Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

STATISTICAL RULES: A NOTION OF DATABASE ABSTRACT AND ITS ROLE IN QUERY PROCESSING

Chung-Dak Shum
Richard Muntz

January 1989
CSD-890007

Statistical Rules: a Notion of Database Abstract and its Role in Query Processing

Chung-Dak Shum

Richard Muntz

Computer Science Department
Unversity of California, Los Angeles

Abstract

A database instance is not an arbitrary collection of data, but rather many correlations exist among data items. The notion of *statistical rules* is introduced as a means of expressing such relationships. We demonstrate that statistical rules can be utilized in the query optimization process. In selectivity factor estimation, for example, statistical rules can actually be used to introduce relevant attributes the same manner as exact rules in semantic query optimization. Other uses of statistical rules include the enhancement of parallelism in database machines, and providing incomplete/quick answers as well as more informative responses.

We quantify the notion of how to measure the "inexactness" of a statistical rule using an entropy measure. The lower the entropy or uncertainty of a rule, the better the rule is. Based on such a measure, we show that constructing statistical rules using a "greedy" algorithm will result in a reasonable, although perhaps not optimal rule.

1. Introduction

Query optimization can be regarded as the problem of selecting an efficient evaluation plan to process a query (usually expressed in logical terms) from among the alternative plans which can be carried out in the physical database. Conventional query optimization methodologies approach this problem by integrating a large number of techniques, ranging from algebraic transformations of queries to the optimization of access paths and the storage of data on the file system level.

In order for an optmizer to select an efficient access path query, fairly accurate selectivity estimates must be available. In [6], simple statistics, such as the miniumum and maximum values of an attribute, are used to estimate selectivity factors. However, using such simple statistics will produce good selectivity estimates only if the attribute values are uniformly distributed. Since attribute values can have other distributions, it is suggested that [5] histogram can be used to more accurately estimate selectivity factors. In [5], the problem of building equi-depth histogram on a single attribute has been well studied.

For queries involving multiple attributes, most database systems assume that attributes are independent [13]. This assumption leads such systems to systematically overestimate the costs of queries and thus to select query plans that substantially increase the queries's processing time. In [13], the concepts of Schur concavity and majorization are used to efficiently estimate the cost of a query when the queried attribute is correlated with the clustering attribute. However, the optimizer has to maintain a representative block access distribution that can be modified based on the type of query it is presented with. Thus, if there is a large variety of queries, it is not clear how "representative" this empirically tabulated block access distribution can be.

Another approach to this attribute dependence problem is to extend the idea of single-attribute histogram [12] to histograms of multiple dimensions. There are two potential draw-

backs with multi-dimensional histograms. First, as the number of dimensions increases, the problem of storing this large multi-dimensional structures becomes increasing difficult, especially if we want to keep them in main memory. Second, although we expect many correlations exist among data items in a database, it is very unlikely that these correlations occur uniformly across the data. Thus, multi-dimension histograms may not be an efficient way to capture such correlations.

In this paper, we suggest the notion of *statistical rules* to capture the correlations among data items. An example of a statistical rule may take the following form:

R1: "80% *of employees who earn over $*35000 *are engineers*".

It basically says if there are 1000 employees who earn more than $35000, 800 of them are engineers. Notice that information is not exact or precise with respect to characterizing particular individuals. Given an employee who earns over $35000, and **R1**, we cannot tell whether he is an engineer or not. One of the contribution of this paper is to quantify the notion of "inexactness" in a statistical rule using an entropy measure. With such an measure, we can compare statistically rules and construct rules that are more informative or better captured the correlations among data items.

There is a similarity between statistical rules and semantic rules used in query optimization. They both reflect certain correlations existed among data items in a database. But semantic rules are exact rules in the sense that they imposes or ascribe a certain property to *all* or *none* of the individuals in a set of entities. For example,

ER1: "*All employees who earn over $35000 are engineers*"

is an exact version of **R1**. Recent research in semantic query optimization [1,2,3] has demonstrated the advantage of incorporating semantic knowledge about database in the optimization process. The process of semantic query optimization can be described in two phase: the query transformation phase and the query plan selection phase. In the query transformation phase,

semantically equivalent queries are generated. It is in the transformation of queries that semantic rules are involved. Statistical rules fail to be of use here because of their inexactness. In the query plan selection phase, however, we discover that statistical rule can be very helpful. As will be seen later, the manner in which statisical rules are used requires that they closely, but not necessarily precisely, reflect the current instance of the database. Thus, update is much less of a problem than in the case of exact rules.

The organization of the paper is as follows. Section 2 describes through examples, the role of statistical rules in query optimization. Section 3 studies quantitatively the preciseness or information content of such statistical rules. Section 4 suggests other interesting, non-conventional applications in which statistical rules are important. Section 5 summarizes the paper.

2. The Role of Statistical Rules in Query Optimization

In this section, we will demonstrate some uses of statistical rules in query optimization. Before we do that, we need to be more specific about the kind of statistical rules that we are interested in. Consider, for example, a richer version of the statistical rule **R1**

R2: *Engineers salary profile:*

Salary Range	Ratio of Engineer : Employee
[5000-15000]	1 : 20
[15000-35000]	1 : 3
[35000-70000]	8 : 10

Notice that the third entry in **R2** is actually the same as **R1**. In fact, the three entries of **R2** can also be regarded as three different statistical rules. But since they form a partition on salary range, we group them together as one rule. Now if the salary of employees actually ranges from $5000 to $70000, we call **R2** *complete*. For convenience, we will, from here on, deal with complete statistical rules, although the same ideas also apply to all statistical rules. From another

perspective, **R2** can be seen as approximating how dense the engineers are with respect to all employees in various salary ranges. In the next section, we illustrate how this information can be used in query optimization.

