Pragmatic Estimation of Join Sizes and Attribute Correlations

Bell DA, Ling DHO, McClean S

Institute of Informatics,
University of Ulster,
Northern Ireland, UK, BT37 0QB

ABSTRACT

A new method is presented of modelling attribute value distributions in database relations for the purpose of obtaining accurate estimates of intermediate relation sizes during query evaluation. The basic idea is that instead of keeping a single (average) value to represent the number of occurrences of each attribute value, we keep m (typically 10) parameters, each representing the number of occurrences of attribute values in a piece, or partition, corresponding to a sub-range of 1/m th of the original value range. The "uniformity assumption", taken here as an estimation technique rather than as an assumption, holds for each partition - hence the name "piecewise uniform".

The piecewise uniform method is independent of the level of the query tree at which the estimation is done and gives better estimates than can be obtained by the uniformity assumption. We establish some of the conditions under which this improvement occurs by analysis and back this up by experimental results.

The distribution method is extended to the modelling of important intra-relational attribute correlations. This and other enhancements to the technique such as for application to semi-join operation are suggested.

The technique is being used on two multi-database management systems.

1. Introduction

Query optimisers in relational database management systems, RDBMS, support ad hoc access to data via flexible and powerful non-procedural query languages. Users of these languages state what data they require rather than the method and manner of its extraction from the database. The query execution module of a RDBMS must therefore specify strategies for accessing the requested data and in particular its query optimiser should ensure that these are as efficient as possible. Much has been written on the subject of query optimisation in distributed database systems (assumed here to be accessed via relational views) in the last 10-15 years.

Prominent issues addressed are whether static or dynamic methods are employed, the cost formulae for the calculation, whether time or cost is optimised, whether exhaustive search or heuristic search is used, and, in the case of static evaluation, how estimates are made of the size of sub-results at various levels of the query tree (or query graph).

The central subject of the present study is to produce a more satisfactory method of estimating intermediate join sizes during relational query execution than the usual one of assuming uniform distribution of the attribute values over a range and assuming independence amongst the attribute values within a single relation. The importance of good estimation techniques is universally acknowledged - get these wrong and usually the optimisation procedure is a waste of effort. The uniformity assumption is known to give poor results for practical RDBMS applications.

Our model supports non-uniformity and intra-relation attribute value correlations. The objectives are to provide an elegant and simple, universal method of characterising value distributions which can be treated in a uniform manner for all real distribution-types and at each level of a query tree, under the constraint that the accuracy of the estimation is not compromised intolerably. The reason for the traditional ubiquity of the uniformity and independence assumptions is their simplifying effects, but we believe, as does for example Christodoulakis [6], that a little of this simplification and corresponding ease of manipulation could be traded for enhanced fidelity to the real distributions being modelled.

One possibility is that we could use general or common distributions as models of the attribute value variances and characterise each of these as appropriate by for example a Beta, Gamma, Binomial or Poisson distribution. A few parameters in each case would "customise" the chosen distribution to the particular case under consideration. Problems arise when we try to characterise the...
distribution of a set of values obtained as a join over, say, 2 binomially distributed join attributes, or one binomially and one exponentially distributed attributes. The new distribution is needed for joins "higher up" the query graph which use this temporary join relation, and our method should ideally be level-independent. Moreover the accuracy with which a real distribution of attribute values can be represented using, say, 4 parameters and a general equation is clearly open to question.

The method we introduce here is simple, level-independent and general. Its elegance must be judged by the reader, but its simplicity makes it easy to transport. The purpose of this paper is to present the method and show the motivation behind it, and to provide insights into its superiority over uniformity-based approaches using both analysis and experiment. A further important feature to assess is its suitability for handling intra-relational attribute correlations.

The next section provides the background and basic concepts of the piecewise uniform (pu) method for estimation of the sizes of (sub-) query results. It indicates the place of such a method amongst the services provided by a (distributed) database system and presents algorithms to support its application. An analytical evaluation is of the technique is presented in section 3, followed by an experimental comparison using simulation, of the performance of pu with that of uniformity-based estimation (u) based on a benchmark used by researchers in previous related studies. This is followed by a section which discusses proposed enhancements of the basic method of handling attribute correlations and gives analytical and experimental support for these proposals. Section 6 presents the method of application of these techniques to semi-joins and indicates how we intend to apply it.

