Top N MM query optimization

The best of both IR and DB worlds

H.E. Blok, A.P. de Vries, H.M. Blanken
{h.e.blok,arjen,blanken}@cs.utwente.nl
Computer Science Faculty, University of Twente
PO BOX 217, 7500 AE, Enschede
The Netherlands

Abstract

It is commonly recognized that top N queries belong to one of the most important query classes in IR and MM retrieval, or more general, content based retrieval. A lot of work has been done on query optimization in database research but that research has mainly focussed on the area of optimization of databases in a business application environment. In IR research also work has been done on query optimization but this is not directly applicable in a database environment.

Exploiting the integrated content based retrieval technology in the miЯRor database system, we intend to look into optimization of top N queries in MM DBMSs. This paper provides a brief overview of current IR and database technology relevant to top N MM query optimization. It also describes our DB environment and ideas on three major sub topics in the light of top N MM query optimization: incorporate known fragmentation techniques to ensure scalability, introduce a new intermediate optimizer layer that supports inter-object optimization, develop a cost based optimizer for real MM querying. This paper will mainly focus on the physical design part since that part has the strongest ties with the IR field.

Keywords: top N, query optimization, content based retrieval, multi media, databases

1 Introduction

Most current MM[1] database systems are not really what the name suggests them to be. First of all they often only operate on one single media type, in most cases 2D images, or sometimes audio. Secondly, these systems are not really database systems but merely large monolithical systems with a lot of data. The consequence of this is that these systems still do not really address *Multi* Media retrieval and lack the properties and accompanying advantages of a *real* database system, such as scalability, extensibility, data independency, etcetera.

In our group research is being done on building a *real* MM DBMS [HVBB98, dV98, dVW99, Doo99]. This research views MM retrieval as a general case of content based retrieval. Commonly known statistical IR techniques have been incorporated in a new type of DBMS that promises to overcome the historical efficiency bounds that usually render IR in a DB environment too slow. This new prototype DBMS is build around an extensible, structured object algebra, called Moa [BWK98], which functions as a layer on top of the Monet [BK95] main memory DBMS developed at CWI.

Utilizing the basics from well known text retrieval systems, several generalizations were build: a small one to experiment with audio retrieval and one to retrieve images using thesauri [Doo99],

[1] MM = multi media.

which both showed promising results, demonstrating that with this database system content based MM retrieval should be very well possible. Since our system is able to handle the meta information about all sorts of distinct media types, such as text, audio and images, within one query algebra, e.g. Moa, it is expected to be much more easy to query those types together than in the current MM database systems which either operate on single media or behave as mediator systems like Garlic does [HKWY97].

However, the database properties, like scalability and extensibility, need to be proven to be available, still. Optimization plays an important role in supporting these properties, since it can compensate the extra executions costs involved in scalability and extensibility. Optimization also becomes important when dealing with *real* MM querying, since it then serves the scheduling of how, when and where to evaluate which media type to get the desired results as quickly as possible.

Due to the limited perception capabilities of the human end user, content based retrieval results are often ranked and only the best answers are returned. The queries that provide such kind of answers are often called top N queries. It also might be very clear that these queries play a very important role in the content based retrieval area and that optimizing this type of queries might be very rewarding.

In this paper we describe our ideas on the optimization of top N queries in a MM DBMS setting.

The rest of this paper is structured as follows. First we introduce the top N query optimization problem a little more in section 2. In section 3 we give an overview of important techniques from the IR and database field relevant to the top N query optimization problem. In section 4 we describe our ideas on what we as three major topics that one has to deal with when one wants to build top N optimization facilities into an extensible MM DBMS. Finally we end this paper with some concluding remarks in section 5.

2 Problem

A typical content based retrieval query is usually evaluated by computing some ranking based on statistics and distances in feature spaces. The returned 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, in particular when done the naive way. 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 IR field it is known that usually half of all objects (e.g. documents) in the considered collection contains at least one query term, meaning that even if a system only considers these objects it still has to process a lot of them. The second reason is that ranking certain media types can be very expensive per media object. So the number of objects and the computational effort needed just to evaluate one single object both quickly result in excessive computational costs, unless we start making use of the special characteristics of the data to speed up the querying process.

The problem of top N MM query optimization is to find such techniques that utilize this knowledge, in an effort to:

1. Limit the total set of objects of all types taken into consideration during the ranking process as much and as soon as possible,
2. Limit the set of objects with expensive ranking functions, before ranking them, by using ranking results from objects with less expensive ranking functions, in case more than one media type is involved in the query.

Several techniques are known in literature from both the IR field and the database field to achieve such effects more or less thoroughly. In this document we provide an overview of those techniques and try to sketch directions in which those concepts might become interesting to be used in the context of MM database research in our group as described previously.

3 State of the Art

3.1 IR Techniques

IR is a quite old research area (compared to MM retrieval research) resulting in a lot of different techniques. A large group of these techniques makes use of certain statistical methods. We mainly concentrate on (optimization) techniques concerning this group of statistics based methods because:

- this group of techniques has been proven to work quite well, so results from this research are more likely to be widely applicable,
- (also because of the previous reason) the techniques used in our group belong to this group as stated in the introduction, thus already providing a suitable platform to build on, making it much more easy to arrange for experimental verification of this research.