2.1 Selectivity Factor Estimation

In the query selection phase, the cost of carrying out each query plan is estimated and the one with the lowest estimated cost is selected. In estimating the cost of evaluating a query plan, it has been pointed out [6] that an estimation of the number of tuples satisfying a condition can be very useful. Now we show that statistical rules can actually help in making such an estimation. Let us begin with a simple example. Suppose, as before, we have the *Employee* (*employee_name*, *job_category*, *salary*, *education* , $\cdots$) relation and a complete statistical rule **R2**. Further suppose we have a five-bucket equi-depth histogram [5] describing the distribution of attribute *salary*. Say the number of employees in each bucket is 360 and the widths of the buckets are:

[5000-20000], [20000-30000], [30000-40000], [40000-50000], [50000-70000]

We are interested in the number of engineers who earn over \$30000. Given **R2** and the above histogram as the only available information, we can estimate that approximately 60+720 engineers earn over \$30000. First, from **R2**, we know that $\frac{1}{3}$ of the employees who earn between \$15000 and \$35000 are engineers. Then, from the histogram there are $360 \times \frac{35000-30000}{40000-30000}$ such employees. Thus, we estimate 60 engineers earn between \$30000 and \$35000. The 720 can be obtained similarly. Notice that in the above estimation, we have assumed that engineers are uniformly distributed within the salary range [15000-35000]. This raises an important question of how accurate the estimate is or how much information **R2** carries. Intuitively, it would seem that the larger the number of intervals the salary range is being partitioned into, the more accurate we get. However, a more interesting question is: if we are given the opportunity to choose a

fixed number of intervals, how should they be chosen and how do we quantify their preciseness? We will address all these issues later in Section 3.

Now consider a more complicated example. Suppose in addition to **R2**, we also have

R3: *PhD salary profile:*

Salary Range	Ratio of PhD : Employee
[5000-40000]	0 : -
[40000-70000]	2 : 3

The first entry implies no PhD earns less than 40000. We want to estimate the number of tuples in the *Employee* relation satisfying the conditions: $job\_category = engineer$ and $education = PhD$. From **R2** and **R3**, we see that engineers and PhD's can be related somehow through their salaries. The second and third column of Table 1 below shows the average number of engineers and PhD's in different salary ranges. They are calculated from **R2**, **R3** and the salary histogram.

Salary Range	Engineers	PhD's	Max	Min
5000-20000	52	0	0	0
20000-30000	120	0	0	0
30000-40000	204	0	0	0
40000-50000	288	240	240	168
50000-70000	288	240	240	168

Table 1. Combined Salary Profiles of Engineers and PhD's

The third and fourth column are respectively the maximum number and minimum number of PhD engineers. It is easy to see that the maximum number of PhD engineers cannot exceed either the number of engineers or the number of PhD's. The minimum, on the other hand, is the minimum intersection of the two. In this case, since each salary range has 360 employees, the **Min** for salary ranges [40000-50000] and [50000-70000] is $288 + 240 - 360$. Thus, the total number of *Employee* tuples satisfying the conditions must be no less than 336 and no greater than 480. If only the distributions of engineers and PhD's are known, the best that can be said

about the number of PhD engineers is that it is between 0 and 480.

In this last example, the attribute *salary* is not even mentioned in the selection conditions of the *Employee* relation and it may not appear anywhere in the query. It is only by the use of statistical rules that we can introduce it to the query optimization process. Let us call such attributes *relevant attributes* [1]. Note that conventional semantic query optimization also involves the introduction of relevant attributes to a query, but with the restriction that the rules being utilized there are exact. Another important observation is that there is no requirement that the relevant attribute has to be in the same relation. From the above example, there is no reason why it should be. More specifically, if the *salary* attribute is not in the *Employee* relation, the same selectivity factor estimation can be carried out. Further, a heuristic for generating relevant attributes in [1] suggests it is preferable if such attributes are contained in relations mentioned in the original query. The rationale behind this is that otherwise an extra join may be required, which may be proven expensive. In our case, since we are only estimating selectivity factors, no join is involved and thus the warning does not apply here.

2.2 Concurrent Operations

The accurate estimation of selectivity factors [6] is unarguably one of the most important issues in query optimization. In the last section, we demonstrate that statistical rules can actually help in providing better selectivity estimates. However, the benefit we can obtain from such rules is more far reaching, and we will further explore their use in this section. Their benefit in less conventional applications will be discussed in Section 4.

Consider a simple database machine with hardware configuration as shown in Figure 1. It consists of a high bandwidth communication network interconnecting a number of basic components. Each basic component has a processor and a disk, which is used for database storage.