2. Overview of the pu method

In this paper we are concerned only with static query optimisation techniques; i.e. the intermediate relation sizes of a given query must be estimated prior to actual transmission of data. We use query cost as the metric to be optimised, rather than time, because we are concerned with multi-databases [1, 2]. However the estimation technique can also be used for response time optimisation. In order to estimate an intermediate relation's size with reasonable accuracy, some statistical information is required. Such information about relations and their attribute value distributions is stored in statistical files in the database, possibly in the global data dictionary. The amount of statistical information available within the database system has a potentially great effect on the accuracy of the estimates. However, collecting extensive statistical data and keeping it up to date incurs a large overhead for the system. Furthermore, it is impractical to keep and maintain full statistical data for each of the underlying base relations in a distributed environment. A pragmatic approach is therefore sought where the system stores a modest quantity of statistical data and yet provides fairly good estimates. The terminology is deliberately left fuzzy at this stage.

A number of approaches have previously been proposed in the literature for estimating relation sizes. Some commonly known approaches are:

(i) uniform assumption method [11]
(ii) worst case assumption method [8, 12]
(iii) worst case divided by 2 method [5, 8]
(iv) worst case divided by 10 method [8, 11]
(v) perfect knowledge method [7].

The uniform assumption method (u) assumes that each possible value within the domain of an attribute occurs an equal number of times in a relation. Such uniformity is rarely encountered in the real world. For example, names often follow a Zipf distribution [17] and are very imperfectly approximated by a uniform distribution. The same is true for attributes following other prevalent distributions. So this method is unlikely to be widely applicable [7].

The worst case assumption method is far too pessimistic [8]. Here the estimation of the size of the result of a join operation is taken to be the product of the two relation cardinalities. The worst case DIVIDED BY 2 and worst case DIVIDED BY 10 methods are apparently arbitrary developments and lack scientific justification, although they could provide good estimates in particular cases [5, 11].

The perfect knowledge method (pk) allows the calculation of the exact size of an intermediate relation. Exact information about the occurrences of attribute values is stored. As stated earlier, this method is impractical due to the large amount of data to be collected, the large computer storage needed, and the high cost of processing and maintenance such as updating and integrity checking.

In spite of the research effort spent on the issue of finding an accurate and yet operationally feasible method of estimation of intermediate relation sizes none has been established. This paper presents a new practical method, called the piecewise uniform method (pu), of calculating intermediate relation sizes. This technique lies, in a sense, between the uniform assumption and perfect knowledge methods. It improves upon the perfect knowledge method because it requires less storage space for statistical data. Intuitively it gives greater ac-
curacy than the uniform assumption method because the distribution of values of an attribute is represented by a number of parameters (one for each "piece") rather than by a single parameter for the whole relation. It uses a simple calculation technique, and so is only marginally more complex than the uniform assumption method.

The pu estimation method provides a universal way of estimating the cardinality of intermediate relations. The method remains the same regardless of the level of the query tree or graph at which a given relation is generated. Furthermore, the same estimation algorithms can be applied to any type of attribute value distribution. As we will see later it can be easily and usefully extended to handle intra-relation attribute correlations. The pu distribution can approximate for example Binomial, Gamma, Poisson, Normal, or Uniform distributions using a single representation. Attributes with these distributions are often used as join attributes, for example, and the complex new attribute distribution which results can also be represented by the pu distribution.

The pu technique assumes the availability of a minimal set of statistical data for each important attribute in each base relation. The word "important" will be explained later. The statistics include the following:

(i) size (in bytes) of each relational extension;
(ii) an ordered integer value range of each attribute domain. This range is therefore 1, 2, 3, ..., r where r is a whole number (for character string attributes a mapping function is used to map character strings into a corresponding whole number);
(iii) each attribute distribution is represented by m equi-subintervals, or pieces, of the ordered value range. The value within each subinterval comprises the number of tuples in a relation for which the attribute value belongs to the subinterval. The attribute distribution within each equi-subinterval is characterised as uniform. This means that only m values are permanently stored in the data dictionary for representing each "important" attribute value distribution.

