International Journal of Database Theory and Application
Vol.9, No.10 (2016), pp.285-294
http://dx.doi.org/10.14257/ijdta.2016.9.10.24

Selectivity Estimation for Search Predicates over Set Valued Attributes

Alexander Korotkov[1] and Konstantin Kudryavtsev[2,*]

[1]Postgres Professional
7A Dmitriya Ulyanova, Moscow 117036 Russia
a.korotkov@postgrespro.ru
[2]National Research Nuclear University MEPhI
31, Kashirskoe highway, Moscow115409, Russia
kykudryavtsev@gmail.com

Abstract

At the moment many of modern relational databases support set valued attributes. Despite such attributes don't fit in classical relational theory, they expands the possibilities of data storage and manipulation. Search query on set valued attribute can be represented in specific search predicates which can be easily expressed in set-theoretic operations. Accurate enough selectivity estimation for search predicates on set valued attributes is essential for query optimizer in the same way as selectivity estimation for regular search predicates. This paper introduces a probabilistic model for estimating selectivity of search predicates on set valued attributes. This model uses frequencies of set elements occurrences as well as histogram of set values cardinality. Parameters of the model are estimated during preliminary analysis of database contents. The model was implemented for array types of DBMS PostgreSQL, which are implementation of ordered set valued attributes. Experimental verification of this implementation showed that highly accurate selectivity estimation is provided on the basis of the proposed model.

Keywords*: Selectivity estimation, Search predicates, Set valued attributes, DBMS Postgre SQL*

1. Introduction

The query optimizer is one of the most important components of any database management system (DBMS) that provides the declarative query language such as SQL, SPARQL, or any other. Query in a declarative language is a formal description of the data that user needs to retrieve from the database. The query optimizer translates the query into a specific sequence of data manipulation steps, called the query plan. Thus the same query may correspond to a very large set of possible plan sand the optimizer is supposed to select one "optimal" plan. The notion of optimality is usually defined as the minimum "cost" of the query plan, which in turn is an estimation of real resources required for its execution. A sufficiently accurate estimation of the "cost" of the query plan requires knowledge of a variety of parameters, including selectivity estimates for individual search predicated used in the query. Selectivity of the search predicate is defined to be inversely proportional to the number of tuples satisfying it. Thus, when number of tuples satisfying the search predicate is smallest, then such search predicate is the most selective. Selectivity of search predicates determine which of them would be evaluated using index. Also "cheapest" way to evaluate join depends on cardinalities of rowsets to be joined, which in turn depend on selectivity of search predicates. In a word, the whole query plan depends on selectivity estimations.

Search for strictly optimal query plan in terms of real server resources would require much more resources than execution of query in even very naive way. Such approach could make the whole optimization useless. Thus, it's imperative that planning of the query should take much lesser resources than its execution. Therefore, the query optimizer must be consists of with algorithms that have very low time complexity and based on limited statistics. Rather rough assumptions about the distribution of the data stored in the database are used. So it is often assumed that the distributions of values in different attributes of table are independent from each other. On the other hand, selection of a sufficiently good query plan usually does not require a highly precise measurement of its cost; it is usually sufficient to match the order of magnitude. But the big errors in the estimation may lead to the selection of an inefficient query plan.

Thus, we can distinguish task of selectivity estimation of individual search predicates for query optimizer. This task can be divided into two subtasks:

- collection of statistics that reflect the distribution of values for a specific attribute of the table;
- selectivity estimation of the search predicate on particular attribute based on the collected statistics.

One of the most widely used models to estimate the selectivity are histograms [14]. The simplest type is equi-width histogram. Equi-depth histograms can also be used [17]. In this type of histograms the interval between the minimum and the maximum value of the attribute is partitioned into sub intervals, so that the total frequency of values of an attribute at each interval is the same. [19-18] introduces a system for obtaining different histograms. Among histograms discussed in [19-18], the best efficiency is obtained when using MaxDiff (V, A).

Another way to estimate the selectivity is through the use of Random sampling [9,10,13,12,8]. This approach is based on the fact that a large amount of data can be introduced representatively using a small random sample. The advantage of this approach is its simplicity, because is does not imply data pre-processing and storage of statistical information. Additionally, the probability of accuracy of selectivity estimation can be calculated. The disadvantages are the complexity and high cost of obtaining representative samples, as well as the inability to re-use the results for other requests.

