Indexing and selection of data items in huge data sets by constructing and accessing tag collections

Sébastien Ponce
CERN, Geneva
LHCb Experiment
sebastien.ponce@cern.ch
tel +1-41-22-767-2143

Pere Mato Vila
CERN, Geneva
LHCb Experiment
pere.mato@cern.ch
tel +1-41-22-767-8696

Roger D. Hersch
Ecole Polytechnique
Fédérale de Lausanne
Computer Science Department
RD.Hersch@epfl.ch
tel +1-41-21-693-4357
fax +1-41-21-693-6680

Abstract

We present here a new way of indexing and retrieving data in huge datasets having a high dimensionality. The proposed method speeds up the selecting process by replacing scans of the whole data by scans of matching data. It makes use of two levels of catalogs that allow efficient data preselections. First level catalogs only contain a small subset of the data items selected according to given criteria. The first level catalogs allow to carry out queries and to preselect items. Then, a refined query can be carried out on the preselected data items within the full dataset. A second level catalog maintains the list of existing first level catalogs and the type and kind of data items they are storing.

We established a mathematical model of our indexing technique and show that it considerably speeds up the access to LHCb experiment event data at CERN (European Laboratory for Particle Physics).

1 Introduction

Indexing and data selection in a huge data set having a high index dimensionality is one of the key issue in the domain of data management. Recent papers on the subject address this

Proc.[*] 19th IEEE Symposium on Mass Storage Systems & Tenth Goddard Conf. on Mass Storage Systems and Technologies Univ.[†] of Maryland, College Park, Maryland, USA, April 2002, 181-192 problem in specific cases such as spatial databases [6, 5, 7], similarity searches [5, 1, 8] or string matching [4] but do not offer global solutions. Moreover, existing methods are outperformed on average by a simple sequential scan when the number of dimensions is larger than approximately ten[13].

On the other hand, the variety of useful selection criteria on a given set of data is far from being infinite. They can usually be reduced to a small number of indexes, say 20 to 30 maximum (which is already a very high dimension). Thus, from all values contained in a data item (tens of thousands in some cases), only this very reduced subset of 20 to 30 values is relevant for the selection criteria.

This property is used to define a new indexing method based on two levels of catalogs. This method greatly speeds up the linear selecting process by replacing scans of the whole data by scans of matching data. Data is efficiently selected using both server side and client side preselections and the power of the SQL language.

Section 2 presents the context of this work, i.e. the LHCb experiment at CERN and its requirements in terms of data indexing and retrieval. Section 3 presents search results in the domain of data indexing and emphasizes their respective strengths and weaknesses. Section 4 presents the proposed indexing schema and shows how it can be used efficiently for data retrieval. Section 5 evaluates the performance of the new indexing method compared to sequential scan[1]. Section 6 draws the conclusions.

2 Context

The work presented here is based on studies made at CERN (European Laboratory for Particle Physics) in the context of the LHCb [10] experiment. We present here briefly the problem and the requirements we had.

2.1 The LHCb experiment

LHCb [10] is the name of one of the future Large Hadron Collider (LHC) experiments. Its primary goal is the study of the so called CP Violation [11]. This physical theory suggests that, in the world of subatomic particles, the image of a particle in a mirror does not behave like the particle itself [9]. One of the fundamental grounds of this effect is the existence of the bottom-quark and its cousin the top-quark. This is precisely this bottom-quark, under the form of the B-meson that the LHCb experiment intends to study. The only way to produce particles like this meson is to collide other high energy particles (accelerated protons in the case of LHC). This collision will produce hundreds of new particles among which the physicists will try to detect B-mesons and to measure their parameters and behavior (especially the way they decay).

[1]Sequential scan is besides our method the only method which, to our knowledge, fulfills our requirements

2.2 Some figures

LHC will let bunches of protons collide every 25 ns, i.e. at a frequency of 40 MHz. Such a collision is called an event and creates lots of particles (some hundreds). The different detectors constituting LHCb are able to detect all created particles and to specify their energy and momentum. The global output is about 1 MB of data per event across 950000 channels. This yields 40 TB of data every second !