These statistics are obtained by a one-pass procedure carried out at load time – in the case of EDDS [1] and Multi-Star [2] the SQL construct CREATE-STATISTICS is used for this purpose.

The method of obtaining estimates for set-based operations and restriction operations is given in detail elsewhere [9]. We concentrate on the join operation here.

To simplify the explanation, we shall now introduce two terms to explain the concept of the importance of an attribute: the joining attribute and the active attribute. Assume that a relation R includes two attributes, A and A', and another relation R' includes one attribute A. If at a given query tree level a relational join is required between relations R and R' over attribute A, and attribute A' of R' is to be used for joining with another attribute at a higher level of the query tree, then at this present query tree level, attribute A is referred to as the join attribute and attribute A' is called the active attribute. When the join operation is carried out, a new distribution of values from the A domain is obtained, the distribution of the active attribute A' will clearly also vary. An enhancement to the basic pu method is aimed at reflecting this dependency during the estimation procedure. The method of estimation of the distributions of the active attributes has been ignored in most of the literature. If they are mentioned at all, then the active attribute distributions are considered as independent of other attribute distributions (such as the join attributes) within a relation – this is referred to as attribute independence assumption [6].

The enhanced pu technique removes the attribute independence assumption by taking into account the attribute value dependencies. To this end, each active attribute distribution is still characterised by an m parameter vector except that within each of their equi-subintervals there are m additional values to represent the value distribution of some dependent attributes. This is dealt with in Section 5.

Estimation of the relation size is made by simply totalling the m parameters which characterise the attribute value distributions. The estimation of intermediate relation size for the relational operators such as project, select, union, and cartesian product are relatively trivial because they involve straightforward arithmetic operations. However, estimation of the results of join operation involves more computation steps. Firstly, the joining attribute distribution ranges have to be harmonised to obtain a new estimated distribution range – intersection of the two distinct ranges, because the join uses only values common to both of the join attributes. Harmonisation of the active attributes is also required. The distribution of each of the join attributes over the harmonised range is calculated by computing the number of tuples having values in the m equi-subintervals, or pieces, of the value range. These distributions are used to give the estimated intermediate relation size. The new distribution of the active attribute, A' here, in the intermediate relation, is obtained by a scaling method. In this paper we deal only with joins for illustration of the technique.

We assume that the A attributes in R (k=1,2) take the
values $V_i$. We also assume initially that the $V_i$ values in $R_x$, $R_y$ respectively range over the same set of positive integers. Thus the ranges are in harmony (size $s$).

**Algorithm: Join-Size-Estimation**

**Begin**

**Step 1** Partition the range $r$ of $A$ into $m$ (say 10) equal pieces. Similarly for $A'$. Calculate the number of $A$ values ($C_i$, represented as $C_i[i]$) in each piece for both relations, where $i=1$ to $m$ and $k=1,2$.

**Step 2** Compute the distribution of $A$ after joining using procedure JOIN-D.

**Step 3** Compute the distribution of $A'$ after joining using procedure SCALE-D.

**End.**

**Procedure: Join-D**

**Begin**

(*) compute an $m$-element array (newA) for the new distribution *)

$s := r/m$;
FOR $i := 1$ TO $m$ DO newA[i] := 0;
FOR $i := 1$ TO $m$ DO begin
    mult := $C_i[i] * C_i[i]$;
    newA[i] := mult/s;
    end;
End; (* Join-D *)

**Procedure: Scale-D**

**Begin**

(*) calculate the number of $A'$ values (denoted as $d[i]$) in each of the $m$ pieces *)

$n := 0$; $n' := 0$;
FOR $i := 1$ TO $m$ DO begin
    $n := n + A'[i]$;
    $n' := n' + newA[i]$;
    end;

(*) initialise an $m$-element array (newAA) for the new $A'$ distribution *)

FOR $i := 1$ TO $m$ DO newAA[i] := 0;
FOR $i := 1$ TO $m$ DO newAA[i] := $d[i] * n'/n$;
End; (* Scale-D *)

As was stated above this algorithm works for situations where the ranges of $A$ in $R_x$ and $R_y$ are the same. When this is not the case, the ranges have to be harmonised, so that for a particular value of $i$ the corresponding partitions in $R_x$ and $R_y$ have the same end-points. All of our analysis and experimentation work under the assumption that the range of $A'$ is the same. The harmonisation process required to ensure that this is so is carried out by a simple procedure. The basic idea is that only the overlapping portions of $A'$'s range in $R_x$ and $R_y$ are considered. This range which could be less than or equal to the ranges in $R_x$ and $R_y$ respectively is then the range which is partitioned into $m$ pieces in Step 1 of Join-Size-Estimation above. The partitioning over the overlap must be derived from the original piecewise distribution generated at input by the system software, and so this represents a complication to the basic algorithms presented above. However it turns out that such harmonisation is exceedingly simple if we use a kind of sort-merge technique. The algorithm is presented elsewhere [3].

