Size Estimation for Query Results Using Histograms

Shant Karakashian* José Luis Ambite**
Berthe Y. Choueiry*

*Constraint Systems Laboratory
University of Nebraska-Lincoln

**Information Sciences Institute
University of Southern California

Email: shantk@cse.unl.edu

Working Note 2-2008

December 11, 2008

Contents

1 Introduction

The execution cost of query plans highly depends on the order in which the joins are executed. We propose a greedy approach for finding a 'good' join ordering. This approach selects in the sequence of joins the next join operation that will produce the smallest relation (relation with the fewest tuples).
Given n relations, at every selection, the greedy algorithm has to consider $O(n)$ join possibilities by estimating or computing the size of the resulting joined relation, and choose the smallest one. The whole process consists of a sequence of $(n-1)$ joins, resulting in $O(n^2)$ estimations. Given the quadratic complexity, the

estimation operation needs to be carried out as efficiently as possible. For this purpose, we propose to estimate the size of the joined relation using histograms that summarize the content of the relations.

2 The Histogram Structure

Each column in a relation corresponds to a variable. For each relation, we build two structures that characteriza the content of the relation:

- *A histogram*, a 2-dimensional array, stores the frequency of each value for every column in the relation. The use of the historgram is discussed in Section 3.
- *A metadata*, which stores the minimum value, the maximum value, and the count of distinct values for each column in the relation. The metadata is used in Section 4.

Table 1 shows an example of a relation A, Table 2 its metadata, and Table 3 its histogram.

Table 1: Relation A.

Relation A		
a	b	c
2	3	0
2	3	2
2	1	3

Table 2: Metadata of A.

Metadata A			
	a	b	c
minimum	2	1	0
maximum	2	3	3
distinct	1	2	3

Table 3: Histogram of A.

	a	b	c
0	0	0	1
1	0	1	0
2	3	0	1
3	0	2	1

The relation A has three columns 'a,' 'b,' and 'c,' and three tuples. The metadata shows the minimum, maximum values of each column, and the number of distinct values in each. For example, the minimum and the maximum values for column 'a' are 2, and has only one distinct value. The histogram gives the frequency of each value in each column. The first column of the 2-d array table shows the domain of the values, and the next three columns show the frequency of each value in each of columns 'a', 'b' and 'c'.

3 Estimating Joined Relation Size

In general, the size of a relation grows exponentially in the number of columns. A relation with c columns and domain size of d (assuming all the columns have the same domain) can be as large as d^c. The size of the histogram is dominated by the size of the 2-d array, which is $d \times c$. Therefore, analysing the histogram to estimate the size of the joined relation can be much cheaper than analysing

the relation itself.

Given two relations X and Y with exactly one common variable v. $\mathrm{H}_X(i, v)$ is the frequency of value i for variable v in X. The size of $XY = X \bowtie Y$ can be *exactly* computed by the following expression using only the histogram:

$$\textsc{Size}(XY) = \sum_{i \in \textsc{Domain}(v)} \mathrm{H}_X(i, v) \times \mathrm{H}_Y(i, v) \tag{1}$$

When the number of common attributes between two relations is more than one, the exact size of the joined relation can not be exactly computed using the histogram. Instead, we *estimate* the size of the joined relation using the following expression:

$$\textsc{SizeEstmt}(XY) = \Big[\textsc{Size}(X) \times \textsc{Size}(Y)\Big] \times \prod_{v \in V} \frac{\sum_{i \in \textsc{Domain}(v)} H_X(i, v) \times H_Y(i, v)}{\textsc{Size}(X) \times \textsc{Size}(Y)} \tag{2}$$

where V is the set of common variables.

Table 4: Relation B.

Relation B		
b	c	d
3	3	1
1	3	2
1	3	3

Table 5: Metadata of B.

Metadata B			
	b	c	d
minimum	1	3	1
maximum	3	3	3
distinct	2	1	3

Table 6: Histogram of B.

Histogram B			
	b	c	d
0	0	0	0
1	2	0	1
2	0	0	1
3	1	3	1

Consider the relations A and B in Tables 1 and 4. The size estimate for $AB = A \bowtie B$ is computed as:

$$\begin{aligned}
\textsc{SizeEstmt}(AB) = \quad & [3 \times 3] \times \left\{ \tfrac{(0\times0)+(1\times2)+(0\times0)+(2\times1)}{3\times3} \right\} \times \left\{ \tfrac{(1\times0)+(0\times0)+(1\times0)+(1\times3)}{3\times3} \right\} \\
= \quad & 9 \times \tfrac{4}{9} \times \tfrac{3}{9} \\
= \quad & \tfrac{4}{3} = 1.33.
\end{aligned}$$

The actual size of AB is 2, see Table 7.
To understand the intuition behind this estimation, observe that every variable of a tuple that is common to both relations should be matched so that the tuple survives in the joined relation. We are able to exactly calculate the number of tuples surviving when the relations overlap on only one variable. For example, if the two joined relations have 5 tuples each, joining them yields a relation with at most 25 tuples, the case when all the tuples survive the join condition. If we

Table 7: Relation AB obtained by joining relations A and B.

