Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distributions of Column Values

Clifford A. Lynch
Division of Library Automation
Office of the President and Universitywide Services
University of California
Berkeley, CA 94720

Abstract

When column values in a large database follow highly skewed distributions (such as Zipf distributions, typically found in textual databases), query optimizers in current relational systems often fail to choose optimal query plans even for simple single-relation queries. The major cause of these optimization failures is incorrect predicate selectivity estimation; the likelihood and cost of such errors are quantified. A scheme for adding user-defined selectivity estimators to a relational DBMS is proposed. The paper defines a series of new selectivity estimation methods that work well with highly skewed value distributions and then compares them to currently used methods such as uniform approximation and histograms. Empirical data from a large bibliographic database is used throughout the analyses in this paper.

1. Introduction

While current query optimizers such as those in System R/DB2 [Selinger et al. 1979] or INGRES [RTI 1986, Stonebraker et al. 1976] use elaborate algorithms to determine the best execution plan for multirelation joins, their ability to optimize relatively simple single-relation queries is limited by their inability to estimate accurately predicate selectivity. For some applications, including textual and bibliographic databases, this limitation is a major problem since column values are distributed in an irregular fashion, and effective query optimization depends on the ability to correctly estimate selectivity of predicates.

Section 2 discusses models of highly skewed column value distributions, which are based on Zipf distributions. Actual data from a large bibliographic database is used to parameterize and validate the models presented. The third section quantifies the effects of incorrect selectivity estimation in large databases where column values follow these distributions and shows that these errors are unacceptable. Section 4 defines a mechanism for incorporating user-defined selectivity estimators into a relational database using an approach similar to that proposed for user-defined operators in previous research [Stonebraker 1986]. The final section proposes a series of new selectivity estimation methods that work well with highly skewed column value distributions on textual databases and compares the performance of these estimators with previous methods such as histograms and uniform approximation.

This paper uses SQL for a query language. To avoid complexities involved in indexing textual data [Lynch & Stonebraker 1988], it is assumed that the DBMS is augmented to permit column values to be sets as proposed in, for example, [Zaniolo 1983]. Zaniolo's notation, adapted for SQL, is used: braces surrounding a datatype in a table definition indicate that the column value is a set of that datatype, and the IN operator tests for membership in a set. This paper is concerned

only with sets of strings. It is assumed that the DBMS can build a B-tree index for elements of a set of strings by placing each element in the B-tree, and that it will use this B-tree secondary index to resolve set membership (IN operator) predicates. Selectivity estimation for a "standard" relational system with the IN operator is assumed to be a direct extension of the uniform-distribution-based selectivity estimation traditionally used for equality predicates. This is detailed in Section 3. While the paper considers IN-operator predicates, the results are equally applicable to standard equality predicates on columns with similarly distributed values.

For a table T, *CARD(T)* is the number of rows in the table. For an index I, *CARD(I)* is the total number of entries in the index. *UNIQUE(I)* is the number of unique key values in an index I. The *selectivity* of a predicate is defined to be the fraction of the tuples in a relation that the predicate matches. Thus, a predicate with selectivity S will select $S*CARD(T)$ rows from a table T.

The empirical data in this paper is taken from a large bibliographic database used by the MELVYL® online union catalog, a replacement for a traditional library card catalog that provides access to the holdings of the libraries of the University of California nine-campus system. The MELVYL catalog is described in detail in [DLA 1987, Lynch 1987] and the references there.

2. Modeling Bibliographic Databases: Zipf Distributions

An online library catalog is used as a concrete example in this paper. The central relation in such a database describes books and might be defined as:

```
CREATE TABLE BOOKS
(TITLE LONG VARCHAR, TITLE-KEYTERMS
    {VARCHAR},
SUBJECTS LONG VARCHAR, SUBJECT-KEYTERMS
    {VARCHAR}, other columns);
```

Here, TITLE-KEYTERMS and SUBJECT-KEYTERMS are sets of strings, the elements of which are (roughly speaking) the individual words comprising the book's title and the subject headings assigned to the book. (See [Lynch and Stonebraker 1988] for full details on keywords.) A typical online catalog user query would be "find all books with the word history in the title." This would translate to an SQL query like

```
SELECT * FROM BOOKS WHERE "HISTORY"
    IN TITLE-KEYTERMS;
```

The set-valued columns TITLE-KEYTERMS and SUBJECT-KEYTERMS are assumed to have secondary B-tree indices to support IN operator predicates as described above. The distribution of values in these indices is the major focus of interest. A simple, mathematically tractable, and reasonably accurate first-order model of keyterm selectivity in bibliographic databases is provided by a discrete distribution called the Zipf distribution [Fedorowicz 1981]. Many other authors, including [Knuth 1968, 1973, Christodoulakis 1984], also use the Zipf distribution more generally to model distributions of values that are highly nonuniform, without specific reference to bibliographic databases. Word frequency distributions for many textual databases are known to follow Zipf or Zipf-like distributions. The Zipf distribution with parameter n, Zipf(n), is a set of probabilities p_i, $1 \leq i \leq n$, where $p_i = 1/i\,H_n$. Here H_n is the nth harmonic number, $H_n = \sum_{k=1}^{n} 1/k$. The distribution is interpreted as the probability that a given keyterm appears in i rows of the database is p_i. The expected number of appearances of a keyterm is n/H_n.

The following (slightly rounded) values observed from the MELVYL catalog database provide real-world parameter values for the Zipf distribution model.