We justify the formulae underpinning Algorithm Join-Size-Estimation in Section 3. The validity of use of the scaling procedure is open to question. In Section 5 we also consider conditions under which it is valid to make use of it and later extend the method to deal with the other situations that arise – that is where the $A'$ distribution depends on the $A$ distribution. This is our approach to handling correlations between attribute values.

3. Comparing the Uniform and Piecewise Uniform Estimates for Size of a Join

In this section we determine analytically conditions under which the piecewise uniform estimation method gives superior results to those for estimation using a uniformity assumption. We argue that it supports our hypothesis that the $pu$ estimates are superior to $u$ estimates under certain circumstances.

Notation: Assume there is a range of $r$ values for attribute $A$ in relation $R_x$ any $k$ and each relation is partitioned into $m$ (=10) equal divisions $D_i$ so that there are $s$ (=r/m) values of $A$ in partition $D_i$ for $i=1, ..., m$.

Let $v_j^k$ be the actual number of tuples of value $V_j$ of $A$ in relation $R_x$ for $j = 1, ..., r$. $v_j^k$ is the corresponding variable which denotes the multiplicity of the value for the relation $R_x$.

$c_i^k$ is the number of tuples in partition $D_i$ (for $i=1, ..., m$) of relation $R_x$ (for $k=1,2$). $C_i$ is the corresponding variable ranging over the $D_i$'s for relation $R_x$.

If $n_k$ is the number of tuples in relation $R_x$ then

\[ c_i^k = \sum_{V_j} v_j^k \]

\[ c_i^k = \text{sum of number of tuples over values } V_i \text{ in } D_i \text{ of } R_x \]

and

\[ n_k = \sum_{i=1}^{m} c_i^k = \sum_{j=1}^{r} v_j^k \]

= total tuples in $R_x$.

The size of the join of $R_x$ and $R_y$ on $A$ is $N$. It is estimated by the value of $N_{pk}$ where sub takes the values $pk$, $u$ and $pu$ depending on whether the estimation
has perfect knowledge (pk) of the \( V_i \), or makes use of the uniform distribution or piecewise uniform distribution in the calculations. The formulas for these three join size estimates are given below.

**Perfect Knowledge**

\[
N_{pk} = \sum_{j=1}^{s} v'_1 v'_2
\]

**Uniform**

\[
N_u = \frac{s}{r} (n_1 / r) (n_2 / r) = \frac{n_1 n_2}{r}
\]

= \frac{1}{r} \left( \frac{\sum_{j=1}^{s} c'_1 c'_2}{r} \right)

**Piecewise Uniform**

\[
N_{pu} = \sum_{j=1}^{s} \frac{c'_1 c'_2}{s}
\]

**Theorem:**

If \( V_1 \) and \( V_2 \) are positively (negatively) correlated for all values \( V \) of attribute \( A \), in each partition, and \( C_1 \) and \( C_2 \) are positively (negatively) correlated then \( N_{pu} \) is a better estimate for \( N_{pk} \) than \( N_u \).

**Proof:**

(i) Compare \( pk \) and \( u \)

\[
N_{pk} - N_u = \sum_{j=1}^{s} v'_1 v'_2 - 1/r \left( \sum_{j=1}^{s} v'_1 v'_2 \right)
\]

= \frac{r}{s} \left( \sum_{j=1}^{s} v'_1 v'_2 / r - \left( \sum_{j=1}^{s} v'_1 / r \right) \left( \sum_{j=1}^{s} v'_2 / r \right) \right)

= r \text{ cov} (V'_1, V'_2) \) which may be positive or negative

So \( N_{pk} > N_u \) if \( V'_1 \) and \( V'_2 \) are positively correlated, i.e. if the values of \( A \) in \( R_1 \) and \( R_2 \) tend to be abundant and sparse for the same values. If the opposite is the case, i.e. high occurrence of \( V'_1 \) in \( R_1 \) correspond to low occurrences of \( V'_2 \) in \( R_2 \) and vice versa, then \( N_{pk} < N_u \).