Relation AB			
a	b	c	d
2	1	3	2
2	1	3	3

determine that the join has only 10 tuples, then the probability of survival of a tuple is 10/25.

In the case where more than one variable in the two relations need to be matched, we are still able to exactly compute the number of surviving tuples for each common variable. However, the number of surviving tuples that agree on all common variables now needs to be estimated. For this purpose we consider each common variable to be a random variable. The tuple survival event, which is based on the survival caused by each common variable independently, can be computed by taking the product of the independent probabilities. We are making here the assumption that these random variables are independent, i.e. every common variable can be matched independently from the other common variables. This assumption does not necessarily hold, specially for well-structured data, and we hope that the degradation of the estimation quality due to the structured data is not significant and does not invalidate this approach. (Experimental evaluation to follow.) In Expression (2), the right-hand side product term computes the probability of the survival of a tupple, and multiplies this probability with the number of all possible tupples in the joined relation to yield the size estimate of the joined relation.

4 Estimating Joined-Relation Size After Projection

During the execution of the join sequence, the columns corresponding to variables that do not appear in future joins may be dropped by projecting the relation on the remaining columns. After projection, the size of the relation decreases. It could be the case that the pair of relations chosen by the greedy algorithm does not yield the smallest size join after projection, even if it has the smallest size before projection. For this reason, incorporating the projection-size estimation into the join-size estimation may enhance the quality of the estimation of the size of the resulting relation and, thus, the performance of the greedy approach.
We can estimate the upper-bound size of the projected-relation by considering all the combinations of the distinct values in each column of the projected relation. The expression for this estimation for a relation X projected on the

columns P is:

$$\text{P-SizeEstmt}(X, P) = \text{Min}\left(size(X), \prod_{v \in P} \text{Distinct}(v, X)\right) \tag{3}$$

where $\text{Distinct}(v, X)$ is the number of distinct values in the relation X for column v obtained from the metadata of X.
Consider now two relations X and Y with variables V_X and V_Y, respectively. Let $P \subset (V_X \cup V_Y)$ be the set of variables on which we project $X \bowtie Y = XY$. We estimate the size of the relation resulting from projecting the join of X and Y on P using the following expression:

$$\text{P-SizeEstmt}(XY, P) = \text{Min}\left(\text{SizeEstmt}(XY), \prod_{v \in P} \text{Distinct}'(v, \{X, Y\})\right) \tag{4}$$

$\text{SizeEstmt}(XY)$ is computed using Equation (2). For a variable v in P that is exclusively in relation X (respectively, Y), $\text{Distinct}'(v, \{X, Y\})$ is equal to $\text{Distinct}(v, X)$ (respectively, $\text{Distinct}(v, Y)$). For a variable $v \in P$ that is common to X and Y, the number of distinct values is estimated by counting those values that appear in both X and Y. Algorithm 1 gives the details of computing $\text{Distinct}'(v, \{X, Y\})$, where:

- $\text{Domain}(i)$: domain of the variable i.
- $\text{H}_X(i, j)$: frequency of value i in column j of relation X.
- $\text{Distinct}(a, X)$: count of distinct values in column a in relation$X$.

Consider the relations A and B in Tables 1 and 4. The size estimate for $\pi_{P=\{a,c\}}AB$ where $AB = A \bowtie B$ is computed as:

$$\begin{aligned} \text{P-SizeEstmt}(AB, P) &= min\left(1.33,\ \{1 \times 1\}\right) \\ &= 1 \end{aligned} \tag{5}$$

The actual size of $\pi_{P=\{a,c\}}AB$ is 1, see Table 8.

Table 8: Relation $\pi_{\{a,c\}}AB$ obtained by projecting $A \bowtie B$ on $\{a, c\}$.

Relation $\pi_{\{a,c\}}AB$	
a	c
2	3

Algorithm 1 DISTINCT$'(v, \{X, Y\})$

inputs:
v
Relation X with variables V_X
Relation Y with variables V_Y
output: estimated count of tupples
count: counter holding the estimated count

```
1: count ← 0
2: if v ∈ V_X \ V_Y then
3:    count ← DISTINCT(v, X)
4: else if v ∈ V_Y \ V_X then
5:    count ← DISTINCT(v, Y)
6: else
7:    for all i ∈ DOMAIN(v) do
8:       if H_X(i, v) × H_Y(i, v) ≠ 0 then
9:          count ← count + 1
10:      end if
11:   end for
12: end if
13: return count
```