Cost Models for Approximate Query Evaluation Algorithms

Oxana DOLMATOVA, Anna YARYGINA and Boris NOVIKOV
Saint Petersburg State University
oxana.dolmatova@gmail.com, anya_safonova@mail.ru, borisnov@acm.org

Abstract. The optimization is essential for any high-performance querying system. Several optimization techniques were developed and successfully implemented for relational databases. However, these techniques should be re-examined and revised for distributed heterogeneous systems of information resources supporting diverse querying paradigms. We introduce cost models for approximate query evaluation in the context of generalized algebraic operations supporting both exact and similarity queries. The proposed cost models are suitable for approximate evaluation and trade-off between computational performance and the quality of results. We present a rationale for our approach and elaborate our cost model for key operations and algorithms.

Keywords. Cost models, approximate algorithms, query evaluation, heterogeneous systems, information resources

Introduction

The presence of high-level declarative query languages was considered as one of major strengths of database management systems since early 70-ies and became an inherent feature of the relational database model and its variations are implemented in the industrial SQL-based systems.

Providing highly expressive declarative means for query specification, these languages both require and enable sophisticated optimization. In contrast with low-level imperative object-oriented programming languages, which allow only moderate code improvements with local optimization, declarative languages depend dramatically on the quality of the optimizer.

Formally, the task of a query optimizer is to choose an algebraic expression of minimal execution cost among several equivalent expressions. In other words, any high-quality optimizer is inevitably a cost-based one and, hence, the cost model is one of the critical core components of the optimizer.

The abstract concept of cost may include several different measures of query execution performance. Usually the most important is execution time (either CPU or elapsed), amount of I/O, or, in a distributed mobile environment, the battery energy. However, the actual interpretation of cost is not essential for the optimization process.

There are many optimization techniques based on cost models, but in general almost all cost models were prepared for exact query evaluation algorithms.

In broad modern contexts, such as distributed heterogeneous systems, real-time business analytics and distributed mobile environments an approximate query evaluation becomes a must. Indeed, it does not make any sense to use exact query evaluation in uncertain context or for similarity-based queries, where "top k" approach is the best. Approximate query evaluation is ultimately needed to provide timeliness for business analytics or save energy in a mobile device.

As the query evaluation is approximate, the quality of the output may decrease. Our main objective is to provide a cost model capable to support trade-off between the cost of evaluation and the quality of results. Based on our cost model, the query optimizer can either provide best possible quality for a given cost, or minimize the cost providing at least required quality of the query output.

We start from naïve exact cost models and proceed with detailed analysis of approximate operations. We define models for a number of operations including "top k" operations for single and multiple arguments.

In this research we consider query processing in a heterogeneous environment of autonomous information resources. In this type of environments data statistics might be unavailable, hence simple cost models are needed for both of exact and approximate querying.

1. Cost Model Specification

In this paper we discuss query processing in a heterogeneous environment of autonomous information resources where all objects have scores which represent their relevance, similarity, quality and so on. Each operation takes as an input a stream of objects with scores and pushes into the output another stream of objects with their scores.

Further the specification of the cost model that supports the concept of operation result quality is presented and is followed by brief discussion of main operations and corresponding algorithms which are analyzed in this paper.

Operation cost model depends on 4 basic parameters: data size; data quality; data order; and the amount of resources needed for operation execution (operation cost).

On the one hand in traditional cost models the operation cost (amount of resources for its evaluation) can be estimated based on the size of arguments, order of objects in an argument. On the other hand, when we talk about approximate algorithms, the operation behavior depends on the quality of the arguments, the size of the arguments, order of objects in the arguments, and the amount of resources allocated for operation execution.

Thus, an operation cost model is defined as follows:

$opcost(resources, size_{in}, order_{in}) = (quality_{out}, size_{out}, order_{out})$, where

- *opcost* is the operation cost function, *opcost* connects the values of the input and output parameters;
- *resources* are resources allocated for the operation processing or the operation cost;
- $size_{in}$ is the operation input arguments size;
- $order_{in}$ is the operation input arguments order;
- $quality_{out}$ is the operation execution relative result quality, the relative quality of the result of the corresponding subquery;

- $size_{out}$ the result size;
- $order_{out}$ the result order;
- *resources*, $size_{in}$, $order_{in}$, $quality_{out}$, $size_{out}$, $order_{out}$ are objects with attributes which represent different measures of the corresponding parameter of the cost model and are discussed in details below.