[5,21,6] introduce probabilistic counting to estimate the size of the projections, and [4,22] presents a standard statistical method of parametric estimation of selectivity. It is assumed that the data is described by the known laws of distribution with several parameters. Unfortunately, this assumption is rarely performed for the actual values from the database.

If the query contains a condition of selection on multiple attributes, the selectivity depends on the joint distribution of these attributes, *i.e.* the frequencies of all the combinations of attribute values. The main problem in the construction of histograms for the case of multi-attribute is to find correlations among the various attributes. To facilitate the assessment of selectivity most databases use the assumption of statistical independence of attribute values. With this assumption, histograms for individual attributes are used, and the joint probability is calculated as the product of the probabilities of the individual attributes. Real data rarely satisfies the independence condition and, therefore, assessment of selectivity is very inaccurate. [15,20] proposes original algorithms for constructing histograms for multidimensional data, *i.e.* histograms of the joint distribution of data from several attributes.

There are different models to assess the selectivity. Thus, [7] proposes to use Probabilistic relational models, which are actually an extension of graphical statistical models such as Bayesian Networks. These models represent the statistical dependencies between attributes within a table, and between attributes of tables related by foreign keys.

A lot of attention is focused on selectivity estimation of queries that operate on sets of data, such as strings, arrays, arrays of arrays, *etc.* [3] proposes an algorithm for estimating the selectivity for string predicates, which adapts to the characteristics of data and query and avoids underestimating the selectivity. [11] introduces a measure for the sets of weighted similarity and develops an algorithm for estimating the selectivity based on a priori constructed samples.

2. Proposed Model

Let R be some relation; and let A be its set values attribute. Let us denote $N = |R|$ and values of A as $\{a_1, a_2, \dots a_N\}$. Let us denote whole set of elements which are present in values of A as S.

$$S = \bigcup_{i=1}^{N} a_i \tag{1}$$

Assuming each of s_i is finite set and N is finite number, S is also a finite set. Let us denote cardinality of S as n and its elements as e_i: $S = \{e_1, e_2, \dots e_N\}$.

For the random value V from set valued attribute let us denote random numbers X_i which indicates if element e_i is present in V.

$$X_i = \begin{cases} 0, e_i \in V \\ 1, e_i \notin V \end{cases} \tag{2}$$

Probabilistic model should define joint distribution of $X_1, X_2, \dots X_n$. This distribution can be defined by the function $p(x_1, x_2, \dots x_n)$.

$$p(x_1, x_2, \dots x_n) = P(\bigwedge_{i=1}^{N} (X_i = x_i)) \tag{3}$$

Let us also denote $p_i = P(X_i = 1)$.

A. Basic model

Basic model assumes that all X_i are independent.

$$p_b(x_1, x_2, \dots x_n) = \prod_{i=1}^{N} p_i^{x_i} (1 - p_i)^{(1-x_i)} \tag{4}$$

Particular values of $p_1, p_2, \dots p_n$ can be estimated on production database by random sampling statistics. Thus $p_b(x_1, x_2, \dots x_n)$ can be also estimated by such statistics.

B. Extended model

Let us denote probability that V has cardinality of m in the basic model as I_m.

$$I_m = P(|V| = m) = \sum_{\substack{x_1 x_2, \dots x_n \\ x_1 + x_2 +, \dots + x_n = m}} p_b(x_1, x_2, \dots x_n) \tag{5}$$

Let us also denote probability of particular X_i values occurrence while cardinality of V is m in the basic model as $p_m(x_1, x_2, \dots x_n)$.

$$p_m(x_1, x_2, \dots x_n) = P\left(\bigwedge_{i=1}^{N} (X_i = x_i) | \sum_{i=1}^{N} X_i = m \right) = \frac{p_b(x_1, x_2, \dots x_n)}{I_m} \tag{6}$$

Extended model comes from the assumption that values of $p_m(x_1, x_2, \dots x_n)$ function are accurate enough while I_m values are not accurate. Then we can introduce $p_e(x_1, x_2, \dots x_n)$ function which is intended to be more accurate version of $p_b(x_1, x_2, \dots x_n)$.

$$p_e(x_1, x_2, \dots x_n) = H_m p_m(x_1, x_2, \dots x_n) = \frac{H_m}{I_m} p_b(x_1, x_2, \dots x_n) \tag{7}$$

$p_e(x_1, x_2, \dots x_n)$ used H_m to denote true probability of V cardinality equal to m while I_m is estimate based on assumption that X_i are independent. On the production database H_m can be estimated by random sampling statistics as well as $p_1, p_2, \dots p_n$.