This section is structured as follows. First we give an overview of basic technology in the field. Next we provide a brief overview of common optimization techniques that have been developed over the past years in this research area. In the third subsection we briefly touch on some of the most important advantages and disadvantages of the common approach in this field.

3.1.1 Query Processing in IR

This subsection contains two paragraphs, reflecting the two major phases in the life of an IR system. The first paragraph addresses the first phase: indexing. The second paragraph addresses the retrieval phase.

Indexing Phase The basic components of a statistics based information retrieval system, constructed during some kind of indexing process, are[2]:

- a (usually large) set of text documents (denoted by $\{d_j\}$),
- a set of all terms occurring in the documents set (denoted by $\{t_i\}$),
- a set containing the frequency of every term per document, i.e. the number of times each term occurs in each document (denoted $\{tf_{ij}\}$),
- a set containing the in document frequency of every term, i.e. the number of documents a term occurs in (denoted by $\{df_i\}$).

Usually the $\{tf_{ij}\}$ and $\{df_i\}$ are both normalized to eliminate the effects of document respectively collection size. The resulting sets are respectively denoted as $\{ntf_{ij}\}$ and $\{ndf_i\}$.

The parts we describe here are the most important ones. However, a lot of additional techniques are known and used to increase computational efficiency and speed and minimize requirements on system resources like disk, memory and/or CPU. Also some analysis algorithms may be run over the terms like stemming (reducing several terms to one common base form: for instance convert different forms of the same verb to its base).

[2]For a more elaborate overview [Bro95]

Retrieval Phase Given a set of query terms (denoted by $\{q_k\}$) it takes usually the following steps to compute a ranking of the document set $\{d_j\}$:

1. Compute the subset of the terms known by the system and occurring in the query, let us denote this set by $QT := \{t_i\} \cap \{q_k\}$.
2. Compute the subset of the documents known by the system which contain at least one of the terms in QT.
3. Compute the corresponding subsets of $\{ntf_{ij}\}$ and $\{ndf_i\}$, here denoted by $\{ntf_{ij}\}'$ and $\{ndf_i\}'$ respectively.
4. Compute the belief[3] contributions per document per term
$$B := \{bel_{ij} | bel_{ij} = \frac{ntf_{ij}}{ndf_i}, ntf_{ij} \in \{ntf_{ij}\}', ndf_i \in \{ndf_i\}'\}$$
.
5. Compute the final belief per document by aggregating the belief contributions per document per term.

The quotient $\frac{ntf_{ij}}{ndf_i}$ is usually formulated as the product $ntf_{ij} \cdot nidf_i$, where $nidf_i = \frac{1}{ndf_i}$.

Since a user is limited in his/her perception only a limited set of most relevant (i.e. highest ranking) documents is to be presented to the user. Therefore the result of the last step above is usually followed by a selection and ordering of the highest ranking documents. The number of documents finally returned to the user is in most cases not more than a very small subset of the whole documents set $\{d_j\}$.

3.1.2 Query Optimization in IR

In this subsection we provide a brief overview of the most important optimization techniques in the IR field.

This section is divided in three paragraphs. The first paragraph describes the basic algorithm on which most of the optimization techniques (that we are interested in) are based. The second paragraph shows the possible consequences for the quality of the returned answers when using certain versions of this algorithm. The last paragraph describes some issues for physical design of the data storage which might help increase the profits of the algorithm in the first paragraph.

Effects of a Smarter Algorithm In IR-research a lot of effort has been done on optimizing query processing in IR-systems. Most of the work done in this area is based on some quite simple ideas from [Fag98, Fag99, FM]. In these papers several algorithms are proposed that stop executing as soon as the required answers are computed. Most importantly is that these algorithms and especially the used stop criteria are proven to be correct. This means that using these algorithms under the given constraints will result for sure in the required set of answers or a superset of those.

The basic principle of the algorithms goes as follows (loosely formulated):

Given two sequences of equal length, say $A = [a_i]$ and $B = [b_i]$, of numbers, with $|A| = |B| = n$.

We want to compute a subsequence D of C, with $C = [c_i | c_i = a_i \cdot b_i]$, where $|D| = N$ and $min(D) \geq max(C - D)$, meaning D contains the N highest elements of the product of A and B.

The steps of the algorithm are then (roughly speaking):

[3] We use the term *belief* here because this product is called this way in the system developed at our group.

1. Suppose we order one of the sequences, say A, descending. The resulting sequence A' is therefore a permutation of the original sequence A. Let's call this permutation π. Then:

$$A' = [a'_i | a'_i = a_{\pi(i)}]$$

   such that:

$$\forall i \in \{2, \ldots, n-1\} : a'_{i-1} \geq a'_i \geq a'_{i+1}$$

2. Create a sequence B', that has the same ordering as A':

$$B' = [b'_i | b'_i = b_{\pi(i)}]$$

3. Create an empty set E.
4. For each ascending $i \in \{1, \ldots, n\}$:
   (a) Compute $v = a'_i \times b'_i$.
   (b) IF $|E| < N$ OR $v > min(E)$ THEN $E := E \cup \{v\}$.
   (c) IF $|E| > N$ THEN $E := E - \{min(E)\}$.
   (d) IF $max[b'_{i+1}, \ldots, b'_n] \times a'_i < min(E)$ THEN stop looping over i.
5. Create sequence D from set E by ordering the elements ascending. D is the requested top N of the product of A and B.

