# Histogram-Based Solutions to Diverse Database Estimation Problems

**Yannis Ioannidis**[†]
**Viswanath Poosala**
Computer Sciences Department, University of Wisconsin, Madison, WI 53706
{yannis,poosala}@cs.wisc.edu

**Abstract**

*Many current database systems use some form of histograms to approximate the frequency distribution of values in the attributes of relations and based on them estimate some query result sizes and access plan costs. In this paper, we overview the line of research on histograms that we have followed at the Univ. of Wisconsin. Our goal has been to identify classes of histograms that combine three features in most realistic cases: (i) they produce estimates with small errors, (ii) they are inexpensive to construct, use, and maintain, and (iii) they can be used for many diverse estimation problems. Based on that goal, we present several results, which eventually point towards a class of histograms that are practical, close to optimal, and effective in estimating sizes of query results, frequency distributions of attribute values in query results, and even costs of accesses using secondary indices.*

## 1 Introduction

Database management systems (DBMSs) contain several modules that require estimates of the sizes of query results or of the costs of queries and query access plans. First, the query optimizer needs size estimates so that it can plug them into cost formulas to estimate the costs of access plans. These cost estimates are then used to determine which plan is expected to be the cheapest to execute the corresponding query. Second, a sophisticated user interface needs size and cost estimates to provide them as feedback to users before a query is actually executed, as a way to detect errors in the query or misconceptions about the database.

Both size and cost estimations are usually based on statistics that are maintained in the database catalogs. In many systems (e.g., DB2, Informix, Ingres, Sybase, Microsoft SQL Server), these statistics take the form of histograms, which represent approximations of the frequency distributions of values in attributes of the database relations. These approximations are then used to obtain the necessary estimates.

For the past few years, we have been working on identifying classes of histograms that are desirable for various estimation problems [IC93, Ioa93, IP95]. In this paper, we provide an overview of several earlier and some more recent results, which eventually point towards a class of histograms that are practical, close to optimal, and effective in many estimation problems.

[†]Partially supported by the National Science Foundation under Grants IRI-9113736 and IRI-9157368 (PYI Award), by Lockheed as part of an MDDS contract, and by grants from IBM, DEC, HP, AT&T, Informix, and Oracle

# 2 Definitions

## 2.1 Frequency Distributions of Values

Given an attribute of a relation, the *frequency distribution of its values* is a set of pairs indicating, for each value in the attribute's domain, the number of tuples in the relation where the value appears. As an example, consider the simple relation *OLYMPIAN* in Table 1. The frequency distribution of the various values in its Department attribute is shown in Table 2.

| **OLYMPIAN** | | |
|---|---|---|
| Name | Salary | Department |
| Apollo | 60K | Energy |
| Aphrodite | 60K | Domestic Affairs |
| Aris | 50K | Defense |
| Artemis | 60K | Energy |
| Athena | 70K | Education |
| Demeter | 60K | Agriculture |
| Ermis | 60K | Commerce |
| Hefestus | 50K | Energy |
| Hera | 90K | General Management |
| Hestia | 50K | Domestic Affairs |
| Poseidon | 80K | Defense |
| Pluto | 80K | Justice |
| Zeus | 100K | General Management |

Table 1: The OLYMPIAN relation

| Department | Frequency |
|---|---|
| General Management | 2 |
| Defense | 2 |
| Education | 1 |
| Domestic Affairs | 2 |
| Agriculture | 1 |
| Commerce | 1 |
| Justice | 1 |
| Energy | 3 |

Table 2: Frequency distribution of Department

One can generalize the above and discuss frequency distributions of combinations of multiple attributes. For example, the joint frequency distribution of the attributes Department and Salary would indicate the numbers of tuples in the relation where each pair of Department-Salary values occurs. Multi-attribute joint frequency distributions essentially capture the correlation between the values of the participating attributes. In principle, such distributions (or their approximations) are required to calculate the size of any query that involves multiple attributes from a single relation. In practice, however, DBMSs deal with frequency distributions of individual attributes only, because considering all possible combinations of attributes is very expensive. This essentially corresponds to what is known as the *attribute value independence assumption*, and although rarely true, it is adopted by all current DBMSs. To simplify presentation, with few necessary exceptions, our description below deals with single-attribute distributions only. Our results, however, hold for the general case.