Thus, we have two functions $p_b(x_1, x_2, \dots x_n)$ and $p_e(x_1, x_2, \dots x_n)$ to predict distribution of V – random value from set valued attribute A. $p_b(x_1, x_2, \dots x_n)$ describes basic model based on independent occurrence of set elements. $p_e(x_1, x_2, \dots x_n)$ describes extended model which uses distribution of set cardinalities as well.

3. Selectivity Estimation

If we denote search predicate as $Q(X_1, X_2, \dots X_n)$ then probability of random value V satisfy Q can be calculated as following.

$$P(Q(X_1, X_2, \dots X_n)) = \sum_{\substack{x_1 x_2, \dots x_n \\ Q(X_1, X_2, \dots X_n)}} p(x_1, x_2, \dots x_n) \tag{8}$$

We check proposed model for following three most typical search predicates. For given value V of set valued attribute and given constant $c = \{e_1, e_2, \dots e_l\}, l \le n$ these predicates are defined in the following way.

1) Intersect predicate

$$Q_{int}(X_1, X_2, \dots X_n) = (V \cap c \neq 0) = (\bigvee_{i=1}^{l} X_i = 1))$$

2) Subset predicate

$$Q_{sub}(X_1, X_2, \dots X_n) = (V \subseteq c) = (\bigwedge_{i=l+1}^{n} (X_i = 0))$$

3) Superset predicate

$$Q_{sup}(X_1, X_2, \dots X_n) = (V \supseteq c) = (\bigwedge_{i=1}^{n} (X_i = 1))$$

Let us denote probabilities of event M in basic and extended models as $P_a(M)$ and $P_b(M)$correspondingly. Estimates for these three search predicated could be calculated as following.

A. Intersect search predicate

$$P_b(V \cap c \neq 0) = \sum_{\substack{x_1 x_2, \dots x_n \\ x_1 + x_2 + \dots + x_l \ge 1}} p_b(x_1, x_2, \dots x_n) = 1 - \prod_{i=1}^{l} (1 - p_i) \tag{9}$$

$$P_e(V \cap c \neq 0) = \sum_{\substack{x_1 x_2, \dots x_n \\ x_1 + x_2 + \dots + x_l \ge 1}} p_e(x_1, x_2, \dots x_n) = \sum_{m=1}^{n} \frac{H_m}{I_m} \sum_{\substack{x_1 x_2, \dots x_n \\ x_1 + x_2 + \dots + x_n = m \\ x_1 + x_2 + \dots + x_l \ge 1}} p_b(x_1, x_2, \dots x_n) \tag{10}$$

B. Subset search predicate

$$P_b(V \subseteq c) = \sum_{x_{l+1} x_{l+2}, \dots x_n} p_b(1, \dots 1, x_{l+1}, \dots x_n) = \prod_{i=1}^{l} p_i \tag{11}$$

$$P_e(V \subseteq c) = \sum_{x_{l+1} x_{l+2}, \dots x_n} p_e(1, \dots 1, x_{l+1}, \dots x_n)$$
$$= \sum_{m=l}^{n} \frac{H_m}{I_m} \sum_{\substack{x_{l+1} x_{l+2}, \dots x_n \\ x_{l+1} + x_{l+2} + \dots + x_n = m - l}} p_b(x_1, x_2, \dots x_n) \tag{12}$$

C. Superset search predicate

$$P_b(V \supseteq c) = \sum_{x_{l+1}x_{l+2},\dots x_n} p_b(1,\dots 1, x_{l+1},\dots x_n) = \prod_{i=l+1}^{n} (1 - p_i) \tag{13}$$

$$P_e(V \supseteq c) = \sum_{\substack{x_{l+1}x_{l+2},\dots x_n \\ l}} p_e(x_1,\dots x_l, 0, \dots 0)$$
$$= \sum_{m=1} \frac{H_m}{I_m} \sum_{\substack{x_{l+1}x_{l+2},\dots x_n \\ x_1+x_2,\dots+x_l=m}} p_b(x_1, x_2, \dots x_l, 0, \dots 0) \tag{14}$$

Using dynamic programming all of the above estimates could be calculated in $O(n^2)$

4. Experimental Evaluation

If we denote search predicate as $Q(X_1, X_2, \dots X_n)$ then probability of random value V satisfy Q can be calculated as following.