Both [Bro95] and [CP97] use (modified versions of) this idea to calculate the document belief-contribution per term, also known as the $tf \cdot idf$. Where tf is taken as the A and idf as B or vice versa. In fact, this means that one computes the $tf \cdot idf$ of the most promising tf (or idf) before the $tf \cdot idf$ of the lesser promising tf (or idf), hoping that not all tf (or idf) values will have to be taken into consideration but only a certain, hopefully small, set of 'best' ones. Of course the algorithm takes some extra administrative overhead that will make the algorithm only profitable in cases where the considered part of the ordered sequence A is small enough to compensate this overhead.

Note that tf and idf are not of equal length, so formally these two sequences cannot be used in the mentioned algorithm. By repeating each idf_i value for all tf_{ij} with corresponding i and distinct j this little problem can be fixed. The implementation of IR techniques used in our group uses this trick. Of course, with a slight, more or less trivial, modification of the algorithm, the desired effect can be achieved without repeating values.

Safe and Unsafe methods An extra option is to skip the last step of ordering the final results. In cases of IR this might be very well acceptable for several reasons. Two of these reasons are:

- The top N is the most important issue, the ordering is often less important.
- If the elements in E do not differ much, the ordering of E is disputable since these figures result from statistically based computations of which the error margin might be much greater than that inter-element margin.

This point also brings me to another point in general, noted by [Bro95] as the difference between so-called *safe methods* and *unsafe methods* of optimization:

safe methods This class of methods still return (at least) the required answers in the desired form (i.e. ordering and such).

unsafe methods This class of methods promise to return good answers, but do not guarantee to return all the required answers or in the desired form. Correct answers might be replaced by other, usually just a little less optimal, answers. And, as already addressed above, the answers might not be ordered correctly.

Since IR is by no means exact retrieval, in contradiction to querying a DBMS containing only alphanumerical data, *the* good, unique answer does not exist, or cannot be determined. Only good answers and better answers exist. And even the question which answer is better than the other one is hard to answer solidly. Also, in systems where the querying process is an interactive one with relevance feedback from the user, even the worse elements in an answer set might serve a good purpose of providing the user with an option to tell the system what he/she explicitly does not want.

Therefore a great reduction in query execution time might very well be preferable above a 'better' answer of which the computation takes much more time.

Effects of Physical Design One other important issue is the tuning of the storage system underneath/within the IR-system. By placing the data on disk in the right order, with the right clustering and appended by some efficient access accelerators, a lot of time can be saved, just streamlining disk read IO. [Bro95] uses properties of the IR level to place the document-term statistics efficiently ordered in an inverted file structure. The statistics are stored in descending *idf*. This way he achieves two effects in one:

1. He is able to use an algorithm as described above.
2. He is able to consider a lot of different terms per block read operation. The reason for this is that a large *idf* means that a term occurs in only a few documents. This results in a small data segment in the inverted file for a certain term (per term the list of documents is small). In turn, this means that a lot of those small document-per-term lists can be stacked in one block.

[Bro95] implemented and tested his ideas using INQUERY, and claims interesting performance improvements, and only a minor decrease in precision and recall.

3.1.3 (Dis)advantages of the IR approach

The main advantage of the approach in IR optimization techniques as described previously is the fact that the element-at-a-time iteration while computing the beliefs allows recomputation of the boundaries that form the basis for the stop criterium. As proven by [Fag98, Fag99, FM] these accurate boundaries can result in computation of only a very minimal set of belief candidates which of course is highly computationally efficient.

However, most IR systems tend to be quite monolithic requiring lots of reimplementation for sometimes only minor conceptual model changes. Also the integration with other systems can be difficult or highly inefficient [dVW99].

On the other hand, the theoretical work of [Fag98, Fag99, FM] is clearly very well founded and provides means for very interesting optimizations, which also holds for the work of [Bro95]. Due to the characteristic Zipf distribution of the indexed content data these methods often result in very high profits with often only very limited (when using unsafe methods) or even no (when using safe methods) loss in answer quality. However, these techniques did unfortunately not get implemented in most of the IR systems. The work of [Bro95] was implemented using INQUERY but we are not sure whether it is still used in the current INQUERY implementation.

The main issue that summarizes the negative aspects of IR-systems is that they tend to be quite inflexible to adaptation of new technologies since it requires a lot of code replacements.

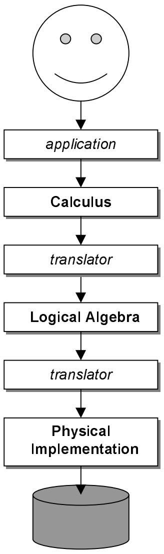

Figure 1: Query processing layers in a typical relational DBMS.