## 2.2 Histograms

Below we define histograms in a way that is more general than is common in query optimization. Specifically, a *histogram* of some attribute is an approximation of the frequency distribution of its values obtained as follows: the (attribute value,frequency) pairs of the distribution are partitioned into *buckets*; the frequency of each value in a bucket is approximated by the average of the frequencies of all values in the bucket; and the set of values in a bucket is usually approximated in some compact fashion as well. Note that, although it is uncommon in database histograms, in principle, the ranges of attribute values grouped in two buckets could overlap. Also note that a histogram with a single bucket generates the same approximate frequency for all attribute values. Such a histogram is called *trivial* and corresponds to making the *uniform distribution assumption* over the entire attribute domain.

| Department | Histogram H1 | | Histogram H2 | | Histogram H3 | |
|---|---|---|---|---|---|---|
| | Frequency in Bucket | Approximate Frequency | Frequency in Bucket | Approximate Frequency | Frequency in Bucket | Approximate Frequency |
| Agriculture | 1 | 1.50 | 1 | 1.33 | 1 | 1.43 |
| Commerce | 1 | 1.50 | 1 | 1.33 | 1 | 1.43 |
| Defense | 2 | 1.50 | 2 | 1.33 | 2 | 1.43 |
| Domestic Affairs | 2 | 1.50 | 2 | 2.50 | 2 | 1.43 |
| Education | 1 | 1.75 | 1 | 1.33 | 1 | 1.43 |
| Energy | 3 | 1.75 | 3 | 2.50 | 3 | 3.00 |
| General Management | 2 | 1.75 | 2 | 1.33 | 2 | 1.43 |
| Justice | 1 | 1.75 | 1 | 1.33 | 1 | 1.43 |

Table 3: Three types of histograms for the Department attribute

Continuing on with the example of the *OLYMPIAN* relation, Table 3 presents three different histograms on the Department attribute, all with two buckets. For each histogram, it first shows which frequencies are grouped in the same bucket by enclosing them in the same shape (box or circle), and then shows the resulting approximate frequencies, i.e., the averages of all frequencies enclosed by identical shapes.

There are various classes of histograms that systems use or researchers have proposed for estimation. Most of the earlier prototypes, and still some of the commercial DBMSs, use trivial histograms, i.e., make the uniform distribution assumption [SAC+79]. That assumption, however, rarely holds in real data and estimates based on it usually have large errors [Chr84, IC91]. Excluding trivial ones, the histograms that are typically used by systems belong to the class of *equi-width* histograms, first discussed for database query optimization in Kooi's thesis [Koo80] (under the name *variable-count* histograms). In such a histogram, there is no overlap among the ranges of attribute values associated with its buckets, and the size of the range (or the number) of values in each bucket is the same, independent of the value frequencies. Since these histograms store much more information than trivial histograms (they typically have 10-20 buckets), their estimations are most often better. Histogram H1 above is equi-width, since the first bucket contains four values in the range A-D and the second bucket also contains four values in the range E-J.

Although we are not aware of any system that currently uses histograms in any other class than those mentioned above, several more advanced classes have been proposed and are worth discussing. In Kooi's thesis [Koo80], the reverse of equi-width histograms was also introduced (under the name *variable-range* histograms). In such a histogram, the sum of the frequencies of the attribute values associated with each bucket is the same, independent of the size of the range (or the number) of these values. Piatetsky-Shapiro and Connell gave this class of histograms the name that is currently used most often, *equi-depth* (or *equi-height*). Their extensive study showed that equi-depth histograms have a much lower worst-case and average error for a variety of selection queries than equi-width histograms [PSC84]. Muralikrishna and DeWitt [MD88] extended the above work for multidimensional histograms that are appropriate for multi-attribute selection queries. As mentioned above, our line of work has focused on identifying optimal histograms (those with the least error in their estimates) and has introduced serial and end-biased histograms, discussed in detail below.