Let us note that $size_{in}$ and $order_{in}$ are vectors of objects that describe all arguments of the operation.

The equation describes the cost model and binds all its variables and parameters. Thus, substituting in the equation the values of the variables, we can express and evaluate the remaining parameters of the cost model.

Apart from the basic cost model equations there are some restrictions on the cost models of the specific operations:

$$resources^{min} \leq \text{resources} \leq resources^{max}$$
$$quality^{min} \leq \text{quality} \leq quality^{max}$$

If the operation has $resources^{max}$ resources available, the result has the best possible quality with given arguments, that is relative $quality_{out}=quality^{max}$.

If the operation has $resources^{min}$ resources available, the result has the worst possible relative quality ($quality_{out}=quality^{min}$) with given arguments. In this case the algorithm spends the least possible amount of resources for operation execution.

Our cost models suggest the estimation of the unknown parameters. We assume that objects are not ordered in an input or output data if no information about order is available. The value of the quality and size parameters are estimated based on the history of the previous queries and collected statistics.

Further we discuss how to measure resources and data characteristics in our cost models.

1.1. Resources

Resources needed for a particular operation processing can be measured and evaluated in different ways. For example the operation processing cost can be estimated by its execution time. However, this approach restricts the proposed cost model to a specific query processing system.

In this research we evaluate the cost of operations in terms of the number of disk accesses, sequential accesses, executions of external operations, and so on. It is important to note that in this case, the cost model is more general. In this case the proposed cost model can be tuned depending on the specific characteristics of the query processing engine.

There are several characteristics and measure resources:

- *sa* the number sequential accesses to read objects from pipe;
- *dsa* the number of sequential disk accesses to read objects;
- *ma* the number of random memory accesses;
- *da* the number of random disk accesses;
- *pred* the number of predicate evaluations;
- *ha* the number of hash table accesses.

The operation cost or the amount of resources needed for its execution will be evaluated in terms of these measures. The cost of each parameter in terms of time (T_{sa},

T_{dsa}, T_{ma}, T_{da}, T_{pred}), which depends on the system configuration, will be estimated based on the experiments and available statistics. Thus the operation cost can be simply evaluated:

$$time = sa*T_{sa}+dsa*T_{dsa}+ma*T_{ma}+da*T_{da}+pred*T_{pred}+ha*T_{ha}$$

1.2. Size

The size of the input data, as well as the result sizes can be evaluated at least in two different ways: size, which data occupy in memory (*size*), and the number of objects, semantic units, which the operation processes (*cardinality*). These characteristics influence the behavior of algorithms that implement operations.

1.3. Quality

We distinguish absolute and relative quality which operations produce.

The absolute quality shows how the produced result suites to the user expectations. To measure the absolute quality we need to obtain the actual relevance scores which are usually not know in advance and sometimes at all.

The relative quality shows how the limited, approximate implementation of an operation changes the absolute quality of the result. In our cost models we operate with relative quality of the operations and queries. The most important property of the relative quality of a query is its monotony on the amount of resources allocated for its processing. The monotony of the relative quality depends on its definition and algorithms implementing approximate operations.

1.4. Order

The order of objects in the input data influences the operation execution cost. At this phase of work, the order of objects is defined as the order of their scores in a data set. The cost models depend on the fact whether the objects in the input data are naturally ordered according to their scores.

2. Operations and Algorithms

We consider three operations in this paper. Exact and approximate algorithms implementing them are discussed in this section and corresponding cost models are developed and analyzed further.

2.1. Top k

Top k operation takes as an input the stream of objects with scores and returns to the output the stream of best k objects according to their input scores. The naïve exact algorithm orders objects according to their scores if needed and returns k first of them. An approximate top k algorithm reads objects sequentially, while free time for operation execution is available, and pastes them into the sorted list of already processed objects. When the resources are over first k objects are set to be the result.