Most of this data will not be stored since more than 99,9999% of it is not interesting. Actually, the detector has a four level trigger system that allows a reduction of the data rate from 40 TB/s to 20 MB/s per second, which is two million times less. This factor is due to both a reduction of the event size to the order of 100 KB and to a reduction of the event rate to 200 Hz. Assuming that the LHC will run 24 hours a day and 7 days a week, LHCb will produce an order of 10^{10} events per year, which is one petabyte (1 PB = 10^{15} bytes) in term of data size.

Table 1 summarizes the figures concerning the data being saved, indexed and later retrieved by physicists for analysis.

Size of a data item	100 to 200 KB
Nb of items	10^9 to 10^{10} per year
Global size of the database	$\sim 10^{15}$ bytes = 1 PB per year
Data items input rate	200 Hz
Data input rate	20 to 40 MB/s

Table 1: Figures concerning LHCb data

2.3 Data selections

The analysis by physicists of the LHCb data is rather specific. It is mainly based on an iterative process consisting in selecting some data items (typically in the order of 10^6) with rather complicated selection criteria, downloading the items, running some computation on them and modifying the selection criteria. A criterion may for example make use of the energy of the event, of the types of particles involved or of the number of decays. The number of iterations is rather small (in the order of 10) but the selection of the data still appears to be the key of the physics analysis.

Another issue is the number of indexes that a given criterion uses. This is typically in the range of 10 to 30 parameters with a mixture of numeric, boolean and strings. These indexes are not always the same for all criteria but a few number of criterion types can be defined (less than 10) for which the set of parameters is fixed. Due to the high dimensionality of the event data (10 to 30 indexes), up to now, at CERN, the only data selection algorithm was a linear scan of the whole dataset.

3 Related Work

There are not many research approaches addressing the issue of indexing generic data in a high dimension space. Weber et al [13] show that there exists a dimension over which any algorithm is outperformed by a sequential scan. Experimentations show that the sequential scan outperforms the best known algorithms such as X-trees[2] and SR-Trees[5] for even a moderate dimensionality (i.e. ~ 10).

These two algorithms are based on data partitioning methods. The ancestor of the data partitioning method is the R-tree method [3] which was further developed under the form of R^*-Trees [6]. However, these data partitioning methods perform poorly as dimensionality increases due to large overlaps in the partitions they define. This is due to exponential increase of the volume of a partition when the number of dimensions grows.

The SR-Tree method tries to overcome this problem by defining a new partition schema, where regions are defined as an intersection of a sphere and a rectangle. The X-Tree method, on the other side tries to introduce a new organization of the partition tree which uses a split-algorithm minimizing overlaps. The results are good at low and moderate dimensions but are outperformed by a sequential scan for dimensions larger than 10.

4 A two level indexing schema

The aim of our proposed schema is to allow most of the selection to be carried out using catalogs (tag collections) that contain only a part of the data items and, for each item, only a subset of its values (a tag). Several catalogs are built, each for a different type of query. This allows to perform a very efficient preselection of the items before accessing the real data items.

4.1 Tags

A tag is a subset of a data item comprising several parameters plus a pointer on this data item. A pointer is simply the information needed to find and retrieve the data item, be it a regular pointer (memory address), a file name, an URL or something else.

A tag contains the few values (also called parameters) of the data item that are used as selection criteria. For a given criterion, or even a given type of criterion, the number of tag values is small (10 to 30) which results in a tag size of 10 to 200 bytes. For example, in the case of some physics events, one may want to include in the tag the energy, the nature of the event and the number of particles involved.

Several types of tags can be defined, with different sizes and contents, even for the same data item. Different tags will point to different subsets of the data items and correspond to different criteria.

Tags are small, well structured objects that can be easily stored in a relational database.

Thus, they can be searched using the power of SQL-like languages. The storage of tags in a relational database is trivial : each type of tag is stored in a different table, whose columns are the different values included in the tag plus one for the pointer to the real data item. The data item itself does not need to be part of a database.

Tags will be used to make preselections without loading the data items, which reduces the amount of loaded data by a factor of 10^3 in the case of LHCb.