## 2.3 Serial and End-biased Histograms

For many types of queries, an important role with respect to optimality is played by the class of *serial* histograms [IC93]. In those, the frequencies of the attribute values associated with each bucket are either all greater or all less than the frequencies of the attribute values associated with any other bucket. That is, the buckets of a serial histogram group frequencies that are close to each other with no interleaving. Histogram H1 in Table 3 is not serial as frequencies 1 and 3 appear in one bucket and frequency 2 appears in the other, while histograms H2 and H3 are.

An important subclass of serial histograms is that of *end-biased* histograms. In those, some number of the highest frequencies and some number of the lowest frequencies in an attribute are explicitly and accurately maintained in separate individual buckets, and the remaining (middle) frequencies are all approximated together in a single bucket. End-biased histograms are serial since their buckets group frequencies with no interleaving. Histogram H3 in Table 3 is end-biased, since it accurately maintains the frequency of the Energy Department, while averaging the frequencies of all others. One could define an intermediate class of histograms as well (subclass of serial and superclass of end-biased), where the middle frequencies are partitioned into multiple buckets. These histograms, however, have all the problems of serial histograms and not all of their advantages (both to be discussed later), and are therefore not interesting.

# 3 Histogram Effectiveness for Various Estimation Problems

In this section, we focus on four estimation problems. Each of the following subsections deals with one of these problems, defines a desirable notion of optimality with respect to estimation errors, and then describes our results regarding optimal or suboptimal but effective classes of histograms. All results assume that histograms maintain the precise set of values associated with each bucket. We first introduce some notation regarding a histogram with $\beta$ buckets:

$b_i$ The $i$-th bucket in the histogram, $1 \leq i \leq \beta$ (numbering is in no particular order).

$n_i$ The number of attribute values placed in bucket $b_i$.

$V_i$ The variance of the frequencies of the attribute values placed in bucket $b_i$.

## 3.1 Result Sizes of Arbitrary Equality Join and Selection Queries

Assume that for every attribute of every relation, a specific number of buckets has been prespecified for a histogram approximating its frequency distribution. Our goal is, for each attribute of interest of each relation, to obtain a single histogram and use that in all queries where the attribute participates. This histogram may not be optimal in every case, i.e., may not return the closest approximation to the result size of all queries applied on all possible databases, but should be optimal "on the average".

Working with this notion of optimality, our work [IC93, Ioa93, IP95] has shown the following for tree, function-free, equality join and selection queries:

- Among all histograms with $\beta$ buckets on some attribute, the one that minimizes

$$\sum_{i=1}^{\beta} n_i \, V_i \tag{3}$$

  is optimal.

- Optimal histograms belong to the class of serial histograms.

- Within the restricted class of end-biased histograms, again the one that minimizes (3) is optimal.

## 3.2 Frequency Distributions of Attribute Values in Results of Equality Join Queries

In addition to the size of a query result, it is often necessary to obtain good estimates for the overall frequency distribution of values in some of its attributes. This is useful in optimization of complex queries, because the distribution of an intermediate result affects the result size and cost of the subsequent operation. It is also useful

in data partitioning for parallel executions of joins, because a good estimate of the join result distribution helps taking into account not only the input relations skew but the result skew as well. By partitioning the input relations so that work is balanced among processors with respect to all skews, response time of a parallel join execution can be reduced.

Working with the same notion of optimality as in Section 3.1, our work has shown the following for single equality join queries:

- The same (serial) histograms that are optimal for query result size estimation (identified in Section 3.1) are also optimal for the estimation of the frequency distributions in any attribute of these query results.

Thus, a single histogram can achieve optimality on multiple fronts.

### 3.3 Result Sizes of Range Selection Queries

Unlike equality selection predicates, the result size of a range predicate depends on the frequencies of the attribute values as well as the distribution of the values in the attribute's domain. Based on our work, as well as on some results from numerical analysis [dB78], we believe that a formal characterization of optimal histograms is unlikely for this problem. Hence, we have focused mainly on the design and efficient construction of highly accurate, but not necessarily optimal, histograms. This work is performed in collaboration with researchers from IBM Almaden and so far has resulted in the following:

- Even if not optimal, the (serial) histograms that are optimal for the previous two estimation problems are rather effective in estimating the result sizes of range queries as well. Although one would expect that histograms grouping arbitrary attribute values in buckets would not perform well for range queries, the fact that serial histograms approximate frequencies very well makes them reasonably accurate for these queries as well.