(ii) Compare \( u \) and \( N \)

\[
N_{pu} - N_u = 1/s \sum_{i=1}^{m} c'_i c'_2 - 1/r \left( \sum_{i=1}^{m} c'_i \right) \left( \sum_{i=1}^{m} c'_2 \right)
\]

\[
= m / s \left( \sum_{i=1}^{m} c'_i c'_2 / m - \left( \sum_{i=1}^{m} c'_i \right) \left( \sum_{i=1}^{m} c'_2 \right) / m \right)
\]

= \frac{m}{s} \text{ cov} (C_1, C_2)

So \( N_{pu} > N_u \) if \( C_1 \) and \( C_2 \) are positively correlated and \( N_{pu} < N_u \) if \( C_1 \) and \( C_2 \) are negatively correlated.

This is for the same reason as before. Moreover if there are a lot of values in \( R_1 \) which are not represented in \( R_2 \) then these make a zero contribution to \( N_{pu} \) but a non-zero contribution to \( N_{pk} \). Similarly if there are partitions \( D \) in \( R_1 \) or \( R_2 \) which have no values then these make zero contribution to \( N_{pu} \) but a non-zero contribution \( N_{pk} \).

Hence negative correlation leads to \( N_u < N_{pu} \)

Positive correlation leads to \( N_{pk} > N_u \)

(iii) Compare \( pk \) and \( pu \)

\[
N_{pk} - N_{pu} = \sum_{j=1}^{s} v'_1 v'_2 - 1/s \sum_{i=1}^{m} c'_i c'_2
\]

= \frac{1}{s} \sum_{i=1}^{m} \left( \sum_{j=1}^{s} v'_1 / r - \left( \sum_{j=1}^{s} v'_1 / r \right) \left( \sum_{j=1}^{s} v'_2 / r \right) \right)

= \frac{1}{s} \sum_{i=1}^{m} \text{ cov} (V'_1, V'_2)

where \( \text{cov} (V'_1, V'_2) \) is the covariance of values \( V'_1 \) in \( R_1 \) and \( V'_2 \) in \( R_2 \) when \( V'_1 \) and \( V'_2 \) are the appropriate values in partition \( D_i \).

So \( N_{pk} > N_{pu} \) if values \( V'_1 \) and \( V'_2 \) are positively correlated for each partition and \( N_{pk} < N_{pu} \) if values \( V'_1 \) and \( V'_2 \) are negatively correlated for each partition.

**Conclusion**

Therefore, if there is positive (negative) correlation between \( V'_1 \) and \( V'_2 \) for all values, \( V'_1 \) and \( V'_2 \) for each partition and a corresponding correlation between \( C_1 \) and \( C_2 \) we have

\[
N_{pk} > N_{pu} > N_u \quad (N_{pk} < N_{pu} < N_u)
\]

Notice that \( V'_1 \) and \( V'_2 \) correlation does not necessarily agree with (i.e. "positive when positive", etc) the \( C_1 \) and \( C_2 \) correlation, and this accounts for some of our experimental results and also for the fourth result in the sample below.

**Example 3.1 Four illustrative cases**:

In each case, the first column is the multiplicity of the values for relation \( R_1 \) and the second column is the multiplicity for relation \( R_2 \).
Let \( r = 4, m = 2, s = 2 \)

<table>
<thead>
<tr>
<th>( V_1 )</th>
<th>( V_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

\( N_{pk} = 30 \) \( N_p = 25 \) \( N_{pu} = 29 \)

All +ve correlation
\( N_{pk} > N_p > N_u \)

<table>
<thead>
<tr>
<th>( V_1 )</th>
<th>( V_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\( N_{pk} = 30 \) \( N_p = 25 \) \( N_{pu} = 26 \)

C's and V's are correlated (all partitions also)
\( N_{pk} > N_p > N_u \)

Order OK
but not such a good estimate

<table>
<thead>
<tr>
<th>( V_1 )</th>
<th>( V_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\( N_u = N_{pk} \)
\( D_1 \) is -ve, \( D_2 \) is -ve so
\( N_{pk} < N_{pu} \)

C's are +ve correlated so \( N_u < N_{pu} \)

Note that \( N_{pu} \) is bigger than both \( N_{pk} \) and \( N_u \)

This shows that \( N_{pu} \) can be further away from \( N_{pk} \) than \( N_u \) due to the correlation sign.