2.2. Fusion

Binary fusion operation processes two input streams of objects with scores and returns stream of received objects with newly generated aggregated scores. The naïve exact algorithm implementing fusion is based on nested loops.

2.3. Aggregation

The aggregation operation is the same as defined in [7]. The operation takes as an input two streams, where objects are naturally ordered according to their scores, and returns to the output best k objects according to the aggregated scores based on the input ones. The naïve exact algorithm is a simple combination of fusion and top k operation. However, effective and optimal algorithms for aggregation operation were developed in [7]: FA (Fagin's Algorithm) and NRA (No Random Access Algorithm). Because the input streams are coming from two independent sources and random accesses in the first algorithm can be expensive and sometimes impossible, they have been removed from the implementation of algorithms and replaced with sorted accesses.

3. Cost Models for Exact Algorithms

Let's describe a basic cost models for different operations in case when the relative quality of the arguments and the result of the operation as the best possible and is not regulated from the outside.

Actually cost models depend on algorithms implementing this operations rather than operations themselves. Here we describe cost models for natural exact operation algorithms discussed in section 3.

All restrictions will be defined based on the cost model described by the equation:
$opcost(resources, size_{in}, order_{in}) = (quality_{out}, size_{out}, order_{out})$.

3.1. Top k

For exact top k operation the cost model can be defined as follows:

$cardinality_{out} = min\{k, cardinality_{in}\}$
$order_{out} = true$
$cardinality_{in} \geq cardinality_{out}$
$size_{in} \geq size_{out}$
if $order_{in} = false$ then
    $sa = cardinality_{in}$
    $ma = C * cardinality_{in} * ln(k)$
if $order_{in} = true$ then
    $sa = cardinality_{out}$

3.2. Fusion

Here we consider the binary fusion operation. In this case $size_{in}, order_{in}$ represent two-dimensional vectors, since the fusion operation takes two arguments. For fusion operation (based on nested loop algorithm) we have the following cost model:

$sa=\sum cardinality_{in}$
$dsa=\sum cardinality_{in}$
$da=\prod cardinality_{in}$
$cardinality_{out} \leq \sum cardinality_{in}$

3.3. Aggregation

First of all we describe the intermediary parameters which can help us to construct formulas:

- t – estimated size of a table, which shows all objects with already read scores;
- ns – estimated number of scores needed to fill the table of size t;
- ks – estimated number of scores needed to fill the table of size k.

We assume independence of scores and uniform distribution and use expectation for evaluate these parameters.

$order_{in}=true$
$cardinality_{out}=min\{k, cardinality_{in}\}$
$order_{out}=true$
$cardinality_{in} \geq cardinality_{out}$
$size_{in} \geq size_{out}$

3.3.1. FA

$time = cardinality_{in}T_{sa}+(t\log t+4t)T_{ma}+tT_{ha}$

Summands include sorted access, sort, input/output and calculation of aggregate score as well as search for a place to insert a score just obtained from stream respectively.

After evaluation and substitution of intermediary parameters we obtain the following:

$time=((3/4\ln(cardinality_{in})+3/2)T_{ma}+3/4T_{ha})k+T_{ma}cardinality_{in}+1/4cardinality_{in}T_{ha}+1/4cardinality_{in}(\ln(cardinality_{in})-2)T_{ma}+cardinality_{in}T_{sa}$

3.3.2. NRA

$time=cardinality_{in}T_{sa}+((ns-ks)/2t\log t+2t(2+t))T_{ma}+tT_{ha}$

Summands include sorted access, sort, input/output and calculation of worst case score and best case score as well as search for a place to insert a score just obtained from stream respectively.