Rows in Books Table	4,000,000
Unique Subject Keyterms	300,000
Total Occurrences of Subject Keyterms	45,000,000
Unique Title Keyterms	1,000,000
Total Occurrences of Title Keyterms	20,000,000

While the Zipf distribution model is simple, analytically tractable, and does capture the highly skewed nature of keyterm distributions, it is far from perfect. The value for n in the Zipf distribution model for each of the keyterm indices can be determined by observing that CARD(I) = UNIQUE(I) * "average number of occurrences of a value in I", or CARD(I)/UNIQUE(I) = n/H_n. For subject keyterms $n/H_n = 150$, so n is 1143; for title keyterms $n/H_n = 20$, yielding $n = 105$. It is clear from these figures that, as is common with Zipf distributions used to model bibliographic databases, the match between the theoretical distribution and the empirical data is not close for the most frequently occurring terms. The most common terms in the actual title file, which contains about 950,000 keyterms, occur perhaps 70,000 times. (There are six terms occurring more than 70,000 times.) However, modeling this keyterm collection with Zipf(70,000) would imply much larger numbers of total keyterm occurrences than actually are found in the

database. In addition, the standard Zipf distribution tends to estimate incorrectly the number of infrequently occurring terms. A Zipf (1143) distribution would predict that $1{,}000{,}000/H_{1443} = 131{,}249$ terms would occur only once. In actuality, there are some 484,505 of these terms in the MELVYL catalog title keyterm file. More elaborate distributions can be used to obtain more precise models [Lynch 1987], but space considerations preclude a discussion here.

3. The Effects of Selectivity Estimation Failure on the Performance of Current Optimizers

In determining the evaluation plan for a single-relation conjunctive query involving several predicates, current systems perform a selectivity estimate on each predicate [Selinger et al. 1979]. This selectivity estimate provides the approximate number of tuples from the target relation that satisfies the predicate. Assuming that secondary indices exist for all predicates involved in the query (which will be typical in a bibliographic retrieval environment), the query planner will choose the predicate it believes to be most selective (i.e., satisfied by the smallest number of tuples), use the index for that predicate to obtain the TIDs of the tuples satisfying the predicate, and then read these tuples and verify that they satisfy the remaining predicates in the conjunctive query. Those tuples satisfying all predicates comprise the result from the query. Consider the following two queries:

```
SELECT * FROM BOOKS WHERE
    term1 IN TITLE-KEYTERMS AND term2
    IN SUBJECT-KEYTERMS;
```

and

```
SELECT * FROM BOOKS WHERE
    term1 IN TITLE-KEYTERMS AND term2
    IN TITLE-KEYTERMS;
```

A current query planner will typically estimate selectivity for an equality predicate by assuming a uniform distribution of column values when an index is available, or statistics on the number of unique values appearing in a table for a given column are maintained. Assuming a column C is indexed by an index I, selectivity of a predicate such as $(C = value)$ will be estimated by $1/UNIQUE(I)$. For set-valued relations, where there can be more than one entry in the index for a single column value appearing in the relation, this formula must be generalized to estimate selectivity of the IN operator to $CARD(I)/CARD(T) * UNIQUE(I)$. This uniform-distribution-based estimate gives an expected selectivity of 1/200,000 for title keyterms and 3/80,000 for subject keyterms. The key point about uniform distribution selectivity estimation is that selectivity is estimated independently of the constant that appears in the predicate.

In the first query, System R or INGRES will always choose the predicate involving the title keyterm since the selectivity of values in the title keyterm index is much smaller than the selectivity of values in the subject keyterm index, and will use the title predicate to select rows to read and examine. This may or may not be a good decision, depending on the values of *term1* and *term2*; in many cases a randomly selected subject keyterm will be more selective than a randomly selected title keyterm. Specifically, if the subject keyterms are Zipf(m) and the title keyterms are Zipf(n), the probability that using the title predicate to select the access path is *correct* is $= 1 + \frac{H_n^{(2)}}{H_m H_n} - \frac{H_n}{2H_m}$. where $H_n^{(\theta)} = \sum_{i=1}^{n} 1/i^{\theta}$, the nth generalized harmonic number. (Details of this and other computations are omitted here in the interest of space; see [Lynch 1987] for full details.)

For $m = 1143$ and $n = 105$, this probability is .6768; thus, the selected plan is correct about two out of three times.

In the second example query, the optimizer cannot differentiate between the selectivity of the first and second predicates involving title keyterms; it will arbitrarily select one. The likelihood that it will choose the more selective predicate is somewhat better than .5. (It is better than .5 because the two terms may have equal selectivity.) Assuming the distribution of keyterms modeled by Zipf(n), a similar computations shows that the *correct* choice will be made with probability $1/2 + H_n^{(2)}/2H_n^2$.

For $n = 105$ (the value for title keyterms), the probability that the correct plan is chosen is .5298. For $n = 1143$ (a query similar to the second example but involving subject keyterms rather than title keyterms), the probability that the planner selects the correct plan is .5142. The reason the probability is better than .5 is that some $1/H_n$ of the subject keyterms only occur once; no matter which predicate is selected for the access path, the selection will always be correct at least that often (13% of the time for $n = 1143$). As queries become more complex, the likelihood that the planner will choose the optimal plan drops off quickly. Similar computations show that for $n = 105$, a five-term query only will be evaluated optimally with probability about .27.

Thus, the planner often will select the wrong plan. This erroneous choice may not be disastrous if the selected plan is only slightly suboptimal. If the selected plan requires an order of magnitude more I/O than the optimal plan, however, the error is quite serious. To measure the effects of bad plan selection due to incorrect selectivity estimation, the average number of I/O operations required for queries in the form

SELECT * FROM BOOKS WHERE *term1* IN TITLE-KEYTERMS AND *term2* IN SUBJECT-KEYTERMS;

will be computed over all pairs (*term1*, *term2*) where *term1* is a title keyterm and *term2* is a subject keyterm. Query plans will be selected both by using the System R/DB2 algorithm and under the assumption that the DBMS has perfect knowledge of predicate selectivity (and, thus, always selects the optimal plan from the set of plans considered by System R/DB2 in this situation). We assume that the values of TITLE-KEYTERMS are Zipf(n) and that those of SUBJECT-KEYTERMS are Zipf(m).