3.2 DB Techniques

DBMSs were originally designed as smart data storage systems which could handle data requests from users on a higher level than the file system of an OS can. In administrative environments relational DBMSs have proven to be quite useful. However, that data usually consisted of just alphanumerical parts of limited (often even fixed) lengths. Due to practical demands most DBMSs nowadays are extensible: new data types and/or operations can be added. Often also other functionality can be added to the DBMS like new index/access structures. Modern DBMSs also often support OO data modeling and access interfaces instead of, or additionally to, the relational way of handling the data, in an attempt to cope with the growing number of different uses and stored types of data.

Note that several types of DBMSs exist. Two of the most important ones are:

- relational DBMSs (or RDBMSs),
- object oriented DBMSs (or OODBMSs).

RDBMSs is by far the oldest of these two types, and also the most widely used one. OODBMSs might seem very interesting due to their OO nature but still are often not much more than persistent object stores which cannot compete in efficiency and scalability with their relational counterparts. Also attempts to build an OO layer on top of an RDBMS resulted in most cases in unrealistically unhandy or slow systems. Since our research has more to do with relational than OO data base technology we will not elaborate on the latter one any more.

For a more complete overview on relational database theory see [Dat95].

3.2.1 Query Processing in a DBMS

Note that relational DBMSs are designed to operate on sets providing operations to retrieve strictly defined subsets or elements given a query. This is in contrast with the basic design of IR systems which is usually referred to as providing inexact retrieval (as opposed to exact retrieval in a relational DBMS).

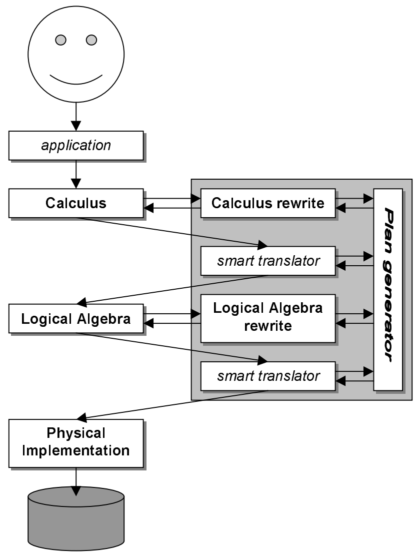

Figure 2: Adding query optimization to the three layered query processing architecture.

Most relational DBMSs are currently build in divided in three, independent, layers, on top of each other:

1. Conceptual level
2. Logical level
3. Physical level

The conceptual level is the most abstract level and is mapped onto the logical level, which in turn is mapped onto the physical level. Queries are formulated in a declarative way at the conceptual level and translated into an algebra at the logical level. The algebra then is translated to methods on the physical level [Figure 1].

The current standard conceptual level query language is SQL[4], with both calculus and algebra elements in it [Ste95]. The algebra at the logical level and the operators at the physical level are system specific whereas the implemented SQL version only shows minor extensions and/or differences, usually concerning constructs not or not clearly defined in the standard. Another importing issue to note is that the conceptual level language (i.e. SQL) usually is tuple oriented whereas the two lower levels usually are set based.

3.2.2 Query Optimization in a DBMS

As described in [Dat95, Ste95, Cha98] and shown in [Figure 2] optimization can take place at almost every level in the system.

Just a naive translation from a calculus expression into an algebra expression will in most cases result in a so called most costly expression [Ste95]. This means that the resulting algebra expression has even worse execution time than nested loop iteration over all elements in the data set processed for the query. Therefore it is not also possible to optimize at all levels but necessary to take advantage of every possibility, too.

The basic conceptual components of a relational query optimizer are [Figure 2]:

[4]SQL = Structured Query Language

Calculus rewriter In this phase it is often difficult to really optimize an expression. In most cases it it used to transform the expression into some normal form thus limiting the search space for lower level optimization processes. However, normalization should be done careful to avoid decreasing execution performance promises.

Calculus to logical algebra translator It is at this level that a lot of performance can be gained by trying to achieve a not too naive translation since otherwise the resulting algebra expression in most cases would provide a very inefficient starting point for any further optimization. However, in literature not much information is available on this important link in the chain of query processing.

Logical algebra rewriter At this level algebra equivalence rules (based on common algebra properties like commutativity, associativity, distributivity, idempotency, etc.) are used to transform the expression into a form that should translate more easily and more efficiently into the physical algebra at the physical layer. Most systems do not use a cost function at this level but just (simple) heuristics that usually produce better expressions than the initial input. Commonly known heuristics are: push select/project down or the more sophisticated predicate move around [HS93, LMS94, Hel98], most restrictive join first.

Logical algebra to physical algebra translator This stage aims at translating the logical algebra expression into the most efficient program at the physical level. It is usually at this stage that a cost model of the physical layer is used to find the cheapest executable program. The costs of a certain program are computed by estimating the costs of each operator given known or estimated data properties such as number of tuples [Dat95], ordering [SSM96, CK97a, CK97b, CK98, DR99], availability of access accelerators, etc. Also for each operator the properties of the results are computed/estimated which in turn can server as input for costs computations of depending operators [Cha98].

Plan generator The plan generator is responsible for keeping track of all possible plans that have been or are taken into consideration. In [Figure 2] we placed it as a global component next to the three other stages. This is just a way of displaying things because at a certain level knowledge of all lower levels might be required. The reason that such knowledge is required at the higher level is that a certain decision at that level might result in better or worse optimizable/executable query plans at subsequent lower levels, thus influencing that high level decision.