A. Dataset

Precision of developed selectivity estimation methods was evaluated using real-life sample dataset. We've extracted tag arrays from delicious [1] dataset for this evaluation. Total number of arrays is 1138532, average length of array is 3:37. Distribution of array lengths is given on the Figure 1.

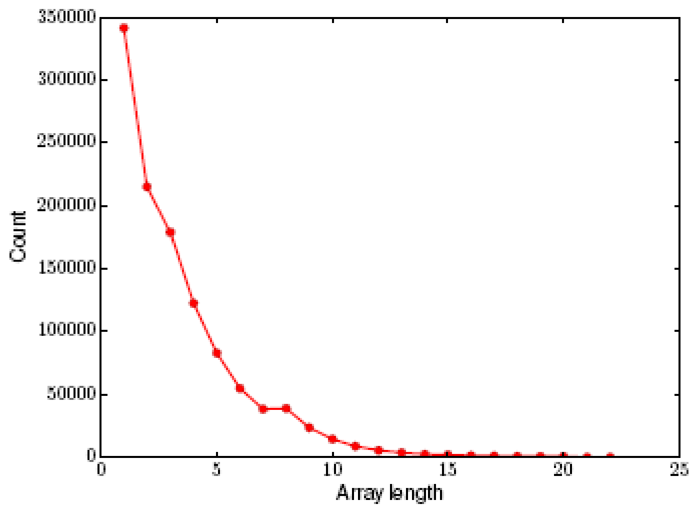

Figure 1. Distribution of Array Lengths

B. Compared methods

Following methods of selectivity estimation were compared using sample dataset.

1) Estimation produced by PostgreSQL query optimizer when data are in 3NF. In this case array contents was split into multiple rows of child table.

2) Estimation by basic model.

3) Estimation by extended model. The SQL queries used for estimation method #2 are presented in listings 1, 2 and 3.

Listing 1. Intersect search predicate

```
SELECT *
FROM parent_table p
```

```
WHERE EXISTS (
            SELECT 1
            FROM child_table c
            WHERE p.id = c. parent_id AND
            c. value IN (val_1 , val_2 , ... , val_n ))
```

Listing 2. Subset search predicate

```
SELECT *
FROM parent_table p
WHERE EXISTS (
            SELECT 1
            FROM child_table c
            WHERE p.id = c. parent_id AND
            c. value = val_1
            ) AND EXISTS (
                      SELECT 1
                      FROM child_table c
                      WHERE p.id = c. parent_id AND
                      c. value = val_2
                      ) AND
........
AND EXISTS (
            SELECT 1
            FROM child_table c
            WHERE p.id = c. parent_id AND
            c. value = val_n )
```

Listing 3. Superset search predicate

```
SELECT *
FROM parent_table p
WHERE NOT EXISTS (
            SELECT 1
            FROM child_table c
            WHERE p.id = c. parent_id AND
            c. value NOT IN (val_1 , val_2 , ... , val_n )
```

C. Results

Results of experimental evaluation are given on Table I and Figures 2, 3 and 4. We group test cases by search predicate and actual number of selected rows. For each group error was calculated as average difference between estimated and actual rows counts in orders of magnitude 15.