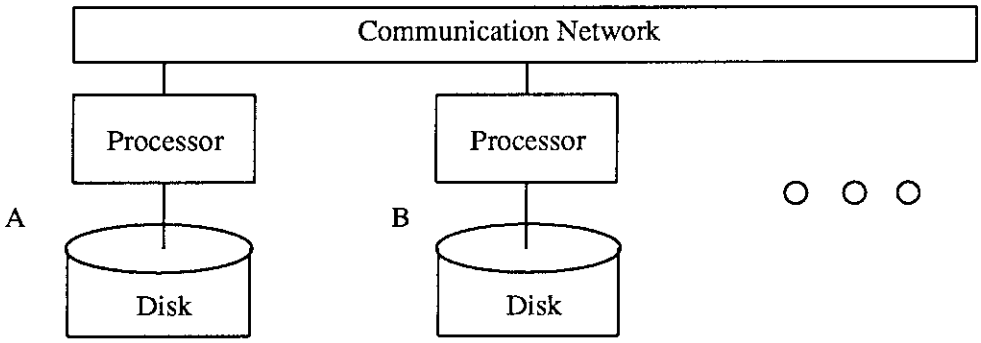

Figure 1. Simple Database Machine

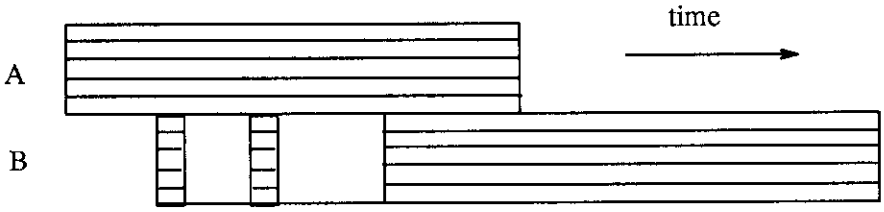

Figure 2a. I/O operations of component A and B

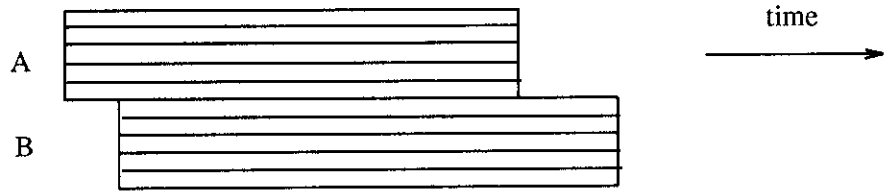

Figure 2b. More overlapped I/O operation

Associated with each processor is some amount of local memory. We assume that communication cost is negligible and the cost of processing a query is still measured in I/O operations.

Let us begin with a simple example. Suppose we have to perform a selection operation on one relation followed by a join with some other relation. Say our familiar *Employee* relation is the first relation and *job_category = engineer* is the desired selection and let us call the other relation *R*. Assume that the *Employee* relation has its attribute *salary* as the only clustered index. Further, assume that the relations *Employee* and *R* reside on components *A* and *B* respectively, where *A* and *B* are interconnected by the communication network. Now suppose a query optimizer suggests, perhaps based on selectivity factor, that the selection operation should be performed in component *A* and the results of the selection should then be sent to component *B* for the join operation. If the query is evaluated as planned, we show in Figure 2a, the I/O operations of components *A* and *B* as a function of time. The shaded region represents the time during which I/O operations are being performed.

Note it is possible that the I/O operations of the two components may exhibit very little concurrency. First, since *job_category* is not indexed, a segment scan will have to be performed on the *Employee* relation to select the engineers. Now if engineers are sparse at the beginning of the segment and become increasingly more dense towards the end, a lot of the join operations, which account for most I/O's in component *B*, can only occur in the latter part of the segment scan. This is illustrated in Figure 2a. Since attribute *salary* is the only clustered index, statistical rule **R2** provides exactly this information. If **R2** is available to the optimizer, the segment scan can be done in such a way that component *B* will be kept busy all the time. More specifically, instead of scanning a segment in its physical order, we can make use of the information provided by statistical rules, like **R2**, partition the segment into a number portion and scan those portions with higher selectivity factor first. Figure 2b shows the time savings involved if such information is utilized.

Now consider another example. To facilitate more efficient processing of queries, relations in database machines are usually horizontally partitioned across a number of disks in the system. For example, a hash function can be applied to the attribute *job_category* of each tuple in our *Employee* relation to select a storage component. The resulting partition may have engineers in one component, managers in another component and secretaries in still another component. The main purpose of partitioning data across a number of components is to exploit any potential parallelism. Say we want to join another relation *R* with our *Employee* relation. All that is required is to send *R* to each component which contains a fragment of the *Employee* relation and the join operation can take place concurrently.

Suppose, for a slightly more complicated case, we are to select employees whose salary is above $35000 before joining with *R*. Assume that there is a strong correlation between job category and salary; for instance, most engineers earn over $35000, while only few others do. Let us call the relation after the selection operation: the *High_Pay_Employee* relation. The distribution of the number of *High_Pay_Employee* tuples in each component is highly uneven. If the join is performed, the component containing engineers will certainly have the majority of work, and parallelism cannot be fully exploited. Of course, if such information is available in advance to the query optimizer, the engineer *High_Pay_Employee* tuples can be partitioned and sent to other components before the join is carried out. Again, statistical rules like **R2** can provide such information.

3. Statistical Rules

In Section 2, we demonstrated several potential uses of statistical rules in query optimization. But we have not addressed the questions: (i) how does one decide which attributes to relate via statistical rules? (ii) what criteria do we use to measure the "goodness" of the rules? (iii) how do we construct the rules? We will suggest, in some detail, approaches to questions (ii) and

(iii) in this section. As for (i), we will only give a few brief comments. In any particular domain of application, it is very likely that there exists certain relationships among different attributes. Statistical rules are intended to capture or summarize the relationships among those attributes that are highly correlated. For example, the job category and the salary of an employee is highly correlated; whereas, the social security number and the salary of an employee is probably not correlated at all. The first question basically asks how to acquire these sets of highly correlated attributes. An obvious way is, of course, to rely on the domain expert to supply the information. Another way is to extract the information automatically [8].

Suppose job category and salary are highly correlated attributes, we can construct rules like **R2** for each job category. However, with a different partitioning of the salary range, we can also have

R2′: *Engineers salary profile:*

Salary Range	Ratio of Engineer : Employee
[5000-30000]	1 : 4
[30000-50000]	1 : 2
[50000-70000]	8 : 10

The obvious question is "which is a better rule?" This is exactly question (ii) mentioned earlier. Intuitively, the closer the ratios are to 0 or 1, the more informative the rule is. For example, given an employee with salary range [50000-70000], we are almost sure that he is an engineer. On the other hand, it is very uncertain whether an employee with salary range [30000-50000] is an engineer or not. We will introduce a formal measure of "goodness", and describe approaches to construct such "good" rules in the next two subsections.