3.2.3 (Dis)advantages of the DB approach

The main advantages of any DBMS is its flexibility to store and manipulate a lot of data in a more or less abstract manner. In particular the ANSI/SPARC properties allow changes at a certain level without having to modify large parts in other levels of the system [Dat95]. The price of this layered structure is a lot of inefficiency. Fortunately the fact that many systems use one or more algebras to handle the data allows for a great deal of optimization. The other advantage is that the optimization is usually better than in cases where a user needs to implement the physical operator implementation by himself: the system can incorporate of the shelf optimal implementations which resembles the union of programming skills of several expert programmers together [Dat95]. Also because of the set-based/bulk behavior of, in particular relational systems, IO can often be arranged more efficiently since several types of overhead can be eliminated over larger quantities of data.

One of the few disadvantages of this bulk behavior and/or lack of control of how elements are iterated over finally, is the fact that stop criteria as the ones used in section 3.1.2 often cannot be computed equally well, disallowing such efficient breaks in computing over large sets.

4 Our Approach

In this section we describe our approach concerning top N MM query optimization, given the problems and solutions provided by the IR and DB research areas as described in the previous sections.

First we describe briefly our DB environment. Next we give an overview of our approach including some first results.

4.1 Our DB Environment

In this section we briefly describe the software systems that we have been using and/or developing lately. We start with introducing Monet, a main memory DBMS developed at CWI[5]. Next we will introduce Moa, a structured object algebra, currently build on top of Monet. After introducing both Monet and Moa, we give an introduction to the MM retrieval work that has been going on at that this group and that is being implemented using Moa.

4.1.1 Monet

Monet[6] is the main memory DBMS that is being developed by CWI. Monet is able to cope with modern day DBMS requirements emerging from areas like DSS[7] and MM retrieval. These areas often use wide tables with very many elements/tuples. However, during query processing only a few columns are needed at a time. To exploit this property Monet uses full vertically fragmentation, resulting in a binary data model in which tables consist of only two columns, so called BATs[8]. Relations that need more than two columns can be made by joining several BATs. This vertical fragmentation allows Monet to keep the hot set[9] in main memory, in contrast with most other DBMSs. This way Monet outperforms most other systems when used in situations that wide relations with many elements do exists but only a few fields are part of the hot set, like often occurs in DSS and MM retrieval. Monet has proven to be fast in those cases using several benchmarks like OO7 [vdBvdH96], TPC-D [BWK98] and Wisconsin.

In the scope of this paper it is also important to note that Monet uses a dynamical physical query optimization system, choosing the right physical operators just before the operator needs to be executed, thus ruling out the usual error introducing guess work during optimization. The resulting better optimization decisions, facilitate speeding up query execution even more.

4.1.2 Moa

Moa [BWK98] is an extensible structured object algebra which is mapped on top of a binary physical algebra. Currently Monet is used as the binary algebra layer underneath Moa. Moa is designed as an intermediate language, e.g. logical algebra, on which a high level language like OQL could be mapped. Moa was originally developed for GIS[10] and is being extended to serve other areas, like content based retrieval, as well.

One of the key advantages of Moa over other systems is that the structured object nature of Moa allows for unlimited nesting of structures without loss of performance. Most other systems that allow deep/unlimited nesting usually tend to be very slow since they iterate in a depth-first manner through the nested structures during query processing. In case of a large database with

[5] CWI = Center for Mathematics and Computer Science, Amsterdam, The Netherlands.
[6] See [BK95, KBQ+97, BK] or on the CWI web site http://www.cwi.nl/~monet for more info.
[7] DSS = decision support system.
[8] BAT = Binary Association Table
[9] hot set = parts of the database under consideration at a certain moment during query processing
[10] GIS = geographic information system.

many levels of nesting this element-at-a-time query processing has proven to highly inefficient compared to the set-at-a-time way of query processing common to most relational DBMSs, that do not allow nesting at all. Moa combines the best of both worlds in that it allows nesting but processes queries in a set-based manner.

4.1.3 MM Retrieval

Exploiting the facilities of Moa and Monet a MM DBMS is being built. Currently a basic extensible open distributed architecture has been set up [HVBB98] providing a framework around Monet and Moa. With respect to the content retrieval part currently a text retrieval system has been implemented in the Moa/Monet environment [dV98, dVW99]. The retrieval model originally used resembles the INQUERY inference network [CCH92]. Lately another retrieval model [Hie98], which had proven been to work quite good on TREC, has been implemented in the system as an alternative for the INQUERY model. Our text retrieval DBMS using this new model has is been evaluated at TREC-8 [VH99].

We plan to generalize the text retrieval system to become a MM retrieval system. We have already been experimenting with audio retrieval (only on a very small an primitive scale) and image retrieval [Doo99] using the generalized text retrieval framework.

It is in the context of this MM retrieval setting where our top N MM query optimization research is situated.

4.2 Our Optimization Efforts

When looking at top N MM query optimization, in particular in a setting where we use Moa on top of Monet, the following three main topics are interesting to look into:

- Fragmentation,
- Intra v.s. inter object/ADT optimization,
- Real MM querying.