$$Error = avg\{|\log(N_{estimated} + 1) - \log(N_{actual} + 1)|\} \tag{15}$$

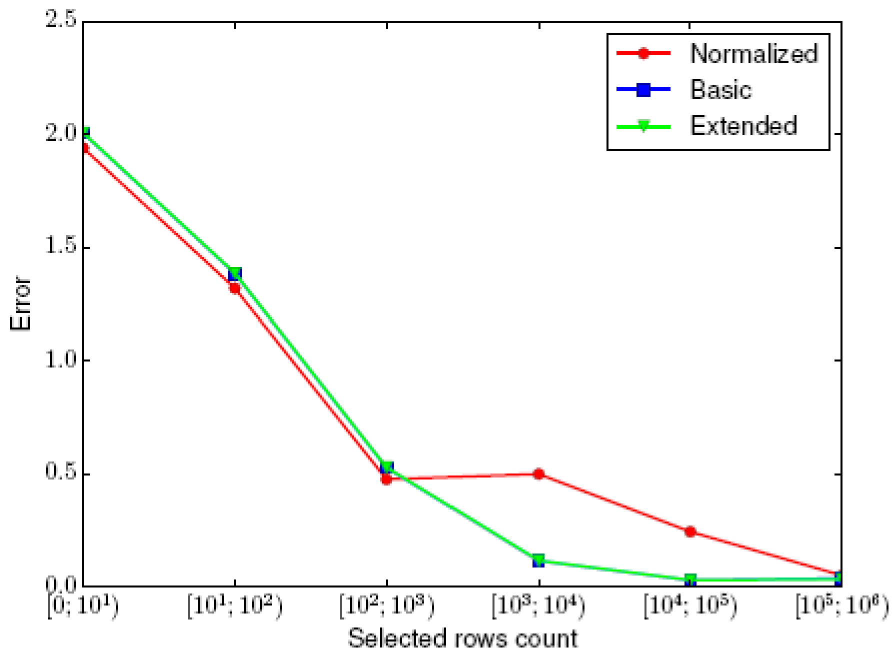

Figure 2. Intersect Predicate Estimation

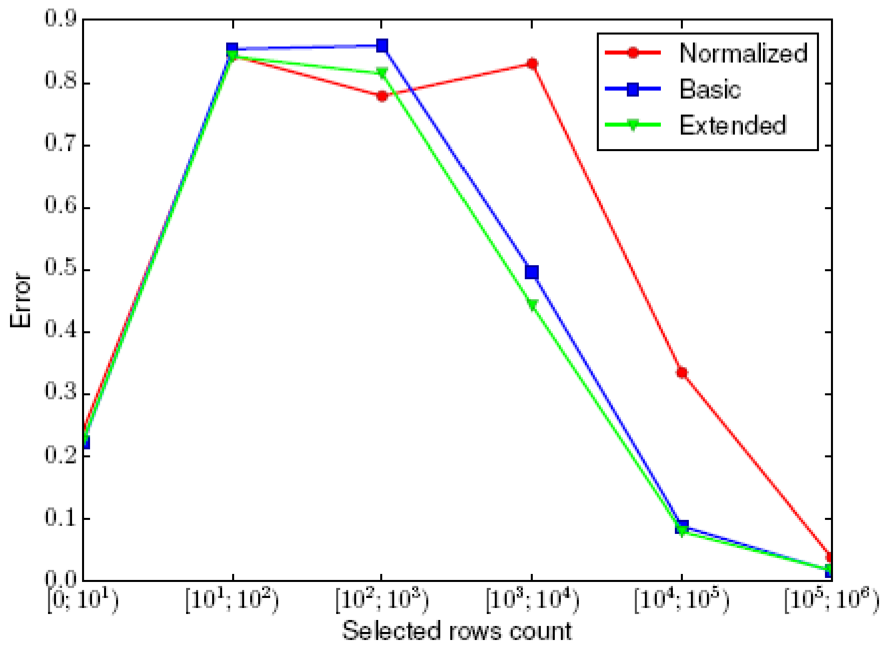

Figure 3. Subset Predicate Estimation

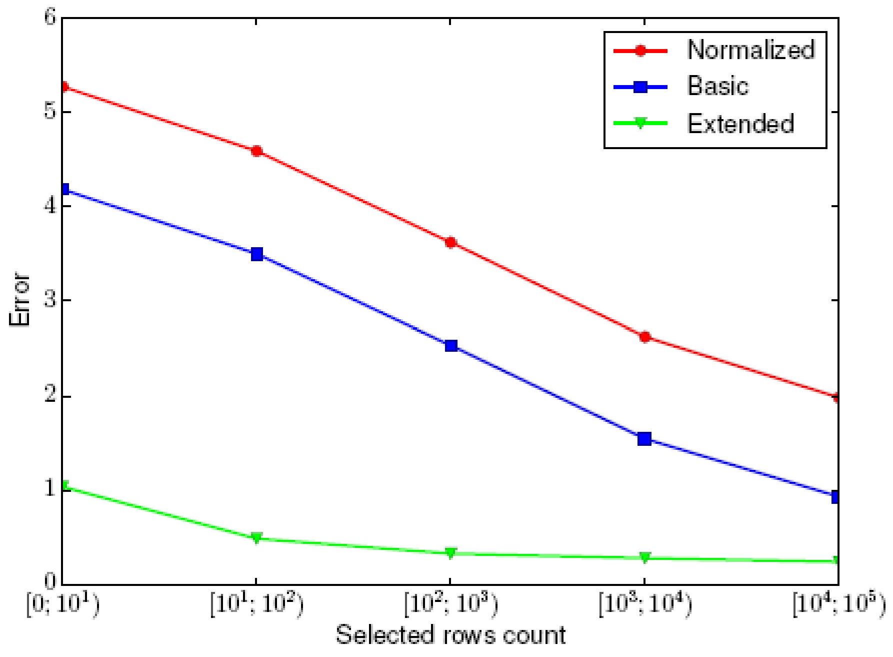

Figure 4. Superset Predicate Estimation

Table 1. Experimental Results

Pred	Row count	Queries	Normalized	Basic	Extended
&&	$[0; 10^1)$	54842	1.9394	2.0047	2.0068
&&	$[10^1; 10^2)$	74562	1.3194	1.3845	1.3868
&&	$[10^2; 10^3)$	100479	0.4761	0.5271	0.5294
&&	$[10^3; 10^4)$	177684	0.4994	0.1158	0.1166
&&	$[10^4; 10^5)$	356899	0.2442	0.0298	0.0302
&&	$[10^5; 10^6)$	235534	0.0505	0.0377	0.0343
@ >	$[0; 10^1)$	520391	0, 2401	0.2213	0.2238
@ >	$[10^1; 10^2)$	175318	0, 8434	0.8539	0.8418
@ >	$[10^2; 10^3)$	124908	0, 7783	0.8594	0.8142
@ >	$[10^3; 10^4)$	102055	0, 8305	0.4950	0.4418
@ >	$[10^4; 10^5)$	70445	0, 3343	0.0877	0.0789
@ >	$[10^5; 10^6)$	6883	0, 0377	0.0158	0.0168
< @	$[0; 10^1)$	95716	5.2740	4.1838	1.0347
< @	$[10^1; 10^2)$	135219	4, 5910	3.5009	0.4847
< @	$[10^2; 10^3)$	216160	3, 6221	2.5330	0.3265
< @	$[10^3; 10^4)$	441276	2, 6248	1.5446	0.2783
< @	$[10^4; 10^5)$	111629	1, 9817	0.9304	0.2396

We made following conclusions from experimental evaluation.

1) A high level of error for a small number of selected row by intersect search predicate is because frequencies were collected only for most frequent elements. Rare values used in these queries get the default estimation.

2) Positive effect of extended model is not great for intersect and subset search predicates. In order to save computational costs basic model was selected for these predicates in PostgreSQL[2].

3) Extended model produces reasonable accuracy for superset search predicate unlike other methods.

4) In general, the estimations that have been used in PostgreSQL [2], give sufficient accuracy for practical applications.

In general, the results obtained show that the proposed model enables assessment of selectivity with a fairly high degree of accuracy.

5. Conclusion

A probabilistic model is proposed for selectivity estimation of predicates over set-valued attributes. The basic model is based on the elements occurrence frequencies in the datasets. The extended model also uses distribution of sets cardinalities. Proposed model was experimentally evaluated on the real life dataset. Evaluation shows that extended model gives sufficient accuracy for practical applications. Basic model can save some computational costs in the cases when difference between basic and extended models isn't critical.

The combination of basic and extended models was implemented and committed to the open source DBMS PostgreSQL for arrays selectivity estimation [2].

Selectivity estimation which takes care about per-element correlations is advised for further research.

