Top N optimization issues in MM databases

H.E. Blok
h.e.blok@cs.utwente.nl
Faculty of Computer Science, University of Twente
PO BOX 217, NL 7500 AE, Enschede, The Netherlands

1 Introduction

In multi media (MM) DBMSs the usual way of operation in case of a MM retrieval query is to compute some ranking based on statistics and distances in feature spaces. The MM objects are then sorted by descending relevance relative to the given query. Since users are limited in their capabilities of reviewing all objects in that ranked list only a reasonable top of say N objects is returned.

However, this can turn out to be a quite time consuming process. The first reason is that the number of objects (i.e. documents) in the DBMS is usually very large (10^6 or even more). From the information retrieval field it is known that usually half of all objects (e.g. documents) contains at least one query term; so, even considering only these objects might be very time consuming. The same may hold for MM in general.

The problem of top N MM query optimization is to find techniques to limit the set of objects taken into consideration during the ranking process as much and as soon as possible. My main research interest is to chart what requirements a DBMS has to fulfill to facilitate optimization of integrated top N queries on several content and alpha numerical types.

Within the scope of this research I plan to implement a general optimizer architecture for Moa [BWK98, VW99], the extensible, structured object algebra being developed by our group. Within the accompanying architecture [VB98, Vri98] of Moa I will implement some of the top N query optimization techniques known in literature and introduce new data base optimizer concepts to suit MM querying. If this succeeds and results in relatively good performance, this research will (partially) fulfill the promise that Moa is an exception to the rule that content querying in a DBMS performs badly.

2 State of the Art

In the information retrieval (IR) field several interesting ideas have been investigated to speed up processing top N retrieval queries. Most of these ideas are based on the theoretically well founded work of [FM, Fag98, Fag99]. The basic idea in this work is that one can take advantage of lists being ordered when processing top N like operations by maintaining the proper upper and lower bound administration while computing the required results. This allows for ending the processing as soon as it is certain that the required top N answers have been computed. Combined with the knowledge that terms in natural language have a Zipf distribution this principle was exploited by [Bro95] using the INQUERY system [CCH92]. Two types of techniques exist: unsafe techniques that speed up the process but might lower the answer quality (e.g. precision and/or recall) and safe techniques that do increase speed, although often much less, but maintain answer quality compared to the unoptimized case.

In the database research field also some work has been done on top N query optimization by [SSM96, CK98, DR99]. Here the ordering of elements is also exploited to stop processing earlier when only a top N of best answers is required. However, these techniques have not been implemented in many systems yet, and are also still very primitive. Furthermore, modern day DBMSs still have trouble with handling extensions during query optimization. Although the E-ADT approach of PREDATOR [SP97] does solve a lot, I will argue that it still does not solve a crucial problem, one that arises in MM querying in particular.

Finally, the complete integration of content based retrieval, MM retrieval in particular, and database

technology still lacks. Most MM DBMSs either are mediator systems like Garlic [HKWY97], or are not able to handle the content in any other way than BLOBs, or do not function efficiently enough to be able to compete with custom build retrieval systems like INQUERY.

3 Approach

To achieve the aforementioned main goal of developing an optimizer architecture for Moa, I distinguish three sub problems. Each of these three corresponds to one of the basic steps that I plan to take as the guideline for my top N query optimization research (see also [BVB99]):