These three topics can, to some extent, be linked to the three levels in the ANSI/SPARC architecture, often, although not quite correctly, designated by:

- Physical level,
- Logical level,
- Conceptual level.

Furthermore it should be noted, that right from the start, for any of the topics, it is also necessary to define and set a baseline for the performance of the system that will serve as a reasonable reference point to compare an optimized version of the system with. Note that performance in an IR or MM retrieval system does comprise more than execution time alone. Also the precision and recall are part of the performance. Since no combination function for these three components (execution time, precision, recall) exists, performance should be noted as the 3-tuple. Display techniques like a 3D-projection of a plot of the performance points for different situations might provide an easy way for humans to compare the results in a flexible way (i.e. not aggregating the 3-tuple into one number, thus not throwing away a lot of information).

4.2.1 Fragmentation [Physical design]

To facilitate any optimization on a higher level and to make use of the main memory characteristics of Monet we need to (horizontally) fragment the data in Monet. In case of a real world application the size of the data even will be that huge that fragmentation will be necessary to get things running anyway, since otherwise memory will be too limited to do anything at all.

Looking at paragraph 3.1.2, where the ideas of [Bro95] concerning storage of IR data are explained, fragmenting *tf* (or *ntf*) and *idf* (or *nidf*) vectors in Monet in a similar manner based on *idf* (respectively *nidf*) seems to be an interesting first choice.

However, since we are dealing with a database system that preferably operates set based, in contrast with the element-at-a-time approach used in the INQUERY system, the ideas of [Bro95] are note directly applicable in our case. In other words: we have to adapt these ideas to suit a set based way of query processing. Since we want the best of both worlds a way in between seems to very interesting. A possible solution might be exploiting the commonly known Zipf distribution of the IR data. The Zipf law for the English language says that that the distribution of terms over a set of documents is hyperbolically correlated to the number of those documents they occur in. This means for example that 5% of the terms is related to 95% of the total amount of data in the retrieval system. These are the terms that occur in a lot of documents and therefore have a low *idf*. The other 95% of the terms only is responsible 5% of the space.

Further note that the *tf* vectors used to compute the beliefs are in fact inverted lists noting a *tf* for each term per document it occurs in. This means that using the observation about the Zipf distribution of this vector, we get two (horizontal) fragments:

- a fragment containing the *tf* values corresponding with the other 95% of the terms, with higher *idf*, only using 5% of the space.
- a fragment corresponding to 5% of the terms corresponding with the terms with the lowest *idf*, but taking up 95% of the space,

Now, we can start the evaluation of a typical IR document ranking query by ignoring the second fragment that takes 95% of the space. This way we can compute a quite good belief estimate by only having to consider 5% of the original document statistics. Assuming that time complexity is linear in the size of the considered vectors this reduces execution time with 95%. Some first experiments based on this idea show that execution times at least drop with 60%.

As described in paragraph 3.1.2, this technique is unsafe, which is obvious, since not all terms are taken into consideration thus ignoring belief contributions from those terms. And also, although the terms that are ignored have a low *idf* and are therefore unlikely to contribute much to document beliefs, this does not exclude the possibility that some indeed would have contributed significantly in the case when they have a very high *tf* that compensates for the low *idf*. Resulting from the same early experiments that showed the 60% time profit we found that in the case of the above suggested fragmentation precision drops a few percents and recall drops not more than 30%.

Some additional, but still primitive, experiments using different percentages for fragmentation and/or more than two fragments (up to 20) have shown that better precision and recall, approaching the non-optimized versions, or higher executions time gains, over 90%, can be achieved. More thorough experiments need to be done to get a better view on the trade off between execution time and answer quality. We expect to be able to reach an optimization level that still keeps the quality of the answers in terms of precision and recall equal to the non-optimized query processing system but with at least 25% faster execution times. The test were performed using the TREC topics 300 till 350 on the Financial Times sub collection of TREC.

We also are thinking of some other experiments. We did for instance neglect any restrictions that need be obeyed when one wants to really exploit the benefits of main memory execution, as provided by Monet. When running main memory it is important to keep the hot set scaled to fit in main memory at all time. This brings us to the question what the best fragment size and number of fragments are to keep swapping to disk as limited as possible. In case of too many too small fragments, a lot of administrative overhead will result in a loss of speed. Also small fragments contain fewer information needed at query processing time increasing the probability that several fragments need to processed to reach a certain level of answer quality. In case of too large fragments the chances that other fragments need to be considered due to a lack of information are low. However, processing a single large fragment can take much time, approaching the evaluating time of an unfragmented database or even worse due to administrative overhead. Also, the case that extra fragments need to be considered still is not excluded and when that occurs it increases execution time extensively.

4.2.2 Intra v.s. inter object/ADT optimization [Logical design]

Although the techniques used in the fragmentation approach described in the previous paragraph are mostly realized on the physical level, in our case Monet, it certainly has several consequences for the logical level. First of all it is obvious that given the different physical lay out the translation of logical algebra expressions to physical algebra expressions need to be adapted accordingly. Secondly, the new translations allow the easy introduction of some special top N operators [Bro95, CK98] on the logical level, that can exploit the new opportunities in a better way than was possible before. This can of course be seen as giving up some of the independency between the two layers but it is also likely to be interesting from an optimization point of view since it allows more efficient expressions at the logical level when dealing with top N queries.