The average number of I/O operations required for plans selected by the System R/DB2 planner is simply $\sum_{i=1}^{n} i/i\,H_n = n/H_n$ since when a term that occurs i times is used as the access path, i reads are necessary to obtain the corresponding tuples. In contrast, if the planner always selects the optimized plan, then the average number of I/O operations per query is $= 1/H_n H_m \left((H_m + 2)n - (n+1)H_n\right)$. For the special case $m = n$, this simplifies to $(2n - H_n)/H_n$. The standard System R/DB2 plans require $n/H_n - (2n - H_n)/H_n^2 = ((n+1)H_n - 2n)/H_n^2$ additional I/O operations, on average.

For the first example query, the expected number of I/Os for a plan selected by System R/DB2 is 20.057. Always choosing the optimal plan would produce an average of 11.4054 I/Os. For the second example, System R/DB2 requires 20.0537 I/Os on average, again, while the average is 7.4691 for optimal plans. On a query analogous to the second example, but involving subject rather than title keyterms, System R/DB2 requires an average of 150.0184 I/Os, while the optimal plan only needs 39.2485 I/Os on average. The expected cost for the plan selected by DB2 diverges even more rapidly from the expected cost for the optimal plan for queries involving more than two terms. For example, with three conjuncts and n =105, the plan selected by DB2 still requires 20.0537 I/Os on average, while the optimal plan requires only 3.9873 I/Os. For three conjuncts and n = 200, DB2 plans average 34.0250 I/Os, while optimal plans average 5.5626 I/Os.

4. Selectivity Estimation

Previous query planners have defined estimators by built-in formulas. The System R/DB2 planner incorporates a number of hard-coded formulas [Selinger et al. 1979] to estimate the selectivity of various predicates having the form (*column relop value*). The definitions of user-supplied estimators for the user-defined access methods given in [Stonebraker 1986] also rely on formulas. In addition, with the exception of histograms in commercial INGRES, existing systems estimate selectivity for many types of predicates, such as (*column = value*), without reference to *value*. Other predicates, such as (*column* $\leq$ *value*), are estimated using only simple interpolation computations on the specified *value* (again, except when histograms are used). A selectivity estimate in current systems is tailored to a specific relation, a specific distribution of column values, and a specific value appearing in a predicate only through reference to a few basic parameters, such as the overall number of tuples in the relation or the number of unique values that appear in the index on a given column. Existing estimation formulas are quite general and consequently are not likely to be highly accurate for any specific predicate on a specific column of a specific table.

This paper proposes the incorporation of user-defined selectivity estimation procedures as a method of improving the accuracy of selectivity estimation. These procedures may be of arbitrary complexity and may include both sophisticated algorithms and internal lookups for exceptional cases using sizable data structures built into the estimator procedures. Clearly, user-defined estimation procedures must accompany the user-defined operators currently being incorporated in various extensible database systems such as POSTGRES [Stonebraker & Rowe 1985]. However, there are compelling justifications for permitting user-defined estimation procedures to be supplied for built-in operators as well:

- Because of the tremendous variation between value distributions in different columns of a relation or from relation to relation, estimation methods closely tailored to specific tables are required for precise estimation.

- User-defined estimators permit the incorporation of application-specific knowledge into the estimation (and hence the optimization) process. For example, it may be known that longer words tend

to occur less frequently or that a value distribution is relatively uniform when some group of exceptional values is removed. Examples of the use of application-specific knowledge in estimation will be developed in Section 5.

- Selectivity estimators are used to estimate the selectivity of predicates appearing in *queries*. The statistics that characterize the selectivity of such predicates may be different from the statistics that characterize a database. Given a large enough sampling of queries, it may be worthwhile to adjust the selectivity estimators to reflect the biases of users querying the database. With some bibliographic databases (particularly when users receive training before using them), users tend to avoid the most commonly occurring keyterms because they have little retrieval precision and produce unwieldy query results. Similarly, a substantial number of predicates of the form (*column = value*) actually fail to match any tuples in a public access information retrieval system (often because of a high incidence of keyboarding and spelling errors when queries are entered and a lack of understanding of indexing practice by retrieval system users). The effects of these phenomena can only be reflected in selectivity estimation by a user-defined estimator procedure.

4.1 Definition Registry and Choice of Predicate Selectivity Estimators

A predicate selectivity estimator is a procedure (either built-in or user-defined) that returns an integer giving the estimated number of tuples that will satisfy the predicate, and has as input parameters the predicate being estimated and statistical data maintained by the DBMS about the table referenced in the predicate and any associated indices.

A predicate selectivity estimator procedure is registered with the DBMS through the directive

DEFINE PREDICATE ESTIMATOR *procedure-name*

By definition, all predicate estimators take the same input parameters and return the same result parameters. Thus, there is no need to enumerate the parameter datatypes in the registration directive. A registered estimator is attached to a user-defined Boolean-valued binary operator, *operator-name*, through the ESTIMATOR parameter that is specified by extending the syntax of the DEFINE OPERATOR directive [Stonebraker 1986]. Such an estimator is called the *global default estimator* for the operator in question. Built-in Boolean-valued binary operators, such as { = , < , > , ≤ , ≥ } on various datatypes, are assumed to have global default estimators associated with them as well.

Local estimators can be defined that take precedence over the global default estimator when predicates involve specific tables or specific columns of specific tables. Such local estimators are defined by associating a registered estimator with an operator using one of the directives:

ESTIMATE *operator name* WITH
estimator-name ON *table-name*
ESTIMATE *operator-name* WITH
estimator-name ON *table-name.column-name*

The first directive indicates that the specified estimator is to be used to estimate selectivity of predicates in the form (*column relop value*) where *column* is any column in the specified table. The second form of the directive defines the estimator to apply to predicates of the form (*column-name relop value*). The query planner always uses the most specific estimator available to compute the selectivity of a predicate.