3.1 Information Measure

Let us restate the problem more abstractly and be explicit about the kind of information we want to measure. Consider a continuous range, say [0,10], and ten samples such that each sample takes on a value from the range. There are two types of sample, marked and unmarked. Figure 3 shows an example distribution of the samples on the range. Now suppose the range is partitioned into intervals and for each interval we are given the ratio of the marked samples to the total number of samples in that interval. Notice that this is exactly the same as a statistical rule. If there are k intervals, we call it an *k-interval statistical rule.* The information of interest is *given the statistical rule and the position of a sample, what can be said about the state of the sample, that is, marked or unmarked?* If the range in Figure 3 is partitioned into three intervals, I1, I2 and I3 as shown, the question above can be answered exactly. For the two-interval partition, given that a sample is in interval J1, we can only say that there is a $\frac{1}{4}$ probability that the sample is marked. Similarly, given that a sample is in interval J2, the probability that the sample is marked is $\frac{3}{6}$. Apparently, the latter partition has more uncertainty with respect to this marked/unmarked question. Next we try to quantify this uncertainty.

Informally, in the language of probability theory, a ratio of marked/unmarked samples in a given interval can be viewed as describing a finite probability space, composed of two mutually exclusive events E_1 and E_2 and their associated probabilities. E_1 is the event that a randomly selected sample from the interval is marked and its associated probability is

$$p_1 = \frac{\text{number of marked samples in the interval}}{\text{total number of samples in the interval}};$$

whereas, E_2 is the event that the sample is not marked and its associated probability $p_2 = 1 - p_1$. It is well-known [7] that entropy

$$H(p_1,p_2) = -(p_1 \log p_1 + p_2 \log p_2)$$

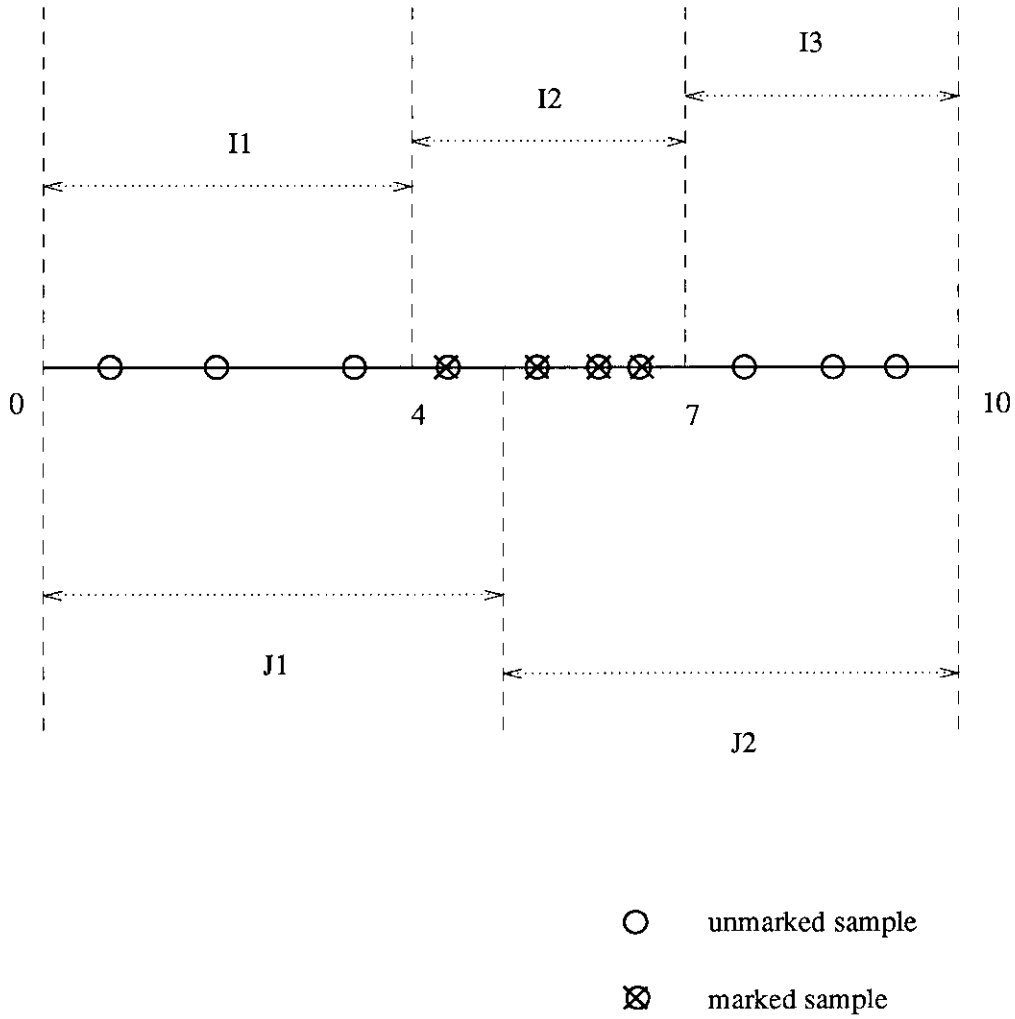

Fig 3. Distribution of Mark/Unmarked Samples

is a very suitable measure of the uncertainty involved. First of all, we see immediately that $H(p_1,p_2)=0$ if and only if one of p_1,p_2 is one and the other is zero. But this is just the case where there is no uncertainty as to its outcome. In all other cases the entropy is positive. Furthermore, it is obvious that the case with most uncertainty is the one with equally likely outcomes, that is, $p_1=p_2=\frac{1}{2}$, and indeed, in this case, the entropy assumes its largest value [9]

$$H(p_1,p_2)\leq \log 2 = H(\tfrac{1}{2},\tfrac{1}{2})$$

When the marked/unmarked question is asked, we assume that the samples are chosen with equal probability. Thus the probability that the sample is in a particular interval is equal to the fraction of samples in that interval. Thus if we are interested in the average uncertainty involved, a weighted average of the uncertainties of all intervals is appropriate. In fact, this is exactly the definition of conditional entropy.