Step 1: fragmentation of the data This step has to do with the physical layout of the data. Since databases preferably operate set-based in contrast with the element-at-a-time operation of most IR systems, IR technology and optimization techniques are not directly applicable in a content based retrieval DBMS. However, some basic principles can be adapted quite easily, like the utilization of the fact that text data is Zipf distributed. This means that the least frequently occurring terms are the most interesting ones while the most frequently occurring/least interesting terms take up most of the storage/memory space. To take advantage of this effect I horizontally fragmented the most important vectors in the database. By processing only a small portion of the data of approximately 5% of the unfragmented size, containing the 95% most interesting terms, I was able to speed up query processing on the FT collection of TREC with at least 60% (see also [VH99]). The answer quality dropped more than 30% due to the unsafe nature of this technique. To solve this problem I inserted a check early in the query plan that is able to detect when the answer quality would be better when the other fragment would be used. This allows query processing to switch accordingly in time. This improved the answer quality significantly but lowered the speed also quite a lot. Currently I'm working on recoding the second fragment and plan to introduce a non-dense index in the system to speed up processing the large fragment. This even will allow for extra computations while still decreasing execution time, bringing the answer quality nearer to or even on the same level as in the unfragmented case. For the case of non-text content data we are yet not aware of a special distribution of the data (such as Zipf for text). Maybe such a distribution can be 'learned' by the system by means of profiling, although the thus found distribution most likely will not be independent from the data set. Finally, introducing special top N operators, which can be seen as special select operators, will allow optimal utilization of the new structure of the data at the query language level.

Step 2: *inter object* optimizer framework This step deals with the requirements at the logical level. Top N operators such as introduced in the previous step must be used in the right place in the query evaluation plan to achieve the best performance. Current query optimizers are not able to support this due to the fact that they are still not very well suited to reason over operators defined in extensions, in particular distinct ones. Note, however, that systems like PREDATOR [SP97] do have a notion of intra-object optimization.

Here follows a small example to make this a little more clear:

Example 1 *Let's look at a system that has a LIST and a BAG structure, both defined as objects by separate extensions (ADTs/data blades/data cartridges). Also, let's assume that both have a select operator that behaves in the usual way, and takes an upper and lower bound to select a range of values. For example* $select([1,2,3,4,4,5],2,4)$ *selects all elements from the list* $[1,2,3,4,4,5]$ *with values ranging from* 2 *up to and including* 4*. This will result in the list* $[2,3,4,4]$.

Besides the select, LIST also provides a projecttobag operator which produces a BAG containing all the elements of the LIST the operator acts on. For example $projecttobag([1,2,3,4,4,5])$ *results in the bag* $\{1,2,3,4,4,5\}$.

Now, consider the expression $select(projecttobag([1,2,3,4,4,5]),2,4)$*. Current optimizer technology, including the E-ADT system of PREDATOR, cannot optimize this expression. However, it is obvious that* $projecttobag(select([1,2,3,4,4,5],2,4))$ *produces exactly the same answer but can be executed more efficient than the original expression. The second expression can be evaluated even more efficiently when the system is aware of the ordering of the elements, which in case of a list is well defined, but formally does not exist for a bag.*

Ranking of documents in a list results often in similar nested operators/structures which are typically defined in different extensions. However, as argued before, ranking a list of documents is the core business of content based retrieval DBMSs. The special kind of optimization that deals with two distinct extensions/structures, which I call inter-object optimization, has not been shown in literature before but can be quite interesting as briefly shown above. Fortunately Moa provides some very good facilities to build such a new type of optimizer layer. This optimizer is conceptually located between the high level, general algebraic logical optimizer and the extension specific optimizer parts, which I call the intra-object optimizers. The last optimizer part I plan to implement like E-ADTs as described in [SP97]. The new inter-object optimizer layer will be responsible for coordinating optimization between operators on distinct extensions.

Step 3: the costs of *real* MM This last step utilizes knowledge of the first step and the facilities of the second step to introduce a notion of costs into the optimization process. Using Moa, we have the means to handle all types of data in one algebra, that does not need any delegation of work to external sub systems. This allows us to keep the cost model much simpler, which clearly has a lot of advantages. This centralized, but still highly flexible approach (due to the flexible nature of our system), shows very interesting opportunities in our opinion.

The goal of my research is to obtain a running optimizer tuned and tested for top N MM queries. I plan to address the three aforementioned subjects in the order as mentioned.

4 Concluding Remarks

The technology I present here is completely new or is placed in a completely different context than it used to be in till now. Hopefully this contributes to a renewed faith in the use of database technology for MM retrieval and IR in particular, in contrast with the current common opinion that DBMSs are not suitable for content based querying.