4. Experimental Validation of the Method

The pu method along with other estimation methods such as perfect knowledge, uniform assumption method, and the worst case methods were implemented using C programming language under the UNIX operating system. The aim of this implementation was to conduct a series of pilot studies to compare the simulated results generated by different estimation methods against the result generated by the perfect knowledge method. In the study we used a query tree based on the one given in [8] but with different distribution types to represent the joining attribute values. The distribution types examined were combinations of Gaussian (Normal), Poisson, Uniform, and Gamma. The experimental results have shown that the pu estimation technique gives a better result than the other estimation methods used in our study. The detailed results of the experiments are given in Table 1.

<table>
<thead>
<tr>
<th>Distribution type</th>
<th>% of cases in which pu method is better than u method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma</td>
<td>75</td>
</tr>
<tr>
<td>Normal</td>
<td>85</td>
</tr>
<tr>
<td>Poisson</td>
<td>85</td>
</tr>
<tr>
<td>Uniform</td>
<td>100</td>
</tr>
<tr>
<td>Mixed</td>
<td>85</td>
</tr>
</tbody>
</table>

TABLE 1: Summary of results of comparison between the pu method and u method (with respect to pk method)

In the table we compare the u method and the pu method in terms of the number of times that the pu method gave a better (nearer pk) estimate than the u method. Using Epstein's tree we did the comparison after two levels and after three levels. Epstein's tree was as follows:

![Epstein's tree diagram]

The first two joins were carried out on join attribute and the third was on the active attribute.

For each distribution type we carried out 20 experiments. We can observe from the table that the general pattern of pu lying between pk and u predicted in Section 3 is in fact validated by the results. A particularly interesting result was that for uniform attribute value distributions the pu always gave better results than u at the second level and that in only 15% of cases (ie 3 out of 20) did the u method give a better estimate. Overall, in 86% of the cases at the first level the pu method showed an improvement, and the figure was 85% of the second level.

The fact that there were several cases where u method out-performed the pu corresponds with the fact that it is possible to have individual values correlated and yet not have correlation between pieces (ranges of values) as predicted in the theory.

When we enhanced the pu method to reflect the active attribute correlations with the joining attribute we got improved results as we shall see in Section 5 (Table 3).
5. Correlations between Attributes and their Effects on the Piecewise Uniform Estimates

These correlations are important when the distribution of the attribute in a relation used for a join at a level higher than the bottom level of the query tree has to be estimated and the join attribute is different from those used on lower levels. It is also important in applying both horizontal and vertical restriction operations before each of the joins. In this section we give some justification for our scaling step (Step 3) of Algorithm Join-Size-Estimation. As before we denote the join attribute in relation \( R_i \) by \( A \) and the active attribute by \( A' \). We are interested in the distribution of the \( A' \) values in a join of \( R_i\) with another relation \( R_j \) on column \( A \). In Algorithm Join-Size-Estimation the assumption was made of "uniform correlation" in a rather special sense. This sense is illustrated below where the diagram shows the pattern assumed for the \( A' \) values over 2 of the \( A \)-partitions. This \( A' \)-distribution is repeated over all the \( m \) (say 10) \( A \)-partitions. Four values of \( A' \) are assumed, values 1-4.

<table>
<thead>
<tr>
<th>( A' ) value</th>
<th>( A' ) distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

There are conditions under which scaling in this way is guaranteed to be valid. This is illustrated by the following proposition (a special case).

**Proposition:**

If the values of \( A \) are uniformly distributed in \( R_i \) then the scaling step of Algorithm Join-Size-Estimation can be safely used.

This is demonstrated by examining the distribution of values of an active attribute after joining (see [3, 16]).