Definition 3.1 Let **S** and **R** be two finite probability spaces with events $\{S_i\}$ $(i=1,2,\cdots,n)$ and $\{R_k\}$ $(k=1,2,\cdots,m)$, respectively. Then the *conditional entropy* of the space **S** averaged over the space **R** is

$$H_{\mathbf{R}}(\mathbf{S})=\sum_{k=1}^{m} p(R_k)\,H_{R_k}(\mathbf{S})$$

Here, **S** is the probability space for describing whether a sample in a given interval is marked or unmarked; R_k is the event that a randomly picked sample belongs to the k-th interval; and $H_{R_k}(\mathbf{S})$ is the uncertainty associated with the k-th interval. Defining **S** and **R** accordingly for the two-interval statistical rule in Figure 3, we get

$$H_{\mathbf{R}}(\mathbf{S})=\tfrac{4}{10}H(\tfrac{1}{4},\tfrac{3}{4})+\tfrac{6}{10}H(\tfrac{3}{6},\tfrac{3}{6})=0.811$$

More formally, we have the following definition:

Definition 3.2 Let S be a statistical rule with sample size n. For each interval T_i ($i = 1,2, \cdots, n'$) of S, m_i out of s_i samples are marked. Then the *entropy* for S is

$$H(S) = \frac{1}{n} \sum_{i=1}^{n'} s_i H(\frac{m_i}{s_i}, 1 - \frac{m_i}{s_i})$$

It is not difficult to see that statistical rules with equal number of intervals can have very different entropies.

3.2 Minimum Entropy Approach

Now the problem of interest here is:

OPTIMAL STATISTICAL RULE
We are given a continuous range, and a collection of samples in which each sample takes on a value from that range. For a fixed number of intervals, find the statistical rule that has the minimum entropy.

Since the number of intervals is fixed, the various statistical rules corresponds to different ways of partitioning the range. If n is the sample size and m is the number of desired intervals, the number of possible statistical rules is $\binom{n-1}{m-1}$. Even for a moderate sample size and a small number of intervals, the number of statistical rules precludes enumeration. Let us study the OPTIMAL STATISTICAL RULE problem carefully and see if there is any way to reduce the number of statistical rules that we have to examine. If the samples are distributed as shown in Figure 4, and a two-interval OPTIMAL STATISTICAL RULE is desired, it should be obvious that the partition point has to occur between the marked and unmarked sample.