References

[Bro95] Eric W. Brown, *Execution Performance Issues in Full-Text Information Retrieval*, Ph.D. Thesis/Technical Report 95-81, University of Massachusetts, Amherst, okt 1995.

[BVB99] H.E. Blok, A.P. de Vries, and H.M. Blanken, *Top N MM query optimization: The best of both IR and DB worlds*, Conferentie Informatiewetenschap 1999 [Eng.: Information Science Conference 1999] (P. De Bra and L. Hardman, eds.), Werkgemeenschap Informatiewetenschap, nov 1999.

[BWK98] Peter A. Boncz, Annita N. Wilschut, and Martin L. Kersten, *Flattening an Object Algebra to Provide Performance*, 14th International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, IEEE Transactions on Knowledge and Data Engineering, IEEE Computer Society, feb 1998.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding, *The INQUERY Retrieval System*, 3rd International Conference on Database and Expert Systems Applications, 1992, pp. 78–83.

[CK98] Michael J. Carey and Donald Kossmann, *Reducing the Braking Distance of an SQL Query Engine*, 24th VLDB Conference, New York, USA, 1998, VLDB, 1998, pp. 158–169.

[DR99] Donko Donjerkovic and Raghu Ramakrishnan, *Probabilistic Optimization of Top N Queries*, Technical Report CR-TR-99-1395, Department of Computer Sciences, University of Wisconsin-Madison, 1999.

[Fag98] Ronald Fagin, *Fuzzy Queries in Multimedia Database Systems*, Proceedings of the 1998 ACM SIGMOD International Conference on Principles of Database Systems, Seattle, USA, Sigmod Record, ACM SIGMOD, ACM, 1998, pp. 1–10.

[Fag99] Ronald Fagin, *Combining fuzzy information from multiple systems*, Journal on Computer and System Sciences **58** (1999), no. 1, 83–99, Special issue for selected papers from the 1996 ACM SIGMOD PODS Conference.

[FM] Ronald Fagin and Yoëlle S. Maarek, *Allowing users to weight search terms*, Retrieved from authors website.

[HKWY97] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang, *Optimizing Queries across Diverse Data Sources*, 23th VLDB Conference, Athens, Greece, 1997, VLDB, 1997.

[SP97] Praveen Seshradri and Mark Paskin, *PREDATOR: An OR-DBMS with Enhanced Data Types*, Proceedings of the 1997 ACM SIGMOD International Conference on the Management of Data, Sigmod Record, ACM SIGMOD, ACM, 1997, pp. 568–571.

[SSM96] David Simmen, Eugene Shekita, and Timothy Malkemus, *Fundamental Techniques for Order Optimization*, Proceedings of the 1996 ACM SIGMOD International Conference on the Management of Data, Montreal, Canada, Sigmod Record, ACM SIGMOD, ACM, 1996, pp. 57–67.

[VB98] Arjen P. de Vries and Henk M. Blanken, *Database technology and the management of multimedia data in* **miЯRor**, Proceedings of SPIE, Boston, November, 1998, Multimedia Storage and Archiving Systems III, vol. 3527, nov 1998.

[VH99] A.P. de Vries and D. Hiemstra, *The* miЯRor *DBMS at TREC*, Proceedings of the Seventh Text Retrieval Conference TREC-8 (Gaithersburg, Maryland), nov 1999, To appear.

[Vri98] Arjen P. de Vries, **miЯRor***: Multimedia Query Processing in Extensible Databases*, 14th Twente Workshop on Language Technology, Language Technology in Multimedia Information Retrieval (Enschede, The Netherlands), University of Twente, dec 1998, pp. 37–47.

[VW99] Arjen P. de Vries and Annita N. Wilschut, *On the Integration of IR and Databases*, 8th IFIP 2.6 Working Conference on Data Semantics 8, 1999.