More general examples and analysis are given in [16] to illustrate the wider value of this estimate for piecewise uniform distribution under the above "uniform correlation" assumption.

These examples demonstrate that for quite a range of distribution correlations the scaling technique works well. More rigorous tests will be reported in the next Section.

For more accurate representation we assume that the marginal distributions of \( A \) and \( A' \) are piecewise uniform, as before. Within each partition \( D_i \) of \( A \) and \( E_i \) of \( A' \) we assume that values of \( A \) and \( A' \) are piecewise independent. This provides an enhanced method of correlation handling for our Algorithm Join-Size-Estimation which also covers the previous case.

As an alternative to the "uniform correlation" assumption made earlier, where we assume that each individual \( A' \) value is evenly spread across the pieces, we now store the \( pu \) distribution of \( A' \) for each piece of the \( A \) distribution (\( A \)-partition) accurately.

The implications of this are however important. We pay for the enhanced accuracy of estimation by having to store a \( 10 \times 10 \) correlation table (\( m=10 \)) as well as the \( A \) and \( A' \) tables in our data dictionary. In fact there could be an \( A'' \) attribute which is active within the \( A' \)-join attribute (i.e. used in a higher level join than \( A' \)). Then, at first sight, a \( 10 \times 10 \times 10 \) table would be needed, and so on! There clearly must be an upper limit to these levels for practical systems. We think this should probably be at the second \( (10 \times 10) \) level. Anything beyond that should use the uniformity assumption or the uniform correlation assumption. However we have found that there are some interesting properties of the \( AA' \) pattern of storing the data, which are of advantage in limiting the size of the statistical summary overheads and yet facilitating flexible ad-hoc query evaluation as far as possible. For example if there are \( n \) attributes in a given relation, only \((n-1)\) parameter vectors are required to hold all

\[
\sum_{i=1}^{n-1} (n-i)
\]

inter-attribute piecewise correlations (details are found in [16]).

The effect of this is ameliorated by the fact that some of the distributions may be uniform and so \( u \) rather than \( pu \) can be used at the bottom level, for example. In other small files \( pk \) can be used.

For very sparse distributions, e.g. ISBN numbers none of these methods \( pk, pu, u \) are likely to be useful and a method such as a tuple-substitution query modification could be used. Alternatively, the estimation can be done by the DBA estimating the likelihood of hit of, say, a small book file on a large ISBN file.

Clearly some heuristics are needed to govern the application of each technique. We have implemented the query optimiser of EDDS in PROLOG to provide the required flexibility [10, 16].

If we allow only two levels, as suggested above then a simple adjustment to the original algorithm is all that is needed to support this increased modelling power.
The 100 experiments carried out for the original pu method were repeated for the enhanced pu method. The results in Tables 2 and 3 were obtained when the A and A' distributions were generated independently. This case does not show the full value of the enhancement which is primarily aimed at reflecting the correlations between A and A' when the values are dependent. However it can be seen that the enhancement considerably improved the performance of the method. On average the enhanced version outperformed the original about twice as many times as vice-versa. The overall ratios were 67–33 for the two-level join and 63–37 for the three-level join.

<table>
<thead>
<tr>
<th>Distribution type</th>
<th>% of cases in which enhanced pu method is better than original pu method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2-level join</td>
</tr>
<tr>
<td>Gamma</td>
<td>55</td>
</tr>
<tr>
<td>Normal</td>
<td>75</td>
</tr>
<tr>
<td>Poisson</td>
<td>80</td>
</tr>
<tr>
<td>Uniform</td>
<td>65</td>
</tr>
<tr>
<td>Mixed</td>
<td>60</td>
</tr>
</tbody>
</table>

**TABLE 2:** Summary of results of comparison between the enhanced pu method and the original pu method, for the case when the A and A' values are generated independently

Table 3 shows how the enhanced method improved upon the u method. In about 91% of the 2-level cases and about 86% of the 3-level cases, the new method is superior to the methods making the uniformity assumption.