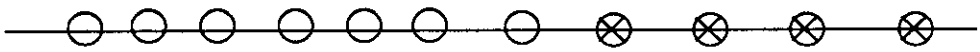

Fig 4. Two Groups of Samples

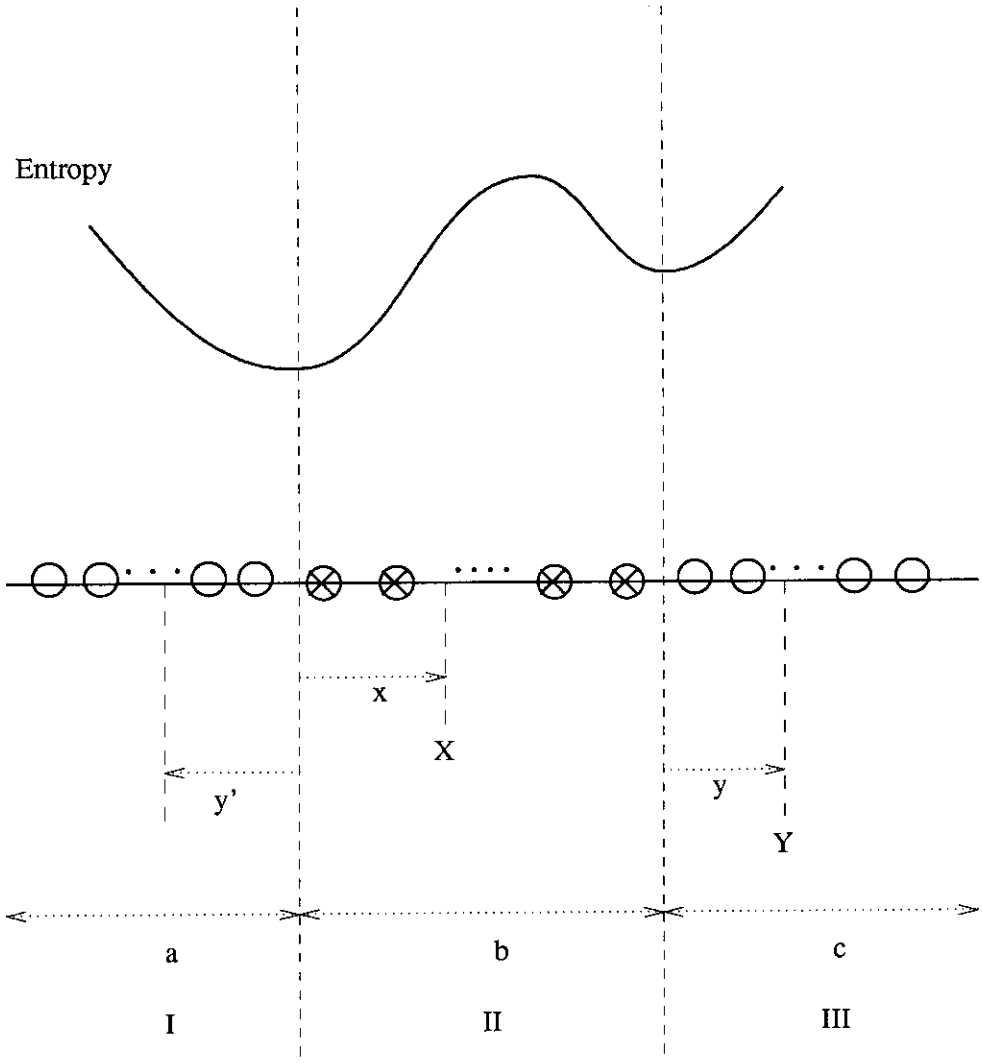

Fig 5. Three Groups of Samples

Lemma 3.1 If the marked and unmarked samples form two groups (Figure 4), the partition point of a two-interval OPTIMAL STATISTICAL RULE must occur between the groups.

Proof: It is easy to see that if the partition point is in between the group, the entropy of that statistical rule is 0. Otherwise, it is greater than 0.

Note that any statistical rule with partition point satisfying this property can be utilized to answer the marked/unmarked question exactly. Otherwise, the answer can only be given probabilistically. The next lemma concerns a slightly more complicated case.

Lemma 3.2 If the marked and unmarked samples forms three groups (Figure 5), the partition point of a two-interval OPTIMAL STATISTICAL RULE must occur between groups.

Proof: Figure 5 shows three groups of samples. Group I has a unmarked samples, group II has b marked samples and group III has c unmarked samples. Our goal is to show that as the partition point varies, the entropy plot, as shown in the figure, has two minima, both of which occurred between groups.

Let

$$E_Y = \frac{a+b+y}{a+b+c} H\left(\frac{b}{a+b+y}, \frac{a+y}{a+b+y}\right) + \frac{c-y}{a+b+c} H\left(\frac{0}{c-y}, \frac{c-y}{c-y}\right)$$

$$= \frac{a+b+y}{a+b+c} \left[-\frac{b}{a+b+y} \log \frac{b}{a+b+y} - \frac{a+y}{a+b+y} \log \frac{a+y}{a+b+y} \right]$$

be the entropy if Y is the partition point. Assume y is continuous for the moment, and differentiate E_Y with respect to y

$$\frac{dE_Y}{dy} = \frac{1}{a+b+c} \log \frac{a+b+y}{a+y}$$

Thus, as y grows the entropy increases. By symmetry, as y' grows the entropy also increases.

Let

$$E_X = \frac{a+x}{a+b+c} H\left(\frac{x}{a+x}, \frac{a}{a+x}\right) + \frac{c+b-x}{a+b+c} H\left(\frac{b-x}{c+b-x}, \frac{c}{c+b-x}\right)$$

Assume x is continuous and its range $(0,b)$. Differentiate E_X with respect to x and

equate it to zero, we discover that E_X has only *one* maximum when

$$x = \left\lceil \frac{a}{a+c} \right\rceil b$$

Thus, the entropy plot must have two minima as shown, and the partition point of a two-interval OPTIMAL STATISTICAL RULE must occur between groups.

We can generalize this observation to the following theorem.

Theorem 3.1 Any partition point for an OPTIMAL STATISTICAL RULE has to occur between a marked and an unmarked sample.

Proof: Similar to Lemma 3.2.

Note that Theorem 3.1 significantly reduce the search space for the OPTIMAL STATISTICAL RULE. If the marked and unmarked samples formed n' groups, the number of statistical rules we have to examine is $\begin{bmatrix} n'-1 \\ m-1 \end{bmatrix}$ where m, again, is the number of desired intervals. Under almost all practical situations, n' would be much smaller than the sample size n. However, for a moderate n' and a small of number intervals, the number of statistical rules is probably still too large to enumerate.

A possible approach here is to consider a relaxed version our original problem; that is, we do not insist on obtaining an OPTIMAL STATISTICAL RULE. If such is the case, a "greedy" algorithm will probably lead to "good", although perhaps not optimal solutions. In fact, we are able to show that for the worst case behavior, the relative error resulted from the "greedy" algorithm is always less than two. The algorithm involves combining adjacent intervals, which is defined as follows.

Definition 3.3 Let S be a statistical rule. For each interval T_i $(i = 1,2,\cdots,n')$ of S, m_i out of s_i samples are marked. Suppose T_j and T_{j+1} are two adjacent intervals of S. We *combine* intervals T_j and T_{j+1} in S to form a new statistical rule S' such that for each interval

$$T_i' = \begin{cases} T_i & 1 \le i < j \\ T_j + T_{j+1} & i = j \\ T_{i+1} & j < i < n' \end{cases}$$

Thus, for interval T_j', $m_j + m_{j+1}$ of $s_j + s_{j+1}$ samples are marked. Combining more than two adjacent intervals can be generalized accordingly.

It is important to see that as intervals are being combined, information is lost and entropy or uncertainty increases.