5. Estimators for Predicates of the Form (*value IN column*) in Bibliographic Retrieval Systems

This section describes the construction of an extremely accurate estimator for predicates of the form (*term* IN TITLE-KEYTERMS) as an example of the applicability of user-defined predicate selectivity estimators. The statistics of title keyterms in the MELVYL database are typical of many bibliographic databases, and a similar analysis and approach can be used to develop estimators for other predicate types such as (*term* IN SUBJECT-KEYTERMS). Section 5.1 discusses criteria used to measure the quality of estimators. Section 5.2 briefly reviews prior work in both parametric and nonparametric estimators for selectivity and discusses why these prior approaches fail to meet the needs of bibliographic retrieval systems. Section 5.3 describes methods for actually constructing estimators. Finally, Section 5.4 uses the criteria described in Section 5.1 to evaluate the performance of the estimators developed using the proposed methods against other approaches discussed in Section 5.2.

5.1 Evaluating the Quality of Estimators: Measures and Sample Sets

Let SEL(p) denote the true selectivity of the predicate p and let SEL_E (p) denote the estimated selectivity

computed by some estimator function E. The metric commonly used in statistical analysis to measure the quality of an estimator E relative to a set P of predicates is the *root mean square (RMS) error* (cf [Bendat & Piersol 1971], p. 170ff). RMS(E, P) is defined as

$$\sqrt{\frac{1}{\mathrm{CARD}(P)} \sum_{p \in P} (\mathrm{SEL}(p) - \mathrm{SEL}_E(p))^2}.$$

If the average is used as the estimator, then the RMS error is equal to the standard deviation. The normalized RMS error, which measures the *relative* error of an estimator, is written NRMS(E, P) and defined as

$$\sqrt{\frac{1}{\mathrm{CARD}(P)} \sum_{p \in P} \frac{(\mathrm{SEL}(p) - \mathrm{SEL}_E(p))^2}{\mathrm{SEL}(p)^2}}.$$

We will be concerned with two main sets of predicates: the set **D** of all predicates of the form (*value IN column*) where *value* appears in *column* for some row of the table, and a set **Q** of sample predicates taken from queries. Note that **Q** can contain predicates that do not match any tuples in the database, while all predicates in **D** match at least one tuple. The subset **Q'** of **Q** consisting of those predicates in **Q** that match at least one tuple in the database will also be used.

For the experimental results given here, the set **Q** contains 817,093 title keyterms that were extracted from a sample of 885,930 MELVYL catalog FIND commands (of which 326,511 referenced the title keyterm index) recorded from public access MELVYL catalog terminals during part of 1986. These 817,093 keyterms were extracted from a total of 1,017,306 title keyterms appearing in the FIND commands; the remaining title keyterms were discarded because they were stoplisted words not indexed in the MELVYL database, they contained invalid characters, or they actually were partial match specifications for title keyterms. In this sample of 817,093 title keyterms entered by users 80,151 distinct title keyterms appear. Some 30,955 of the 817,093 keyterms did not match any keyterm in the MELVYL database; 6467 user keyterms retrieved only a single book from the database. The least selective title keyterms actually used retrieved 79,535 books; these least selective keyterms were used 1363 times.

The set **D** consists of the 951,008 different title keyterms that appeared in the MELVYL database as of December 12, 1986. This set was actually derived from a larger set of 954,531 terms, some of which cannot appear in user queries (because they have been stoplisted but were partially indexed in the database prior to stoplisting, or because they contain characters that cannot be entered by the user in a title keyterm) and thus were eliminated from the sample. Of the keyterms in the set **D**, 484,505 keyterms are used only once. The two keyterms appearing most frequently in book titles occur 143,554 and 526,269 times, respectively. (There is some reason to argue that the most common keyterm, which appears 526,269 times, should not be considered since it appears to be a keyterm that should have been stoplisted and which appears rarely, if at all, in user queries.)

5.2 An Overview of Prior Work on Selectivity Estimation Methods

Parametric Methods

The first parametric approach to selectivity estimation was formalized in [Selinger et al. 1979] and assumed that the selectivity of index values was uniformly distributed and that each value selected approximately CARD(T)/UNIQUE(I) tuples. Inequality operators and range searching were carried out by further assuming that the distribution of values within the index was uniform between the high and low values appearing in the index, and then by using this distribution of values to estimate the number of values in the index satisfying the inequality or range predicate. Once the number of values was known, the selectivity of the range could be readily computed. The method used to estimate selectivity based on uniform distributions has an obvious extension when applied to IN predicates as discussed in Section 3.

More generalized parametric approaches were proposed by [Christodoulakis 1981] who suggested modeling selectivity using a series of (univariate) Pearson distributions which provides a range of distributions from uniform to normal (cf [Christodoulakis 1981], p. 62). The method assumes that for an index I with low value ℓ and high value h the distribution of selectivities is given by some distribution D on the interval $[\ell, h]$ which is either a discrete distribution (when values of I are integers) or a continuous distribution (when values of I are reals).

The difficulty in applying parametric distributions (other than the uniform distribution employed by Selinger, which has been shown to work poorly for bibliographic databases in Section 3) is that the indices are composed of character strings rather than integers or reals. A mapping from character strings to some interval of either integers or reals is required to apply nonuniform parametric estimation techniques.