- A new class of histograms that accurately maintain the high frequencies in the data is also very effective. These histograms are similar to end-biased histograms, but permit more than one bucket to group multiple different frequencies so that each bucket covers a contiguous range of attribute values. They perform particularly well for skewed distributions, which usually cause the highest errors in estimation.

For range selection queries, both types of histograms significantly improve upon the traditional equi-width and equi-depth histograms as well as other statistical approximation techniques (e.g., [JC85]). Which of the two is to be preferred is an issue that we are currently investigating experimentally.

### 3.4 I/O Costs of Relation Accesses through Secondary Indices

All the preceding estimation problems require histograms on frequency distributions of attribute values. As mentioned earlier, in their general form, some of these problems deal with multi-dimensional (multi-attribute) frequency distributions, which capture the *logical* correlation between the values of the participating attributes. There is nothing to prevent us, however, from making one of the dimensions to be the pages of a relation, thus capturing the *physical* correlation of the values in the remaining attribute(s) to these pages. For example, for the two-dimensional case, we can talk about frequency distributions that represent the number of tuples with a specific attribute value stored in a specific page, for all such value-page pairs. In other words, such distributions capture the level of clustering of the data on disk, and can be used (in accurate or approximate form) in many ways. One of their most interesting uses is in calculating the number of data pages accessed when scanning a relation or processing a selection using an unclustered index. This depends heavily on the level of data clustering on disk, as the latter (together with the buffer pool size) determines whether or not a page will be fetched more than once.

Given the fact that physical correlation distributions cannot be maintained accurately, we propose that they are approximated by histograms, as done for logical correlation distributions. General questions on histogram optimality and effectiveness are still part of our investigation. With respect to equality selection using a secondary index, however, one can show that cost calculation for such plans using physical correlation distributions is isomorphic to query result size calculation using logical correlation distributions. This observation indicates the potential of our approach.

# 4 Histogram Implementation

Based on the above analysis, the importance of serial histograms (and their subclass of end-biased histograms) for several estimation problems becomes evident. Therefore, we turn our attention to how such histograms can be implemented. In particular, there are four issues that we briefly discuss: how the relevant data is collected from the database; how the buckets of the histogram are identified; how the histogram is stored in the database; and how the histogram is used for estimation. The following discussion addresses only histograms of logical frequency distributions. The exact same arguments, however, apply to histograms of physical frequency distributions as well, which are useful for cost estimations (Section 3.4).

## 4.1 Data Collection

In principle, to construct a histogram on some attribute R.A, one has to first execute the query

**select** A, Count (*) **from** R **group by** A

to capture its precise frequency distribution. In fact, multiple such queries are usually optimized and executed together to construct multiple histograms, sharing the access to relation R. Processing the above query is very efficient if an index exists on A. Otherwise, it may become expensive, especially when the number of different values in A is high. Since statistics collection is an infrequent operation, this is usually acceptable. When not, however, sampling can be employed to obtain an approximation of the frequency distribution [PSC84].

## 4.2 Bucket Identification

Given a frequency distribution, the next step in constructing a histogram for it is identification of buckets. Identifying the optimal histogram among all serial ones takes exponential time in the number of buckets. On the other hand, identifying the optimal end-biased histogram takes only slightly over linear time in the number of buckets. Moreover, when constructing end-biased histograms where only high frequencies are picked for individual buckets, sampling can be used to combine data collection and bucket identification. This results in a particularly effective method with respect to both space and time. Something similar is done by DB2/MVS in order to identify the 10 highest frequencies in each attribute, which are maintained in its catalogs [Wan92]. This approach will not work when the distribution has relatively many high frequencies and few small ones, in which case low frequencies will be chosen for individual buckets, because there is no known efficient technique to identify the lowest frequencies in a distribution. Such distributions, however, are quite rare in practice (they are, in some sense, the reverse of Zipf distributions), so we believe that the above sampling-based bucket identification is rather effective.