However, since the best mapping of logical algebra expressions on physical expressions is often dependent on the application area, most systems provide extensibility, as does Moa. This allows for adding new structures and/or operators to the algebra to suit specific needs, thus adding the best possible translation(s) for that case. Extensibility of systems is quiet common but keeping optimization at an acceptable level is still a very large problem.

In the PREDATOR project [SP97] a partial solution for this problem is given. Their system architecture does allows for new structures and operators to be added using, what they call, E-ADTs, or Enhanced ADTs. E-ADTs are ADTs but also provide optimization algorithms for optimizing operations in that ADT. The global logical query optimizer delegates optimization to the E-part of the E-ADT if it finds itself unable to optimize the ADT operators in a query expression.

This is, as we said, only a partial solution. E-ADTs are only aware of themselves so they are only able to optimize when two or more operations on that particular E-ADT are involved [Example 1] but fail when operators of distinct ADTs interact [Example 2].

Example 1 (E-ADT optimization succeeds [SP97]) *Consider an Image ADT with two operators: clip and rotate.*

The clip operator takes as operands an image and the x and y coordinate of the upper left and lower right corner of a rectangle within the bounds of the given image.

Suppose we have an image myimage of 40 pixels wide and 60 pixels high.

An example use of the clip operator would be:

$$clip(myimage, 0, 0, 9, 19)$$

which produces a sub image of 10 pixels wide and 20 pixels high of myimage containing its upper left corner (image x coordinates increase in the normal way to the right but y coordinates increase usually downwards instead of upwards).

The rotate operator takes as operands an image and the angle to rotate over in degrees, counter clockwise.

An example use of the rotate operator would be:

$$rotate(myimage, 90)$$

which produces the a copy of myimage placed on its left side.

The following query, combining these two operators, is also valid:

$$clip(rotate(myimage, 90), 0, 0, 9, 19)$$

which produces the upper left corner area of 10 pixels wide by 20 pixels high of the rotated version of the image.

An alternative expression with exactly the same answer is:

$$rotate(clip(myimage, 20, 0, 9, 39), 90)$$

Since the rotate in the second expression has to rotate a smaller images than in first expression, whereas the clip region stays of the same size, the second expression will be executed faster. The E-ADT system now provides the means to the optimizer to translate the first expression, when encountered, into the second one.

Example 2 (E-ADT optimization fails) *Let's look a a system that has a LIST and a BAG structure, both provided by separate ADTs. 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 following 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, anybody can see 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 efficient 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.

Now note that a top N operator is in fact a special select operator on a list structure. Furthermore, more sophisticated versions of expressions like in [Example 2] often occur in top N queries (it goes to far to explain those expression here). This means that it would be preferable to have some optimizer system that can cope with such expressions incorporating operators from different ADTs. We propose to introduce an extra optimizer layer, between the global logical optimizer and the optimizer parts within the (E)ADTs, to fill the gap. We call this an inter ADT (or in generalized case: inter object) optimizer, whereas we call the E part of the E-ADT system an intra ADT (or intra object in the generalized case) optimizer.

4.2.3 Real MM querying [Conceptual design]

Our final challenge is to make our top N query optimizer MM aware. Since we plan to generalize our DBMS to act as a *real* MM DBMS that is able to provide facilities to query different media types in an integrated manner in one query expression, we also get a system that provides new opportunities to optimize. Consider for instance the case in which we deal with both alphanumerical data and images. The alphanumerical data might contain classification information or who the author is of a certain image, so we might be interested in asking for images with certain content and require they are made by a given author. The big question is now, in what order do we process the different query parts: are we going to select on the author field first and then look at the content or do we select on image content prior to selecting the ones with the right author. In this case one would tend to prefer the first option since selecting on alphanumerical attributes usually is much cheaper than processing images on content. However, when dealing with a query that incorporates two equally *expensive* media types this order is not so obvious. To solve this problem one again needs inter ADT/object optimization facilities, in this case augmented with cost information. It is only at the inter ADT level that a system can decide properly, having an idea of the execution costs of the involved ADT operators given their operands, what operator should be executed first or second. This type of cost based query optimization at the logical level and in a MM context is entirely new and we see it as a very interesting and promising area to look into. Note that the conceptual level now having to deal with more than one media type is the key reason for the necessity of the introduction of this new cost model.

5 Concluding Remarks

By providing an overview of existing IR and database optimization techniques and describing three important research topics we have tried to demonstrate that still a lot of work has to be done when one wants to optimize top N queries in a MM database environment.

The three topics as described can be seen as relatively separate topics. On the other hand, it also might be clear that they facilitate and use each other to a certain extent. We are also aware of the fact that the mentioned topics might very well be much larger than depicted here and therefore might not all be solved in the near future due to time limitations. For this reason we have chosen to just start at the ground level, which fragmentation. Since this is more a physical issue, whereas our interest in mainly in the logical level of Moa, we will set up just a basic fragmentation and do not plan to investigate it more than necessary to suit the use Monet environment. That way we can proceed on the, in our opinion, more interesting part of inter ADT optimization. The *real* MM query optimization will probably to far fetched for the near future and is left for the long term.