Proposition 3.1 Let S be a statistical rule with sample size n. For each interval T_i $(i = 1,2,\cdots,n')$ of S, m_i out of s_i samples are marked. Suppose T_j and T_{j+1} are two adjacent intervals of S and T_j and T_{j+1} are combined to form a new statistical rule S'. Then

$$H(S) \le H(S')$$

Proof: Consider
$$H(S') - H(S) = \frac{s_j + s_{j+1}}{n} H\left(\frac{m_j + m_{j+1}}{s_j + s_{j+1}}, 1 - \frac{m_j + m_{j+1}}{s_j + s_{j+1}}\right) - \frac{s_j}{n} H\left(\frac{m_j}{s_j}, 1 - \frac{m_j}{s_j}\right) - \frac{s_{j+1}}{n} H\left(\frac{m_{j+1}}{s_{j+1}}, 1 - \frac{m_{j+1}}{s_{j+1}}\right)$$

Notice also that for any *linear* function $f(\bullet)$, it can be shown

$$(s_j + s_{j+1}) f\left(\frac{m_j + m_{j+1}}{s_j + s_{j+1}}\right) = s_j f\left(\frac{m_j}{s_j}\right) + s_{j+1} f\left(\frac{m_{j+1}}{s_{j+1}}\right)$$

Since $H(\bullet)$ is concave, $H(S') - H(S) \ge 0$ and the proposition follows.

Next we present the algorithm. We made use of Theorem 3.1 to obtain the initial statistical rule S_{in}. In each iteration, the number of intervals is being reduced by one and the stepwise increase in entropy is kept to a minimum.

Input: A statistical rule S_{in} with n' intervals (T_i, $1 \le i \le n'$)

Output: A statistical rule S_{out} with m intervals ($m < n'$)

begin
    $S_{out} := S_{in}$;
    while the number of intervals in S_{out} is greater than m **do**
    begin
        combine intervals T_j and T_{j+1} in S_{out} to form S_{out}' such that
            $H(S_{out}') - H(S_{out})$ is minimum;
        (**comment:** break ties arbitrarily)
        $S_{out} := S_{out}'$
    end
end

Fig 5. Greedy Heuristic for OPTIMAL STATISTICAL RULE

If $n' = m + 1$, the "greedy" algorithm outputs the optimal solution. Suppose $n' = m + 2$. The optimal solution can involve the combination of three consecutive intervals; or, it can involve the combination of two distinct pairs of adjacent intervals. It is easy to see that the "greedy" algorithm will always find the optimal solution if the latter is the case. Now let us consider the former and study its worst case behavior. Assume an initial statistical rule S_{in} with n' intervals (T_i, $1 \le i \le n'$). Suppose the OPTIMAL STATISTICAL RULE involves combining three consecutive intervals T_p, T_{p+1} and T_{p+2}; and the "greedy" algorithm instead combines intervals T_q and T_{q+1} followed by T_r and T_{r+1}. For convenience, let us define the following statistical rules. From S_{in}

i. combine intervals T_p, T_{p+1} and T_{p+2} to form S_{opt};

ii. combine intervals T_p and T_{p+1} to form S_{opt_1}

iii. combine intervals T_{p+1} and T_{p+2} to form S_{opt_2} ;

iv. combine intervals T_q and T_{q+1} followed by T_r and T_{r+1} to form S_g ;

v. combine intervals T_q and T_{q+1} to form S_{g_1} .

Lemma 3.3 $H(S_{opt}) \geq \max (H(S_{opt_1}), H(S_{opt_2}))$

Proof: Apply Proposition 3.1.

Lemma 3.4 $H(S_g) \leq 2 (\min (H(S_{opt_1}), H(S_{opt_2})))$

Proof: Consider the following two facts:

$$H(S_{g_1}) - H(S_{in}) \leq \min (H(S_{opt_1}) - H(S_{in}), H(S_{opt_2}) - H(S_{in})) \qquad \text{(I)}$$

If (I) is not true, intervals T_q and T_{q+1} will not be selected by the algorithm.

$$H(S_g) - H(S_{g_1}) \leq \min (H(S_{opt_1}) - H(S_{in}), H(S_{opt_2}) - H(S_{in})) \qquad \text{(II)}$$

If (II) is not true, intervals T_r and T_{r+1} will not be selected by the algorithm.

Adding (I) and (II), and the lemma follows.

Lemma 3.5 For $n' = m + 2$, the worst case relative error for the "greedy" algorithm is no greater than 2.

Proof: By Lemma 3.3 and 3.4, the worst case relative error

$$\frac{\max (H(S_g))}{\min (H(S_{opt}))} \leq \frac{2 (\min (H(S_{opt_1}), H(S_{opt_2})))}{\max (H(S_{opt_1}), H(S_{opt_2}))} \leq 2$$

Now if $n' = m + 3$, the optimal statistical rule may involve the combination of three consecutive intervals and the combination of another distinct pair of adjacent intervals; whereas, the "greedy" algorithm may involve the combination of three distinct pairs of adjacent intervals. If this is the case, it is easy to see that Lemma 3.5 can be applied and show that the relative error is still no greater than 2. On the other hand, if the optimal statistical rule involves the combination of four consecutive intervals, Lemma 3.5 can also be extended accordingly. Now we state our more general result with regard to the worse case behavior of the "greedy" algorithm.

Theorem 3.2 The relative error resulted from the "greedy" algorithm is no greater than 2.

Proof: By induction and simple extension of Lemma 3.5.

Thus, if we do not insist on obtaining an OPTIMAL STATISTICAL RULE, the "greedy" algorithm will probably lead to reasonable and efficient solution.

4. Other Potential Applications

In this section, we explore other potential applications of statistical rules in less conventional environments.

4.1 Sound and Incomplete Answers

So far, our concern has been on a conventional query answering system. That is, in response to a query, the system has to return a *sound* and *complete* answer [11]. An answer is sound if every listed entity satisfies the query conditions. It is complete if every entity satisfying the query conditions is listed in the answer. Under most circumstances, a user will prefer a sound answer, as decisions based on unsound answer may turn out to be disastrous. However, the same is not necessarily true for a complete answer. In a real-time application, for example, a user may not have the luxury of waiting for a complete answer. Or it may simply be the case that we have an impatient user. Whatever the reason may be, one possibility is to allow the user to specify the minimum proportion or percentage of the answer he wants, and the system is to speedily return the partial answer.