## 4.3 Histogram Storage

The problem of storing a histogram (on an individual attribute) is essentially that of storing a two-column relation; one column for attribute values and one for frequencies. Unlike equi-width and equi-depth histograms, serial histograms are hard to manage because they group arbitrary attribute values into buckets. Since there is

usually no order-correlation between attribute values and their frequencies, storage of serial histograms essentially requires an index that will lead to the approximate frequency of every individual attribute value. If we generalize to histograms on combinations of attributes, then multi-dimensional indices become necessary, e.g., Grid-Files [NHS84], further increasing the complexity of histogram management.

On the other hand, end-biased histograms require very little storage and no index. One must only store the attribute values in the individual buckets and their corresponding frequencies. Since all the remaining attribute values belong in a single bucket, they do not have to be stored explicitly; only their average frequency is enough. When searching the histogram for the frequency of a value, if the value is not among those explicitly stored, it is assumed to be in the other bucket. The storage and retrieval efficiency gains compared to serial histograms are significant.

### 4.4 Histogram Usage

In principle, the frequency distribution of a single attribute may be viewed as a vector, that of a pair of attributes as a two-dimensional matrix, and so on. Based on that abstraction, we have shown that query result sizes, query result distributions, and all other quantities mentioned above can be obtained by multiplication of the appropriate matrices [Ioa93]. Since histograms are approximations of frequency distributions, they can also be thought of as vectors and matrices. Therefore, abstractly, estimation implies multiplying histograms in a linear algebra fashion. For general serial histograms, this essentially translates into joins, and is clearly impractical. For end-biased histograms, however, no real matrix multiplication algorithms or joins need to be invoked; more direct combinations of the simple structures used to hold the histograms are both sufficient and efficient.

## 5 Histogram Comparison

As shown above, the difference in construction costs between the serial and end-biased histograms is dramatic. Serial histograms are exponential in the number of buckets, whereas end-biased histograms are almost linear. Moreover, there is high complexity in the storage and usage of serial histograms. Finally, in several experiments, we have observed that most often the errors in the estimates based on end-biased histograms are not too far off from the corresponding (optimal) errors based on serial histograms. Examples of such comparisons are shown in Figures 1 and 2 [IP95]. These illustrate the relative accuracy's of serial, end-biased, and traditional histograms in estimating the result size of a query joining a relation with itself. In Figure 1, the estimation error is plotted as a function of the number of available buckets, while in Figure 2, it is plotted as a function of the skew in the join attribute (captured by the $z$ parameter of the Zipf distribution). Clearly, the serial and end-biased histograms are significantly better than the others. Moreover, the end-biased histograms perform almost as well as the serial ones. Thus, as a compromise between optimality and practicality, we suggest that the optimal end-biased histograms should be used in real systems.

## 6 Histogram Weaknesses

As a last note, we would like to point out that there are some cases where serial and end-biased histograms will either provide really poor estimates or become very large and unmanageable. First, by their very nature, histograms make the following assumption:

> Uniform Frequency Assumption: All attribute values in a bucket are assumed to have the same frequency.

Clearly, this assumption will not fare well and will cause large errors if most of the attribute values occur with vastly different frequencies. Second, especially serial histograms have been presented as maintaining the precise set of attribute values associated with each bucket. As mentioned earlier, although this approach usually

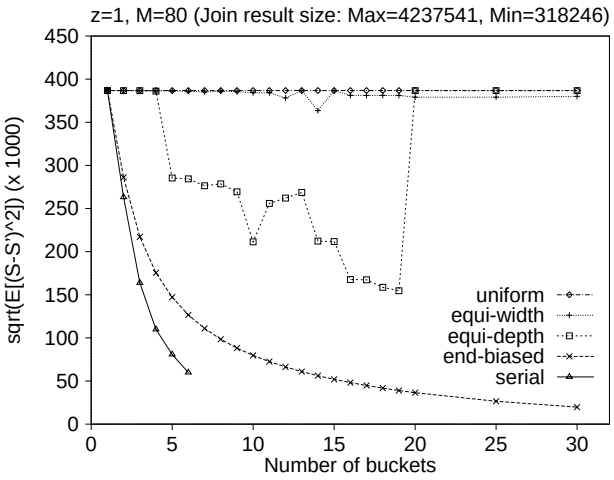


Figure 1: Error as a function of the number of buckets.