After evaluation and substitution of intermediary parameters we obtain the following (c_{in} denotes as $cardinality_{in}$ below):

$time=(45/32-9/64\ln(c_{in}))T_{ma}k^2+((3/4c_{in}+3-3/64c_{in}(\ln(c_{in})-2)+3/16c_{in}(3/4\ln(c_{in})-3/2)T_{ma}+3/4T_{ha})k+(1/2c_{in}(2+1/4c_{in})+3/64c_{in}^2(\ln(c_{in})-2)T_{ma}+c_{in}T_{sa}+1/4c_{in}T_{ha}$

4. Cost Models for Approximate Algorithms

4.1. Aggregation

Now we add the specific resource t0 – time operation execution. The problem is to estimate the result quality.

We create approximate model based on exact one. Let's *k'* denote the number of correct object returned by approximate algorithm. We express it in terms of *t0*. Hence we get number of scores which system can find for the given time. Then we consider ratio between *k'* and *k* which means the accuracy of result.

$quality_{out}=k'/k$

For FA we obtain the following: $k'=(t0-T_{ma}cardinality_{in}-1/4cardinality_{in}T_{ha}-1/4cardinality_{in}(\ln(cardinality_{in})-2)T_{ma}-cardinality_{in}T_{sa})/((3/4\ln(cardinality_{in})+3/2)T_{ma}+3/4T_{ha})$.

We will not give the inverse formula for NRA here because of the limits of paper size.

4.2. Top k

For approximate algorithm implementing top k operation we have:

$cardinality_{out}=\min\{k,cardinality_{in}\}$

$order_{out}=$true

$cardinality_{in}\geq cardinality_{out}$

$size_{in}\geq size_{out}$

Let us estimate $size_{min}<size_{out}<size_{max}$, $resources_{min}<resources_{out}<resources_{max}$, $quality_{out}^{min}<quality_{out}<quality_{out}^{max}$ of the operation processing result:

$cardinality_{min}=cardinality_{out}=cardinality_{max}=\min\{k,cardinality_{in}\}$,

$size_{min}=size_{out}=size_{max}$,

if $order_{in}$=true then

$sa_{min}=sa=sa_{max}=cardinality_{out}$

$quality_{out}^{min}=quality_{out}=quality_{out}^{max}=1$

if $order_{in}$=false then

$sa_{min}=cardinality_{out}$

$sa_{max}=cardinality_{in}$

$sa=S$

$ma_{min}=0$

$ma_{max}=C*cardinality_{in}*\ln(k)$

$ma=C*S*\ln(k)$

$quality_{out}^{max}=1$

$quality_{out}^{min}=0$

$quality_{out}=S/cardinality_{in}$ (in case when object scores are distributed uniform)

5. Related Work

The query optimization became both required and enabled since the advent of high-level declarative query languages, mostly in the context of the relational database model.

A brief overview of classical query optimization techniques can be found in [11]. The optimization techniques for distributed systems are summarized in [10]. An optimizer for distributed heterogeneous systems is proposed in [15].

The cost models proposed in this research are designed similar to those needed to traditional optimizers.

The optimization strategy based on algebraic equivalences between similarity based operations that serve as rewrite rules is outlined in [4]. Optimization rules based on similarity based algebraic framework properties and equivalence laws are also discussed in [1, 14, 5, 12].

It is important to mention that the lack of algebraic equivalences pushes the research to the development of optimization techniques based on performance/quality tradeoffs and approximate algorithms.

The approximate query evaluation techniques were considered in the context of very large data warehouses and mobile networks [2, 6, 3, 8]. The approximation is typically based on sampling, wavelets, or synopsis.

Handling of time constraints on complex SQL queries is proposed in [7]. The authors distinguish approximate (based on sampling) and partial (top k) query evaluation.

The quality and performance trade-offs for stream processing are discussed in [16, 9].

Cost models based on estimation of operation selectivity and cardinality are introduced in [1] for the selected set of operations: union, intersection, and difference; joins; merge; subtract; select; and map. Optimization rules based on the query tree reconstruction are derived from the previous analysis.

A sampling-based method to estimate the cardinality of rank-aware operators is developed for costing plans [12].

A novel multi-criteria query optimization techniques for performing query optimization in databases, such as multimedia and web databases, which rely on imperfect access mechanisms and top-k predicates are proposed in [13]. The size and quality factors are introduced into the cost model and optimization algorithms.