Our goal for the next few years is to set up a good physical basis, as said, and build a prototype optimizer for Moa, including an inter ADT layer. If all works out fine this will provide a new kind of query optimizer that, in contrast with the current query optimizers, is suited to optimize modern day DBMS in a MM environment.

References

[ACM97] ACM SIGMOD, *Proceedings of the 1997 ACM SIGMOD International Conference on the Management of Data*, Sigmod Record, ACM, 1997.

[ACM98] ACM SIGMOD, *Proceedings of the 1998 ACM SIGMOD International Conference on Principles of Database Systems, Seattle, USA*, Sigmod Record, ACM, 1998.

[BK] Peter A. Boncz and Martin L. Kersten, *MIL Primitives For Querying A Fragmented World*, VLDB Journal, accepted.

[BK95] Peter A. Boncz and Martin L. Kersten, *Monet: An Impressionist Sketch of an Advanced Database System*, Basque International Workshop on Information Technology, Data Management Systems, San Sebastian, Spain, July 19-21, 1995 (BIWIT'95), IEEE Computer Society Press, jul 1995.

[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.

[BWK98] Peter 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 Internationl Conference on Database and Expert Systems Applications, 1992, pp. 78–83.

[Cha98] Surajit Chaudhuri, *An Overview of Query Optimization in Relational Systems*, In *Proceedings of the 1998 ACM SIGMOD International Conference on Principles of Database Systems, Seattle, USA* [ACM98], pp. 34–42.

[CK97a] Michael J. Carey and Donald Kossmann, *On Saying "Enough Already!" in SQL*, In *Proceedings of the 1997 ACM SIGMOD International Conference on the Management of Data* [ACM97], pp. 219–230.

[CK97b] Michael J. Carey and Donald Kossmann, *Processing Top and Bottom N Queries*, IEEE Bulletin of the Technical Commitee on Data Engineering **20** (1997), no. 3, 12–19.

[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.

[CP97] Douglass R. Cutting and Jan O. Pedersen, *Space Optimizations for Total Ranking*, Proceedings of RAIO'97, Computer-Assisted Information Searching on Internet, Quebec, Canada, June 1997, jun 1997, pp. 401–412.

[Dat95] C.J. Date, *An Introduction to Database Systems*, 6th. ed., 1995, ISBN 0-201-54329-X.

[Doo99] M.G.L.M. van Doorn, *Thesauri and the Mirror retrieval model: A cognitive approach to intelligent multimedia information retrieval*, Master's thesis, Faculty of Computer Science, University of Twente, Enschede, The Netherlands, jul 1999.

[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.

[dV98] 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.

[dVW99] 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.

[Fag98] Ronald Fagin, *Fuzzy Queries in Multimedia Database Systems*, In *Proceedings of the 1998 ACM SIGMOD International Conference on Principles of Database Systems, Seattle, USA* [ACM98], 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.

[Hel98] Joseph M. Hellerstein, *Optimization Techniques for Queries with Expensive Methods*, ACM Transactions on Database Systems **23** (1998), no. 2, 113–157.

[Hie98] Djoerd Hiemstra, *A Linguistically Motivated Probabilistic Model of Information Retrieval*, Proceeding of the second European Conference on Research and Advanced Technology for Digital Libraries, ECDL'98 (Christos Nicolaou and Constantine Stephanidis, eds.), Springer-Verlag, 1998, pp. 569–584.

[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.

[HS93] Joseph M. Hellerstein and Michael Stonebreaker, *Predicate Migration: Optimizing Queries with Expensive Predicates*, Proceedings of the 1993 ACM SIGMOD International Conference on the Management of Data, Washington DC, Sigmod Record, ACM SIGMOD, ACM, may 1993, pp. 267–276.

[HVBB98] E. van het Hof, A.P. de Vries, H.E. Blok, and H.M. Blanken, *Een architectuur voor multimedia databases [Eng.: An Architecture for Multimedia Databases]*, Conferentie Informatiewetenschap 1998 [Eng.: Information Science Conference 1998] (E. de Smet, ed.), Werkgemeenschap Informatiewetenschap, dec 1998, In Dutch, pp. 89–106.

[KBQ$^+$97] M.L. Kersten, P. Boncz, W. Quak, N. Nes, and J. Karlsson, *The Monet System, version 4.0 (BETA)*, Tech. report, CWI and University of Amsterdam, Amsterdam, oct 1997.

[LMS94] Alon Y. Levy, Inderpal Singh Mumick, and Yehoshua Sagiv, *Query Optimization by Predicate Move-Around*, 20th VLDB Conference, Santiago, Chile, 1994, VLDB, 1994.

[SP97] Praveen Seshradri and Mark Paskin, *PREDATOR: An OR-DBMS with Enhanced Data Types*, In *Proceedings of the 1997 ACM SIGMOD International Conference on the Management of Data* [ACM97], 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.

[Ste95] Hennie Steenhagen, *Optimization of Object Query Languages*, Ph.D. thesis, University of Twente, Enschede, The Netherlands, oct 1995.

[vdBvdH96] C.A. van den Berg and A. van der Hoeven, *Monet meets OO7*, Proceedings of OODS'96, jan 1996.

[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.