The literature contains no consideration of appropriate mappings, although there has been some work done on appropriate distributions to describe bibliographic databases, which assumes that values appearing in an index have been mapped to an integer interval (typically through frequency ranking). Unfortunately, frequency ranking amounts to a map from strings to integers defined by an explicit table. The storage for such a table will be nearly as large as the index itself, rendering it useless for most estimation processes. Certainly, there is no map from keyterms in alphabetical order, for example, to rank frequency, which can be determined by simple interpolation or other nontabular methods.

Nonparametric Estimation

[Kooi 1980, Piatetsky-Shapiro, & Connell 1984, Christodoulakis 1981] propose variations on a nonparametric estimation method called histograms. While their work seems to have been written with the estimation of selectivity for numeric keys in mind, it is easily extended to character-valued keys. The basic idea of histogram estimation is that the range of key values is partitioned into a set of subranges $R_1, R_2, \ldots, R_n$. This can be done either algorithmically (for example, by dividing the range from ℓ to h into fixed-length segments in the case of a numeric index), or by a table of explicit subrange demarcation points determined according to some criteria. Each subrange R_i is then modeled using some distribution D_i. All three papers propose use of the uniform distribution, but an arbitrary distribution could be used.

The major difficulty with the currently proposed nonparametric methods for a bibliographic database is that they assume that ranges of keyterms in some natural ordering (such as collating sequence) have similar distributions, or at least that using a uniform approximation for the selectivity of moderate-sized groups of keyterms will be sufficient. The assumption of local "smoothness" in the distribution does not hold true in practice. when terms are listed in collating sequence, selectivity varies wildly from term to term.

5.3 Construction of an Accurate Selectivity Estimator for Title Keyterms

Construction of a selectivity estimation procedure for title keyterms is approached by piecewise approximation. First, frequently occurring keyterms are identified: their selectivity is estimated precisely by maintaining an actual list of these keyterms and the number of tuples they select within the estimator procedure. Subsection 5.3.1 gives both theoretical and experimental analyses of the trade-off between memory requirements and estimator accuracy.

For the remaining keyterms (which are not identified as very commonplace), the overall approach is to partition the set of possible keyterms into classes, based on some application-dependent criteria, which can be expected to correlate with selectivity, and then to calculate a uniform approximation for keyterm selectivity within each class. The estimator contains lists giving the average value to be used for each class. Once a class is assigned to an input keyterm, the selectivity estimate to be used is looked up in the list. The partitioning mechanisms that are considered in Subsection 5.3.2 are keyterm length (the number of characters in the keyterm) and analysis of digrams (adjacent letter pairs) that occur in keyterms. Keyterm length can be expected to correlate to selectivity in that long words are used less often than short words; there is a natural tendency in the development of language towards abbreviating long words that are frequently used, or supplanting them with shorter synonyms. Digram frequency is a well-known "signature" of romance languages and has been used for centuries in cryptanalysis [Kahn 1967]. It is reasonable to expect that a word containing an infrequently used digram will not be used often.

Analysis of selectivity for those keyterms for which selectivity must be heuristically estimated (since they are not explicitly listed in the estimator procedure as are frequently occurring keyterms) reveals an interesting and somewhat unexpected phenomenon. The selectivity statistics for keyterms *used in queries* are *radically* different than the statistics obtained by considering all keyterms that occur in the database. This application-specific knowledge is also incorporated into the estimation algorithms of Subsection 5.3.2 by assigning the selectivity estimate constant for each keyterm class based on query statistics rather than database statistics.

5.3.1 Selectivity Estimation for Frequently Occurring Keyterms

A relatively small number of common keyterms (typically about .5%) are responsible for much of the variance in keyterm selectivity. These keyterms and their selectivities can simply be stored in a list in the estimator procedure.

The following table demonstrates the relationship between allocating memory for lists of common terms

and their occurrence counts and the RMS error (relative to the set **D** of all predicates containing all title keyterms appearing in the database) when the average selectivity of the remaining terms not stored in memory is used as a selectivity estimator for these terms.

# terms in memory	Approx. storage (K bytes)	Selectivity cut-off for terms listed in memory	RMS error
50	≤ 1	26,905	307.8487
100	1	17,431	264.7333
200	2	12,241	219.2439
500	6	6,434	150.5060
1000	12	3,477	105.4980
2000	25	1,723	68.7911
5002	63	634	36.4704
10,013	127	269	20.8231
15,202	192	158	14.4888
20,111	255	110	11.2171
25,066	318	82	9.0797
50,343	639	31	4.4764
76,210	967	17	2.8640
96,678	1,227	12	2.2050

5.3.2 Selectivity Estimation for Infrequently Occurring Keyterms

If we assume that m keyterms are listed explicitly in the estimator along with their selectivity, several approaches can be taken for estimating the selectivity of input keyterms that do not appear in the list of m keyterms. The simplest approaches use uniform approximation for the remaining keyterms. The more elaborate (and hopefully more accurate) methods attempt to partition the remaining keyterms into keyterm classes so that the selectivity for members of each keyterm class can be well-approximated by a constant estimate over that keyterm class. This section defines a number of approaches and in some cases provides empirical data supporting the use of the approach. Subsection 5.4 actually compares the performance of the estimators defined here.

Uniform Estimators

A Uniform(m, S) estimator employs a list of the m terms that appear most frequently in the database and their selectivity; a selectivity estimate for an input keyterm that appears in the list is taken from the list. For keyterms that do not appear on the list, selectivity is estimated by averaging the selectivities of all the keyterms that appear in the set S but are not among the $m$keyterms explicitly listed in the estimator. Note that Selinger's uniform approximation is simply Uniform(0,**D**). Table 2 can thus be considered as giving the RMS error for the series of estimates Uniform(n, **D**) on the set **D**.

Length Estimators

A Length(m, S) estimator employs a list of the m terms that appear most frequently in the database and their selectivity, and a list of keyterm lengths and selectivity estimates for terms of that length. This second list is obtained by taking all terms in S that are not among the m terms explicitly listed, grouping these terms by length, and then computing the average selectivity for each group of terms of the same length.