6. Conclusion

In this paper we presented several cost models for both exact and approximate query evaluation algorithms in distributed heterogeneous systems. The more resource we use, the more accurate the result is and vice versa. Our models convert these intuitive observations into clear and distinct formulas. It provides formalism for trade-off between quality and performance.

In the future we are going to conduct experiments in order to estimate the accuracy of our cost models.

References

[1] S. Adali, P. Bonatti, M. L. Sapino, and V. S. Subrahmanian, A multi-similarity algebra. In: *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD '98)*, New York, NY, USA. ACM, 1998, 402-413.

[2] B. Babcock, S. Chaudhuri, and G. Das, Dynamic sample selection for approximate query processing. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03),* New York, NY, USA. ACM, 2003, 539-550.

[3] S. Chaudhuri, G. Das, and V. Narasayya, Optimized stratified sampling for approximate query processing, *ACM Transactions on Database Systems* **32**(2) (2007), 9.

[4] S. Chaudhuri, R. Ramakrishnan, and G. Weikum, Integrating DB and IR technologies: what is the sound of one hand clapping? In: *Proceedings of Second Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA, January 4-7, 2005, 1-12.

[5] P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta, Imprecision and user preferences in multimedia queries: a generic algebraic approach. In: *Foundations of Information and Knowledge Systems, Proceedings of First International Symposium (FoIKS'2000)*, Burg, Germany, February 14-17, 2000. Lecture Notes in Computer Science **1762** (2000), Springer, Berlin, 50-71.

[6] C. Dell'Aquila, F. Di Tria, E. Lefons, and F. Tangorra, Accuracy estimation in approximate query processing. In: *Proceedings of the 14th WSEAS International Conference on Computers: Part of the 14th WSEAS CSCC Multi-Conference (ICCOMP'10)* **1,** Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS), 452-458.

[7] R. Fagin, A. Lotem, and M. Naor, Optimal aggregation algorithms for middleware, *Journal of Computer and System Sciences* **66**(4) (2003), 614-656.

[8] Y. Hu, S. Sundara, and J. Srinivasan, Supporting time-constrained SQL queries in Oracle. In: *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07)*, VLDB Endowment, 2007, 1207-1218.

[9] Ch. Jermaine, S. Arumugam, A. Pol, and A. Dobra, Scalable approximate query processing with the DBO engine, *ACM Transactions on Database Systems* **33**(4) (2008), 23:1-23:54.

[10] Q. Jiang, *A Framework for Supporting Quality of Service Requirements in a Data Stream Management System*. PhD thesis, Arlington, TX, USA, 2005.

[11] D. Kossmann, The state of the art in distributed query processing, *ACM Computer Survey* **32**(4) (2000), 422-469.

[12] D. Kossmann and K. Stocker, Iterative dynamic programming: a new class of query optimization algorithms, *ACM Transactions on Database Systems* **25**(1) (2000), 43-82.

[13] Ch. Li, Kevin Ch.-Ch. Chang, I. F. Ilyas, and S. Song, RankSQL: query algebra and optimization for relational top-k queries. In: F. Ozcan, editor, *SIGMOD Conference*. ACM, 2005, 131-142.

[14] L. P. Mahalingam and K. S. Candan, Multi-Criteria Query Optimization in the Presence of Result Size and Quality Tradeoffs, *Multimedia Tools and Applications Journal* **23**(3) (2004), 167-183.

[15] D. Montesi, A. Trombetta, and P. A. Dearnley, A similarity based relational algebra for web and multimedia data, *Information Processing and Management* **39**(2) (2003), 307-322.

[16] F. Pentaris and Y. Ioannidis, Query optimization in distributed networks of autonomous database systems. *ACM Transactions on Database Systems* **31**(2) (2006), 537-583.

[17] R. Zhang, N. Koudas, B. Ch. Ooi, D. Srivastava, and P. Zhou, Streaming multiple aggregations using phantoms, *The VLDB Journal* **19**(4) (2010), 557-583.