The number of tuples in group S_k of R_i is denoted by $\gamma(R_i, S_k)$. If S_k is the root group, $\gamma(R_i, S_k)$ is simply denoted by $\gamma(R_i)$. The join and embed costs are estimated as follows:

$$\begin{aligned} Cost(\bowtie[S_k](R_i, R_j)) &= \gamma(R_i, S_k) \times CJ_j(\gamma(R_j, S_k)) \\ Cost(\varepsilon[S_k](R_i, R_j)) &= \gamma(R_i, S_k) \times CE_j(\gamma(R_j, S_k)). \end{aligned}$$

Here, CJ_j is the cost of the join per tuple in R_i, and CE_j is the cost of the embed per tuple in R_i.

The number of tuples in group S_m of the result relation of a join operation $\bowtie[S_k](R_i, R_j)$, denoted by $\gamma(\bowtie[S_k](R_i, R_j), S_m)$, is calculated as follows:

Case 1: S_m is S_k or a child of S_k

$$\gamma(\bowtie[S_k](R_i, R_j), S_m) = SJ_{ij} \times \gamma(R_i, S_m) \times \gamma(R_j)$$

where SJ_{ij} is the join selectivity.

Case 2: Otherwise

$$\gamma(\bowtie[S_k](R_i, R_j), S_m) = \gamma(R_i, S_m)$$

Similar formulas can be derived for the result of embed operation.

4 Query Optimization

The algorithm *Opt* will find the optimal BLPT for a query graph G under the assumptions of Section 3.1.

Algorithm *Opt*

1. Call *Supopt*.
2. Select a relation R as the bottom of a target BLPT in the root cluster C_0.
3. Call $KBZ(C_0, R)$, and get a BLPT.
4. If there is any other relation not yet chosen as the bottom in C_0, goto 2.
5. Select the optimal BLPT whose cost is the lowest.
6. End.

Algorithm *Subopt*

1. Let the height of the tree be H. Set level variable $i = H$.
2. If $i \equiv 0$, then goto 10.
3. Let $cn(i)$ be the number of clusters at level i, and set cluster variable $j = cn(i)$.
4. If $j \equiv 0$, then goto 9.
5. Choose the root relation R_{ij} of the j-th join cluster C_{ij} at level i.
6. Call $KBZ(C_{ij}, R_{ij})$.
7. Replace the join cluster C_{ij} by a relation $Rep(C_{ij})$ in G, and replace the corresponding embed $\varepsilon[S_k]$ (R_i, R_{ij}) by the join $\bowtie[S_k](R_i, Rep(C_{ij}))$. Here, $Rep(C_{ij})$ is the relation obtained by executing all joins and embeds involved in the join cluster C_{ij} and its descendants. CJ and SJ for join $\bowtie[S_k](R_i, Rep(C_{ij}))$ are given as follows.

   $$\begin{aligned} SJ &= 1/\gamma(Rep(C_{ij})) \\ CJ &= CE_{ij}(\gamma(R_{ij})) + SE_{ij} \times Cost(C'_{ij}) \end{aligned}$$

   Here, $Cost(C'_{ij})$ is the total cost to obtain Rep(C_{ij}).
8. $j = j - 1$, goto 4.
9. $i = i - 1$, goto 2.
10. End.

Subprocedure $KBZ(C, R)$ gives the optimal BLPT whose bottom is R for a query consisting of joins involved in C [2].

5 Example

Let us consider a query graph shown in Figure 2 as an example. The cost parameters for this example are listed in Figure 3. There are twelve candidate BLPTs for this query. The above algorithm *Opt* finds that the following sequence of joins and embeds (denoted by T12 in Figure 3) as the optimal BLPT.

$$T12 = R_2 \ \varepsilon \ R_4 \ \varepsilon \ R_6 \bowtie R_5 \bowtie R_1 \ \varepsilon \ R_3.$$

R	N	CJ	CE	SJ	SE
1	100	10		0.4	
2	200	2		0.6	
3	100		10		0.2
4	500		3		0.4
5	1000	2			0.3
6	100		10		0.5

PT	Cost	PT	Cost
T1	7,249,562,000	T7	19,306,000
T2	28,837,200	T8	120,162,600
T3	7,249,562,000	T9	4,803,306,000
T4	28,956,200	T10	4,803,306,000
T5	28,956,200	T11	48,106,000
T6	7,264,837,200	T12	582,600

Figure 3: Example

6 Conclusion

We have proposed the algorithm *Opt* which obtains the optimal BLPT for a join query involving complex objects. We have set a number of assumptions to restrict the search space for the optimal query execution plan. This may affect the applicability of the algorithm. We will improve the algorithm to find the optimal PT in more general search space.

References

[1] H. Kitagawa, T. L. Kunii: *The Unnormalized Relational Data Model: For Office Form Processor Design*. Springer-Verlag, 1989

[2] R. Krishnamurthy, H. Boral and C. Zaniolo: "Optimization of Nonrecursive Queries," *Proc. 12th Int. Conf. VLDB, Kyoto*, pp. 128–137, Aug. 1986.