One feasible approach to quickly obtain a portion of the answer is to adopt and extend the concept of semantic query optimization. The idea is to consider *subsumed queries* instead of equivalent queries. By subsumed queries, we mean queries that are contained by the original query. And we are only interested in those that can be efficiently evaluated. Let us illustrate by an example. Consider the *Employee* relation and again assume that the attribute *job_category* is

the only clustered indexed. We also have statistical rule **R2**. Now suppose the query is

"Who earn more than $40000?" (75%)

The percentage at the end of the query merely says that the user wants speedy reply to three quarters of the answer. From the third entry in **R2**, we know that 80% of the employees who earn more than $35000 are engineers. Thus we can form a subsumed query

"Who are the engineers that earn more than $40000?"

Since *job_category* is a clustered index, this last query can be answer very efficiently. Now the question whether the answer to the subsumed query contains no less than 75% of the answer to the original query depends on the how accurate the statistical rule **R2** is. If there are a lot of employees who earn between $35000 and $40000 and almost all of them are engineers, then it is possible that less than 75% of the employees who earn more than $40000 are engineers. However, the more informative the statistical rule is, the less likely this is to occur.

4.2 Informative Responses

An enumeration of individual objects is not always the best means of information exchange. It has been pointed out [11] that, for user responses, a desirable property is succinctness. For example, in response to the query

"Who earns more than $30000?"

instead of having a list of names of all employees who satisfy the query, we can have answer of the form

"$\frac{800}{1000}$ *engineers* + $\frac{100}{100}$ *managers*"

It basically says all managers and 800 out of 1000 engineers earn more than $30000. Notice that *engineers* and *managers* are concepts representing a set of individuals and they both belong to the same class, *job category*, in this case. In a database context, we can interpret *job category* as an attribute and *engineers* and *managers* as values belonging to that attribute. The intuitive ap-

peal of such answers lies in its succinctness. Still, given an individual in any *job category*, we can with some degree of certainty, whether he earns more than $30000 or not. There is no reason why the same idea cannot apply to an attribute with a continuous domain. In such a case, a statistical rule such as **R2** can then be the answer to the query

"*Who are the engineers?*"

Again, given an engineer in a certain salary range, we have an indication whether he satisfies the query or not. The obvious questions are how good the statistical rules are and how they can be constructed as answers to queries. But these are just the same issues we already discussed in the previous section.

5. Conclusions

We identify two types of rules in a database: rules that are exact and rules that are inexact or statistical. It is well known by now that exact rules can be used in semantic query optimization. We suspect that the number of exact rules one can extract from a database is probably limited. On the other hand, there is no doubt that many correlations exist among data items. These correlations are usually not exact and can only be expressed by statistical rules. Fortunately, we demonstrate that these statistical rules can be utilized in the query optimization process. In selectivity factor estimation, statistical rules can actually be used to introduce relevant attributes the same manner as being done by exact rules in semantic query optimization. Other use of statistical rules includes the enhancement of parallelism in database machines.

The "goodness" of a statistical rule, in general, is a vague notion. We quantify such notion using the entropy measure. The lower the entropy or uncertainty of a rule, the better the rule is. Based on the such a measure, we show that constructing statistical rules using a "greedy" algorithm will result in a reasonable, although perhaps not optimal rule. We also outline some less conventional applications in which statistical rule can be useful. These include incomplete/quick answer and more informative kind of responses.

References

[1] King, J., *Query Optimization Through Semantic Reasoning*, Ph.D Dissertation, Stanford University 1981.

[2] Chakravarthy, U.S., *Semantic Query Optimization in Deductive Databases*, Ph.D. Dissertation, University of Maryland 1985.

[3] Siegel, M.D., "Automatic Rule Derivation For Semantic Query Optimization", *Expert Database Systems*, Kerschberg, L. (ed.), Benjamin Cummings, New York, 1989.

[4] Borgida, A., Mitchell, T., Williamson, K.E., "Learning Improved Integrity Constraints and Schemas From Exceptions in Data and Knowledge Bases", *On Knowledge Base Management Systems*, Brodie, M.L., Mylopoulos J. (ed.), Springer-Verlag, New York, 1986.

[5] Piatetsky-Shapiro, G., Connell, C., "Accurate Estimation of the Number of Tuples Satisfying a Condition", *Proc. of ACM-SIGMOD* 1984.

[6] Selinger, P.G., et al. "Access Path Selection in a Relational Database Management System", *Proc. of ACM-SIGMOD* 1979.

[7] Shannon, C.E., "The Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27, 1948.

[8] Wong, A.K.C., Chiu, K.Y., "Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data", *IEEE Tran. Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 6, November 1987.

[9] Khinchin, A.I., *Mathematical Foundations of Information Theory*, Dover, New York, 1957.

[10] Reiter, R., "Towards a Logical Reconstruction of Relational Database Theory", *On Conceptual Modelling: Perspectives from Artifical Intelligence, DataBase, and Programming Languages*, Brodie, M., Mylopoulos, J., and Schmidt, J. (eds.), Springer-Verlag, New York, 1984.

[11] Shum, C.D., Muntz, R., "An Information-Theoretic Study on Aggregate Responses", *Proc. 14th Int. Conf. VLDB*, California 1988.

[12] Muralikrishna, M., DeWitt, D.J., "Equi-Depth Multidimensional Histograms", *Proc. of ACM-SIGMOD* 1988.

[13] Vander Zanden, B.T., Taylor, H.M., Bitton, D., "Estimating Block Accesses when Attributes are Correlated", *Proc. 12th Int. Conf. VLDB*, Kyoto, 1986.