<table>
<thead>
<tr>
<th>Distribution type</th>
<th>% of cases in which enhanced pu method is better than u method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2-level join</td>
</tr>
<tr>
<td>Gamma</td>
<td>85</td>
</tr>
<tr>
<td>Normal</td>
<td>85</td>
</tr>
<tr>
<td>Poisson</td>
<td>95</td>
</tr>
<tr>
<td>Uniform</td>
<td>100</td>
</tr>
<tr>
<td>Mixed</td>
<td>90</td>
</tr>
</tbody>
</table>

**TABLE 3:** Summary of results of comparison between enhanced pu and u (with respect to pk)

When the data-sets for the experiment were generated in such a way that the A and A' values were dependent, we found that the enhanced method was better than the uniform method in all 100 cases.

The question arises “How easy is it to obtain the pu correlation table?” Clearly the answer is that a single-pass algorithm can be written to obtain it from the correlation table (which is, of course, available directly from the original relation R_i). This single pass can also be used to collect all of the other statistics required for the estimation.

6. **Summary and Future Development**

In this paper, we have presented an intuitive and new method of estimating the intermediate relation sizes for a relational query tree. We have not only considered the estimation for joining attribute distributions but also the estimation of the distribution for the correlated attributes at each node of the tree. The pu method overcomes the unrealistic attribute independence assumption. Analysis and empirical studies using several well known distribution types when generating data values, have been carried out to justify the pu technique. It was found to give much better estimation than that of the best of the alternative methods (other than the perfect knowledge method).

The pu has been implemented using Prolog programming language as part of the query optimisation module for EDDS [1].

Statistical information used for the query optimisation is captured by the global DBA using the CREATE STATS command, an extended SQL syntax definition.

When this command is entered, EDDS prompts the global DBA for entry of a minimal set of statistical data. This data upon validation will be stored in statistical files in the data dictionary.

The query optimisation of EDDS uses the semi-join concept based on a modified version of SDD-1 algorithm [4] and it has two main global optimisation stages i.e. the transformation and estimation optimisation stages. At the transformation optimisation stage, EDDS transforms the global SQL statement (posed by a user) into an equivalent or improved relational algebraic form of n-ary tree. The optimisation stage involves the transformation of n-ary tree to an optimised binary tree with each node containing the estimated cardinality of the intermediate relation size. The estimated cardinality of the intermediate relation at each node is generated by the pu technique. Knowing these cardinalities, EDDS then selects the most suitable site for execution of each node based on minimal transmission cost strategy.

In order to estimate the intermediate relation size resulting from a semi-join operation, the probability of each common attribute value in the relations is required. In [7] all these values are stored as probability vectors in the perfect knowledge method, thus leading to an accurate estimation of the intermediate relation size but incurring a huge system's overhead. On the other hand, the pu method uses a simple intuitive calculation to estimate the probability vector for each existing value
based on assuming the equality of occurrence of values within a location array or a piece (i.e. we consider that the number of tuples for a particular value in a location could range from zero to twice the average number of tuples per value) and that they are equally likely to occur. Therefore the probability of obtaining a zero tuple count for a particular value in the location array is

\[
\frac{1}{(2 \times \text{average tuples per value} + 1)}
\]

whereas, the probability of obtaining a non-zero tuple is

\[
1 - \frac{1}{(2 \times \text{average tuples per value} + 1)}
\]

For example, let \( c_i \) be the actual number of tuples of a "piece" of array in relation \( R_i \), and \( c_j \) be the number of tuples in a corresponding piece of array in relation \( R_j \). Let these two pieces be associated with the same value range, say \( s \). For \( R_i \) semi-join over \( R_j \), the probability of non-zero values over a given range of \( R_i \), is

\[
1 - \frac{1}{(2 \times \frac{c_i}{s} + 1)}
\]

The estimated intermediate relation size is now calculated as

\[
\sum (c_j / s) \times (1 - \frac{1}{(2 \times (c_j / s) + 1)}) \times s
\]

However this assumption can be refined to give a better approximation for the semi-join operation [16].

Our proposed future research and development work includes size estimation for queries whose results contain null values, historical data in accordance with our temporal data model, and object oriented data values. We are also examining the preservation of data integrity in a DDB environment during updating of the statistical data.

The pu method will also be implemented in the Multi-Star system [2] as part of its query optimisation module. The Multi-Star system is a distributed heterogeneous relational DBMS developed by CRAI, Italy and will be commercially available within 2 years.

Acknowledgements This work was carried out with the support of the EEC Multi Annual Programme (Project 773B).

REFERENCES