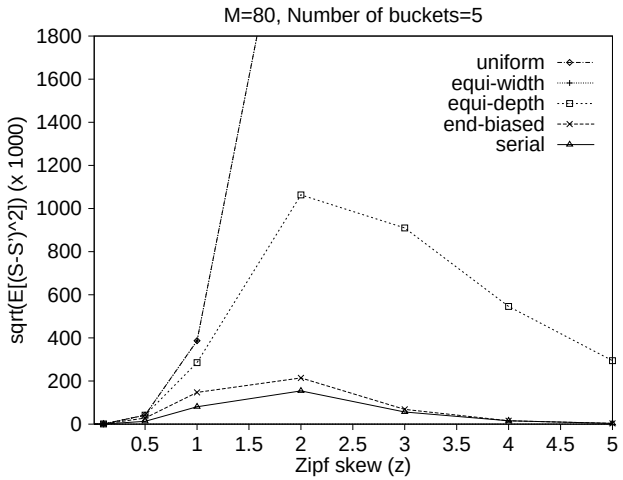


Figure 2: Error as a function of skew ($z$ parameter of Zipf).

results in good estimates, the required storage makes it impractical. Alternatively, one could make the following assumption:

> Continuous Values Assumption: Only the boundary attribute values of each bucket are explicitly stored, and all possible intermediate values are assumed to be present in the relation.

Clearly, this assumption will generate poor results if a large number of attribute values are distributed non-uniformly in the value domain (as is often the case for floating point attributes). In both these cases, none of the histograms can approximate the frequency distribution with high accuracy, unless there are as many buckets as frequencies. We expect that such cases are not very common in real data, so histogram-based techniques will indeed be effective in estimation.

# 7 Summary

We have presented an overview of our work on histograms and their effectiveness in several database estimation problems. We have identified the class of serial histograms as optimal in many cases, and the class of end-biased histograms as suboptimal but far more practical. Our overall conclusions is that, even when they are not optimal, these histograms are quite effective, and can therefore be used as universal tools for many database estimation problems.

Based on the above, we intend to use histograms in building a complete query size/cost estimator. As part of the foundations of this effort, we are currently working on several problems related to histograms that are not completely solved yet. First, we are conducting an experimental study to identify robust and effective histograms for estimating result sizes of range queries. Second, we are investigating optimality of various classes of histograms for estimating access cost to secondary indices. Finally, we are looking into the effect of database updates on optimal histograms and how to automatically detect the point where histogram recalculation becomes necessary.

# References

[Chr84] S. Christodoulakis. Implications of certain assumptions in database performance evaluation. *ACM TODS*, 9(2):163–186, June 1984.

[dB78] C. de Boor. *A Practical Guide to Splines*. Springer Verlag, New York, NY, 1978.

[IC91] Y. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *Proc. of the 1991 ACM-SIGMOD Conference on the Management of Data*, pages 268–277, Denver, CO, May 1991.

[IC93] Y. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM TODS*, 18(4):709–748, December 1993.

[Ioa93] Y. Ioannidis. Universality of serial histograms. In *Proc. 19th Int. VLDB Conference*, pages 256–267, Dublin, Ireland, August 1993.

[IP95] Y. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proc. of the 1995 ACM-SIGMOD Conference on the Management of Data*, pages 233–244, San Jose, CA, May 1995.

[JC85] R. Jain and I. Chlamtac. The $p^2$ algorithm for dynamic calculation of quantiles and histograms without sorting observations. *Communications of the ACM*, 28(10):1076–1085, October 1985.

[Koo80] R. P. Kooi. *The optimization of queries in relational databases*. PhD thesis, Case Western Reserve University, Sept 1980.

[MD88] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data*, pages 28–36, Chicago, IL, June 1988.

[NHS84] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71, March 1984.

[PSC84] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proc. 1984 ACM-SIGMOD Conference on the Management of Data*, pages 256–276, Boston, MA, June 1984.

[SAC+79] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the ACM SIGMOD Int. Symposium on Management of Data*, pages 23–34, Boston, MA, June 1979.

[Wan92] Y. Wang. Experience from a real life query optimizer. In *Proc. of the 1992 ACM-SIGMOD Conference on the Management of Data*, page 286, San Diego, CA, June 1992. Conference presentation.