There is a significant correlation between length and selectivity; in addition, the average selectivity of keyterms appearing in user queries of a given length is quite different from the same list when computed using terms appearing in the database. Table 1 gives average selectivity values by length for several different length estimators based both on database keyterms (the set **D**) and keyterms from user queries (**Q**).

Digram Estimators

A Digram(m, S) estimator again employs a list of the m most frequently occurring terms in the database and their selectivity. Digram estimators use two additional lists. The first list contains all two-letter pairs occurring in database keyterms and the frequency with which each two-letter pair occurs. The second list is of all digrams from terms in S (except for those in the list of the m most common terms) and a selectivity estimate developed by averaging the selectivity of all keyterms in S that have the specified digram as the least frequently occurring digram in the keyterm (based on the occurrence frequencies for digrams in the database given in the first additional list).

More precisely, for any digram δ, let FREQ(δ) = CARD($\{w \in \mathrm{D} | w$ contains the digram $\delta\}$) For any keyterm w, LFD(w), the least frequently occurring digram in w, is defined as LFD(w) = δ, where δ is a digram in w and FREQ(δ) $\leq$ FREQ(γ) for any other digram γ in w. For any digram δ, let $S(\delta)$ = AVG(SEL(w)), where this average is taken over the set $\{w | w \in S - \{$explicitly listed keyterms$\}$ and LFD(w) = $\delta\}$. For any input keyterm not on the list of m most common keyterms, S(LFD(w)) is used as a selectivity estimate.

There is a significant correlation between the value of a word's least common digram and that word's selec-

tivity. Again, there is a substantial variation between the estimated selectivity derived from the sets **Q** and **D**, as with the length estimators. Due to the size of the lists involved, however, they will not be reproduced here.

Minimum Variance Estimators

A Minvar(m, S) estimator combines the length and digram approaches. It employs a length estimator and a digram estimator for input keyterms not on the list of m common keyterms. However, it also includes lists of variances for the estimates provided by the length and digram estimators and selects the estimate with the lower variance for each input keyterm not explicitly listed with its selectivity in the list of the m most common keyterms.

Histogram Estimators

A Histogram(k) estimator is developed by choosing every kth keyterm (t_i) from a list of all keyterms in **D** in alphabetical order, and associating with each one of these chosen keyterms a selectivity estimate developed by averaging the selectivity of the $k-1$ keyterms in **D** immediately precceding the selected keyterm, plus the selectivity of the chosen keyterm t_i itself. If an input keyterm w falls into the sequence of chosen keyterms on the list as $t_n < w \leq t_{n+1}$, then the estimated selectivity value for t_{n+1} is returned by the estimator.

Based on the discussion in 5.2, these estimators should not be expected to perform well in the bibliographic retrieval environment. However, since they are used by INGRES as a means of improving on the selectivity estimates of System R/DB2, their performance will be examined in comparison to the other estimators defined in this section.

5.4 A Comparison of Estimator Performance

Table 2provides RMS and normalized RMS values for the various estimators when applied to three predicate sets: **D** (all keyterms in the database), **Q** (all keyterms in user queries), and **Q'** (all keyterms in user queries that match keyterms in the database). For those estimators that use variable amounts of memory (all those except for the Uniform(0, S) estimator included for comparative purposes since System R/DB2 uses it), two memory sizes were used: about 5,000 terms in memory (requiring about 68KB) and about 25,000 terms in memory (requiring about 318KB). While the estimators are listed as 5,000 and 25,000, actual values varied slightly because the cutoff for storage in memory was by selectivity value, and multiple keyterms existed with the desired selectivity values. Consequently, for example, all of the estimators except for the histogram estimators really used a value of 4,993 rather than 5,000, and of 24,847 rather than 25,000.

A number of conclusions can be drawn immediately from Table 2. The really crucial performance measures are RMS and normalized RMS on the set **Q**, since these figures give a very good sense of how well the various estimators will perform on actual user queries.

- Uniform approximation *never* works well.
- Histograms do not work well either. Moreover, providing more memory to maintain finer histograms does not help. The variation in keyterm selectivity from term to term when they are arranged in alphabetical order is so radical that in some cases a histogram with 25,000 partitions actually gives worse estimates than one with 5,000 partitions.
- Allocating enough space to list 25,000 rather than 5,000 common terms makes an enormous difference in estimator accuracy. If the memory used by the estimator is amortized over many users (as in a server model DBMS supporting a public access information retrieval system), it may be worthwhile to assign large amounts of memory to make critical estimators accurate.
- There is not a great deal of difference in the performance of Length, Digram, and Minvar estimators for a given amount of memory. On the basis of simplicity, Length estimators thus seem to be the best choice.

The poor performance of the digram estimators is somewhat suprising since the digram estimators intrinsically should contain more information than length estimators. It may be necessary to use n-grams with $n \geq 3$ to obtain more precise estimates, or to apply digram estimation in more elaborate ways.

To gain some sense of how the various estimators would perform on keyterms appearing in actual user queries, comparisons for the estimators were also run on several randomly selected subsets of **Q**. The results of these comparisons are similar to those in Table 2 and are not reproduced here due to space limitations. They show similar performance from Length, Digram, and Minvar estimators in most cases, suggesting that

the extra complexity of Digram and Minvar estimators are not warranted.

