Improved Size Estimation using Histogram Values

Deepali Adlakha, Divyam Bansal, J Guna Prasaad, Vipul Venkataraman

CS613 Course Project - Autumn 2014

1 Introduction

In this project, we aim to improve the size estimation of output in the PyroJ query optimizer. Accurate size estimation helps in building better cost models to achieve resourceful plans for execution.

2 Statistics Used

We improve the existing size estimation using histogram values. More specifically, we have used **equi-depth histogram** values. The following statistics about the relational attributes were used in the estimation model:

Symbol	Description
S	size of relation (number of tuples)
D	number of distinct values
m	minimum value of the attribute
M	maximum value of the attribute
C_i	list of common values
$f(C_i)$	frequency of common value
H_i	histogram bounds (excluding min and max) which divide the attribute values into approximately equal-sized bins

3 Size Estimation Model

In the size estimation for different relational operations, the code base is organized based on the following base-level operations on the attribute properties:

- **Equate Attributes**
  Given two attributes $A1$ and $A2$, find the statistics for predicate $A1 = A2$
- **Fix Attribute Value**
  Given an attribute $A1$ and a constant value v, find the statistics for predicate $A1 = v$
- **Upper Bound Attribute**
  Given an attribute $A1$ and a constant value v, find the statistics for predicate $A1 \geq v$
- **Lower Bound Attribute**
  Given an attribute $A1$ and a constant value v, find the statistics for predicate $A1 \leq v$

We explain how we deal with each of these cases in the following subsections.

3.1 Equate Attributes

The values for input attributes $A1$ and $A2$ are denoted by indices 1 and 2 respectively. The values for output relation is denoted by subscript o.

Min, Max and Distinct Values These values are calculated directly based on the corresponding values of the input attributes.

$$m_o = max(m_1, m_2) \tag{1}$$

$$M_o = min(M_1, M_2) \tag{2}$$

$$D_o = min(D_1, D_2) \tag{3}$$

Common Values and Frequencies Common values for the output relation are calculated by comparing the values and adding the corresponding frequencies.

$$C_o = \{v \mid v \in C_1 \cap C_2\} \tag{4}$$

$$f_o(v) = f_1(v) + f_2(v),\ \forall v \in C_o \tag{5}$$

Histogram Values Histogram values are calculated based on the assumption that *the probability of each value lying in a sub-interval is proportional to the ratio of size of sub-interval to that of the interval to which the sub-interval belongs.*

To simplify the explanation let us consider the equi-depth histogram values of $A1$ be $x_1, x_2, ..., x_m$ and $A2$ be $y_1, y_2, ..., y_n$. Further, let us assume that the number of tuples for each of the intervals for $A1$ be p and for $A2$ be q.

Let the intersect set of interval points be $a_1, a_2, ..., a_k$, such that $a_i < a_{i+1}$ and $a_i = x_k$, for some k or $a_i = y_k$, for some k. We can estimate the number of tuples in each of this interval based on the two histograms.

$$f_x(a_k, a_{k+1}) = \frac{a_{k+1} - a_k}{x_{i+1} - x_i} * p, \quad (a_k, a_{k+1}) \in (x_i, x_{i+1}) \tag{6}$$

$$f_y(a_k, a_{k+1}) = \frac{a_{k+1} - a_k}{y_{i+1} - y_i} * q, \quad (a_k, a_{k+1}) \in (y_i, y_{i+1}) \tag{7}$$

$$f(a_k, a_{k+1}) = min(f_x(a_k, a_{k+1}), f_y(a_k, a_{k+1})) * 0.5 \tag{8}$$

Once the estimated number of tuples in each of these intervals is computed using the above formula, we can further divide the complete range into equi-depth intervals based on our base assumption regarding distribution of tuples in a given range.

Size of relation The size of relation is calculated from the equi-depth histogram created above and frequencies of common values.

3.2 Fix Attribute Value

Given an attribute $A1$ and a constant value v, we want to find the statistics for predicate $A1 = v$. The values of output relation is denoted by the subscript o.

$$m_o = v$$
$$M_o = v$$
$$D_o = 1$$

To estimate the size of the relation, we first check if the value v is a common value, else based on our assumption, we return the size to be the following:

$$S_o = S_1 * \frac{(H_{i+1} - H_i)}{M_1 - m_1}, \qquad v \in (H_i, H_{i+1}) \tag{9}$$

Please note that the set H needs to be appropriately extended to include minimum and maximum values. This is needed owing to convention followed in the data statistics obtained. Moreover, in the result, v is a most common value and hence it is added to that array.

3.3 Upper/Lower Bound Attribute

Given an attribute $A1$ and a constant value v, we want to find the statistics for predicate $A1 \geq v$ or $A1 \leq v$. The minimum value and maximum value are updated accordingly. The number of distinct values are estimated based on the assumption of equal distribution of values in a range.

$$D_o = D_1 * \frac{M_o - m_o}{M_1 - m_1} \tag{10}$$

The common values array is updated according the bound set by eliminating elements that are out of the range. The histogram values are re-calculated in such a way that the number of tuples in each of the remaining intervals are same.

4 Implementation Details

The class `AttributeProperty` in the package `pyroj.sqlruntime.properties.logical` records the attribute level statistics. In the current implementation class `IntegerProperty`, which inherits AttributeProperty, is used. `IntegerProperty` records only the number of distinct values and min, max values. Hence, the estimation of size is based on and limited to only these statistics.

- We created a new class `AdvancedProperty` which extends `AttributeProperty`. In this class, we added appropriate members to maintain the above mentioned statistics.
- We modified the existing codebase to create objects of class `AdvancedProperty`, wherever `IntegerProperty` was used earlier.
- We modified parts of code where these statistics were used in estimating the *selectivity* of a particular operation and hence the size of the resultant relation.

5 Future Work

- Add datatype from pg_attributes
- Change the code architecture to handle normal and join predicates suitably
- Look into PostgreSQL code to find out how it handles `equateAttributes`
- Estimate number of distinct values in each interval to improvise relation size estimation