References

[1] "Json dump of delicious bookmarks", http://infochimps.com/datasets/ delicious-bookmarks, **(2009)**.

[2] "Collect and use element-frequency statistics for arrays commit", http://git.postgresql.org/gitweb/?p=postgresql.git;a=commitdiff;h=0e5e167aaea4ceb355a6e20eec96c4f7d05527ab;hp=34c978442c55dd13a3a8c6b90fd4380dad02f3da, **(2012)**.

[3] S. Chaudhuri, V. Ganti, and L. Gravano, "Selectivity estimation for string predicates: Overcoming the underestimation problem", In Data Engineering, 2004. Proceedings. 20th International Conference on IEEE, **(2004)**, pp. 227-238.

[4] S. Christodoulakis, "Estimating block transfers and join sizes", In ACM SIGMOD Record, ACM, vol. 13, **(1983)**, pp. 40-54.

[5] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications", Journal of computer and system sciences, vol. 31, no. 2, **(1985)**, pp. 182-209.

[6] D. Gelebne and E. Gardy, "The size of projections of relations satisfying a functional dependency", In VLDB, Citeseer, **(1982)**, pp. 325-333.

[7] L. Getoor, B. Taskar, and D. Koller, "Selectivity estimation using probabilistic models", In ACM SIGMOD Record, ACM, vol. 30, **(2001)**, pp. 461-472.

[8] P. B. Gibbons and Y. Matias, "New sampling-based summary statistics for improving approximate query answers", In ACM SIGMOD Record, ACM, vol. 27, **(1998)**, pp. 331-342.

[9] P. J. Haas and A. N. Swami, "Sequential sampling procedures for query size estimation", ACM, vol. 21, **(1992)**.

[10] P. J. Haas and A. N. Swami, "Sampling-based selectivity estimation for joins using augmented frequent value statistics", In Data Engineering, 1995. Proceedings of the Eleventh International Conference on IEEE, **(1995)**, pp. 522-531.

[11] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava, "Hashed samples: selectivity estimators for set similarity selection queries", Proceedings of the VLDB Endowment, vol. 1, no. 1, **(2008)**, pp. 201-212.

[12] R. J. Lipton and J. F. Naughton, "Query size estimation by adaptive sampling", In Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, ACM, **(1990)**, pp. 40-46.

[13] R. J. Lipton, J. F. Naughton, and D. A. Schneider, "Practical selectivity estimation through adaptive sampling", ACM, vol. 19, **(1990)**.

[14] Y. Matias, J. S. Vitter, and M. Wang, "Wavelet-based histograms for selectivity estimation", In ACM SIGMoD Record, ACM, vol. 27, **(1998)**, pp. 448-459.

[15] M. Muralikrishna and D. J. DeWitt, "Equi-depth histograms for estimating selectivity factors for multi-dimensional queries", In Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, **(1988)**, pp. 28-36.

[16] G. O¨ zsoyogˇlu, Z. O¨ zsoyogˇlu, and V. Matos, "Extending relational algebra and relational calculus with set-valued attributes and aggregate functions", ACM Transactions on Database Systems (TODS), vol. 12, no. 4, **(1987)**, pp. 566-592.

[17] G. P. Shapiro and C. Connell, "Accurate estimation of the number of tuples satisfying a condition", In ACM SIGMOD Record, ACM, vol. 14, **(1984)**, pp. 256-276.

[18] V. Poosala, "Histogram-based estimation techniques in database systems", **(1997)**.
[19] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates", In ACM SIGMOD Record, ACM, vol. 25, **(1996)**, pp. 294-305.
[20] V. Poosala and Y. E. Ioannidis, "Selectivity estimation without the attribute value independence assumption", In VLDB, vol. 97, **(1997)**, pp. 486-495.
[21] A. Shukla, P. Deshpande, J. F. Naughton, and K. Ramasamy, "Storage estimation for multidimensional aggregates in the presence of hierarchies", In VLDB, Citeseer, vol. 96, **(1996)**, pp. 522-531.
[22] W. Sun, Y. Ling, N. Rishe, and Y. Deng, "An instant and accurate size estimation method for joins and selections in a retrieval-intensive environment", In ACM SIGMOD Record, ACM, vol. 22, **(1993)**, pp. 79-88.