Table 2 also illustrates another interesting phenomenon. Estimators based on the sets **Q** and **Q'** generally display a smaller RMS error than those based on the set **D**. However, the *normalized* RMS error for estimators based on the set **D** are typically *lower* than those based on the sets **Q** and **Q'**. This lower normalized RMS error appears to be due to the fact that estimators based on **Q** or **Q'** use higher values than those based on **D**. As a result, they tend to overestimate selectivity by a larger factor for terms that actually occur only a few times. As a general rule, this type of estimation error is not terribly costly in terms of its effect on query plan cost; it results in a few unnecessary I/O operations rather than a disastrous error. In fact, the construction of estimators with lists of the most commonly occurring terms effectively guarantees that the planner will not make a totally disastrous error due to selectivity estimation error by choosing an unselective term as an access path. For any estimator E, the worst-case error for a two-predicate conjunctive query (i.e., "SELECT * ... WHERE X AND Y") is $max\{\text{SEL}(X) - \text{SEL}(Y)\}$, where $\text{SEL}_E(X) \leq \text{SEL}_E(Y)$.

For the MELVYL catalog title keyterm index, the worst case for the Uniform(0,**D**) estimator used by System R/DB2 is $\geq$ 100,000; for estimators such as Length(m, S) or Digram(m, S), the worst-case error can be determined from Table 3 (as a function of m). For example, for m = 5,000, the worst-case error is reduced to 633; for m = 25,000, the worst-case error is reduced to 81. This reduction in the worst-case error is advantageous, as it leads to more consistent performance from the DBMS.

6. Conclusions

This paper has shown that current query planners often fail to select optimal plans even for simple queries in an environment where column values have highly skewed distributions. These errors result from the inability of current selectivity estimation methods to cope with highly skewed selectivity distributions and are very costly. To solve this problem, a general mechanism for incorporating user-defined selectivity estimation into an RDBMS has been proposed. New estimation techniques suitable for use with textual or bibliographic databases having highly skewed attribute selectivity were defined and compared to existing methods. It seems clear that the estimation techniques described here are useful in a bibliographic or textual database environment. It would be interesting to compare these estimation methods to more standard selectivity estimation methods on text-oriented but nonbibliographic files, such as those found in business applications. If user-defined selectivity estimation is to be incorporated in extensible database systems, it will be necessary to develop tools and theory to assist in the creation of appropriate estimators for various types of databases. This paper takes a first step toward this goal by defining a set of performance measures for estimators, as well as enlarging the repertoire of available estimation methods.

The user-defined estimator scheme described here is actually a somewhat simplified version of [Lynch 1987], which also provides user-defined estimators for AND and OR operators between predicates, and extends the predicate selectivity estimator definition to accommodate LIKE predicates. While techniques similar to those presented here can be used to estimate selectivity of LIKE predicates fairly effectively, estimation for AND and OR operators requires radically different techniques and seems to be a much more difficult problem. The experiments described in [Lynch 1987] do suggest that the simple estimators for AND and OR selectivity presented in [Selinger et al. 1979] do not work well in a bibliographic retrieval environment.

The results presented in Table 2 also give rise to a speculation with far-reaching consequences. To date, all selectivity estimation approaches have worked with values from the database, including uniform approximation and histograms. This paper shows that selectivity statistics as computed even from very large samples of actual queries are radically different from those derived using "static" analysis of the database, and that the use of statistics *derived* from queries yields much better selectivity estimation for predicates that actually appear in queries. This suggests that research on selectivity estimation may have been emphasizing the wrong issue. Rather than concentrating on improving the precision of database statistics, perhaps more emphasis needs to be placed on classifying query loads into categories within which queries display reasonably consistent statistics, and on estimating based on selectivity statistics derived from measuring the behavior of actual queries from these query classes. There seems to be relatively little data concerning selectivity statistics from queries outside of the bibliographic and information retrieval environment. This area calls for further exploration.

Acknowledgements

The research described here was carried out as part of a Ph.D. thesis at the Computer Science Department of the University of California at Berkeley. I would like to thank Professor Michael Stonebraker, my thesis advisor, for his advice and guidance in this research. My thanks also to Nancy Gusack for assistance in the preparation of this paper.

Bibliography

[Bendat & Piersol 1971] Bendat, Julius S. and Piersol, Allan G. *Random Data: Analysis and Measurement Procedures* (New York, NY: Wiley-Interscience, 1971).

[Christodoulakis 1981] Christodoulakis, Stavros. "Estimating Selectivities in Data Bases," *Technical Report CSRG -136* (Toronto, Ontario: Computer Systems Research Group, University of Toronto, December 1981).

[DLA 1987] Division of Library Automation. *MELVYL Online Catalog Reference Manual* (Berkeley, CA: Division of Library Automation, University of California, 1987).

[Fedorowicz 1981] Fedorowicz, Jane. *Modeling an Automatic Bibliographic System: a Zipfian Approach*, Ph.D. thesis (Pittsburgh, PA: Graduate School of Industrial Administration, Carnegie-Mellon University, April 1981).

[Kahn 1967] Kahn, David. *The Codebreakers: The Story of Secret Writing* (New York, NY: Macmillan, 1967).

[Knuth 1968] Knuth, Donald E. *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (Reading, MA: Addison-Wesley Publishing Co., 1968).

[Knuth 1973] Knuth, Donald E. *The Art of Computer Programming, Volume 3: Sorting and Searching* (Reading, MA: Addison-Wesley Publishing Co., 1973).

[Kooi 1980] Kooi, Robert Philip. *The Optimization of Queries in Relational Databases*, Ph.D. thesis (Cleveland, OH: Department of Computer and Information Sciences, Case Western Reserve University, 1980).

[Lynch & Stonebraker 1988] Lynch, Clifford A. and Stonebraker, Michael. "Extended User-Defined Indexing with Application to Textual Databases," to appear in *Proceedings, 14th International Conference on Very Large Databases.*

[Lynch 1987] Lynch, Clifford A. "Extending Relational Database Management Systems for Information Retrieval Applications," Ph.D. thesis (Berkeley, CA: Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1987).

[Piatetsky-Shapiro & Connell 1984] Piatetsky-Shapiro, Gregory and Connell, Charles. "Accurate Estimation of the Number of Tuples Satisfying a Condition," in *Proceedings of the 1984 ACM SIGMOD Conference* (1984), pp. 256–276.

[RTI 1986] Relational Technology, Inc. *INGRES Reference Manual, Release 5.0* (Alameda, CA: Relational Technology, Inc., August 1986).

[Selinger et al. 1979] Selinger, Patricia Griffiths; Astrahan, M.M.; Chamberlin, D.D.; Lorie, R.A.; and Price, T.G. "Access Path Selection in a Relational Database Management System," in *Proceedings of the 1979 ACM SIGMOD Conference* (1979), pp. 23–34.

[Stonebraker et al. 1976] Stonebraker, Michael; Kreps, Peter; Wong, Eugene; and Held, Gerald. "The Design and Implementation of INGRES," in *The INGRES Papers: Anatomy of a Relational Database System*, Stonebraker, Michael (ed.) (Reading, MA: Addison-Wesley Publishing Co., 1986), pp. 5–45.

[Stonebraker 1986] Stonebraker, Michael. "The Inclusion of New Types in Relational Data Base Systems," in *Proceedings, 2nd International Conference on Data Base Engineering*, Los Angeles, CA February 1986.

[Stonebraker & Rowe 1985] Stonebraker, Michael and Rowe, Lawrence A. "The Design of POSTGRES," in *Proceedings of the 1985 ACM SIGMOD Conference*, (1985), pp. 340–355.

[Zaniolo 1983] Zaniolo, Carlos. "The Database Language GEM," *Proceedings, SIGMOD '83*, pp. 207–218.

Table 1

Selectivity estimate values used by various Length(n, S) estimators for input keyterms not appearing on the estimator's list of n most common keyterms.

Length of input keyterm	LEN(5000,D) estimate	LEN(25000,D) estimate	LEN(5000,Q) estimate	LEN(25000,Q) estimate
2	86.4308	24.0760	197.5307	45.5159
3	30.0237	10.4747	193.8650	31.2523
4	21.3743	8.1082	200.3063	23.8138
5	15.0218	6.3543	197.8719	23.0159
6	12.1657	5.5393	187.0229	23.2190
7	10.7595	5.1059	184.4071	21.5149
8	10.0285	4.8539	172.9779	20.4142
9	8.6586	4.5544	160.4237	18.5192
10	7.9595	4.2959	163.8453	19.0465
11	6.9992	4.0134	142.9688	19.2429
12	6.1356	3.7465	145.3403	17.7244
13	5.4916	3.4490	157.8668	19.1518
14	4.5296	3.1732	123.0083	16.8670
15	4.0559	3.0338	114.1361	16.1683
16	3.4247	2.7795	103.7550	19.3652
17	2.9238	2.5258	53.7097	15.7198
18	2.7111	2.3254	53.3232	11.6151
19	2.4987	2.2178	25.9162	10.6258
20	2.1246	2.0158	14.7368	11.1727

Table 2
Comparative Evaluation of Accuracy of Selectivity Estimators

	On D		On Q		On Q′	
	RMS	Norm. RMS	RMS	Norm. RMS	RMS	Norm. RMS
Uniform(0,D)	438.7117	19.6863	14970.8874	86.2833	15262.7879	87.8218
Uniform(5000,D)	36.5123	6.4435	143.4466	7.5934	15262.7879	7.5404
Uniform(5000,Q)	169.9704	139.5262	106.4853	41.2332	102.8396	23,6115
Digram(5000,D)	36.2739	6.9228	142.1616	7.6410	144.9182	7.4999
Digram(5000,Q)	157.9533	127.5859	101.1532	39.1720	97.8610	23.1684
Digram(5000,Q′)	177.5873	143.9888	102.1313	44.7559	97.9755	25.7701
Histogram(5000)	441.8301	99.1798	14555.9069	91.2785	14839.7162	92.9223
Length(5000,D)	36.2158	6.8400	141.8514	7.6875	144.5993	7.4995
Length(5000,Q)	154.4031	124.5423	105.7466	40.3043	102.3710	23.3572
Length(5000,Q′)	176.9654	143.2313	106.5406	45.6190	101.6222	26.3108
Minvar(5000,D)	36.2743	5.2452	143.1128	7.5430	145.8945	7.5239
Minvar(5000,Q)	132.1300	105.4864	102.7227	34.6216	10.7255	20.5920
Minvar((5000,Q′)	176.5009	142.7997	106.4962	45.5152	101.6222	26.3108
Uniform(25000,D)	9.1298	2.9155	11.4930	1.8240	11.6836	1.6350
Uniform(25000,Q)	18.8942	15.9204	9.4498	4.8585	8.6645	2.6064
Digram(25000,D)	9.0590	2.9643	11.3539	1.8468	11.5364	1.6264
Digram(25000,Q)	18.5824	15.4162	9,0767	4.6790	8.3512	2.6204
Digram(25000,Q′)	24.1957	20.5051	9,5115	6.2455	8.031	3.3179
Histogram(25000)	438.8197	51.2145	14873.4377	86.4492	15163.4348	87.9912
Length(25000,D)	9.0319	2.9652	11.2231	1.8789	11.3959	1.6185
Length(25000,Q)	17.4581	14.4929	9,3460	4.7534	8.6041	2.5938
Length(25000,Q′)	23.9356	20.3597	9.7343	6.3205	8.2685	3.3765
Minvar(25000,D)	9.0510	2.5612	11.4675	1.7987	11.6613	1.6331
Minvar(25000,Q)	17.4111	14.4424	9.3363	4.7343	8.6041	2.5938
Minvar(25000,Q′)	23.9300	20.2495	9.6987	6.2729	8.2602	3.3707