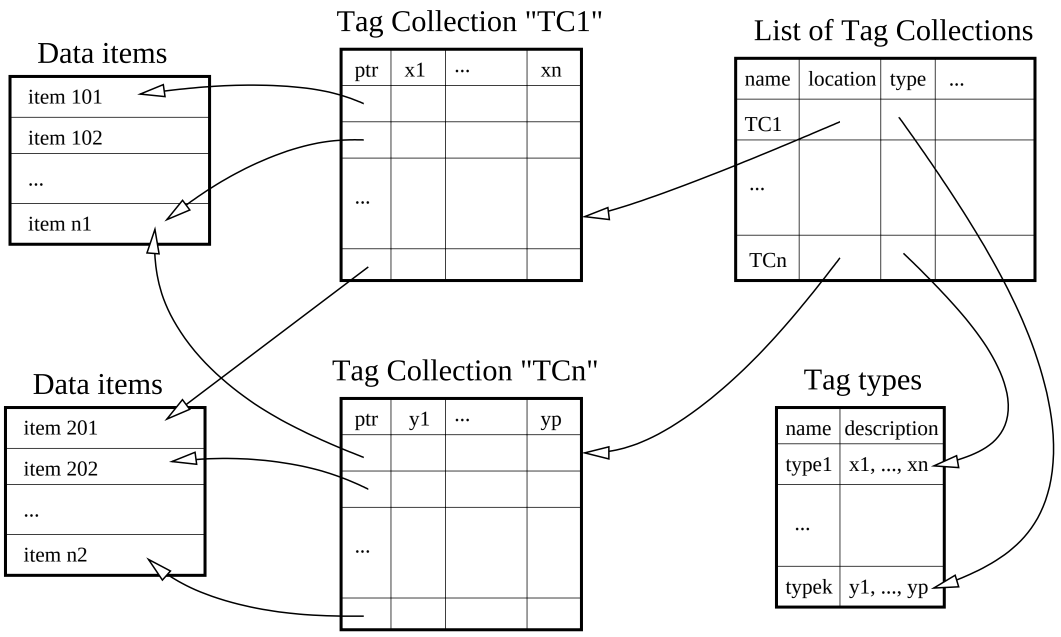

Figure 1: Structure of the tag collections

4.2 Tag collections

As explained above, tags are subsets of data items. A tag collection is a set of tags, all of the same type. It corresponds to a set of data items but with only a subset of the data items values. The values themselves fulfill some criteria, such as being in the interval between a minimal and a maximal value. Thus, two different tag collections may correspond to two different subsets of data items, even if they use the same set of values (type of tags). These subsets may of course overlap.

Tag collections are stored in a relational database as a table, where each line is a tag and columns correspond to the values contained in the tags (Fig. 1). The tag collections form a list of tag collections, each with each associated tag type.

Since tag collections only contain tags for a given subset of the data items, they act as a first preselection on data. For example, in the LHCb experiment, a collection of tags is in the order of 10^5 smaller than the database, i.e. around 10 GB. A factor 10^3 is due to the tag size (section 4.1) and another 10^2 factor comes from the fact that, on average, less than

1% of the data items have a tag in a given collection, i.e. tags whose values are within the predefined ranges associated to that collection. Thus, a collection has typically 10^7 to 10^8 entries.

Collections can be defined by any user or group of users who wants to be able to use a new selection criterion. The creation of a new collection may either require a scan of the full set of data items or is extracted as a subset from another collection. Scanning the full set of data items is time consuming but will be far less frequent than the selection of data items. We expect that there will only be 10 to 20 “base” tag collections in LHCb. All other collections will be subsets of base tag collections.

4.3 Selection process

By selecting tags in tag collections instead of selecting directly data items, there is an immediate gain. Only data items of interest are loaded instead of loading all items for each selection.

This is specially interesting in the case that data items are not located in a database but in regular files and loading a data item requires accessing a file containing many items. With a pointer to the data item within the file, the item of interest is directly accessed and loaded. Such a strategy of storing the actual data in regular files may actually be applied to many problems since database management systems cannot handle petabytes of data easily.

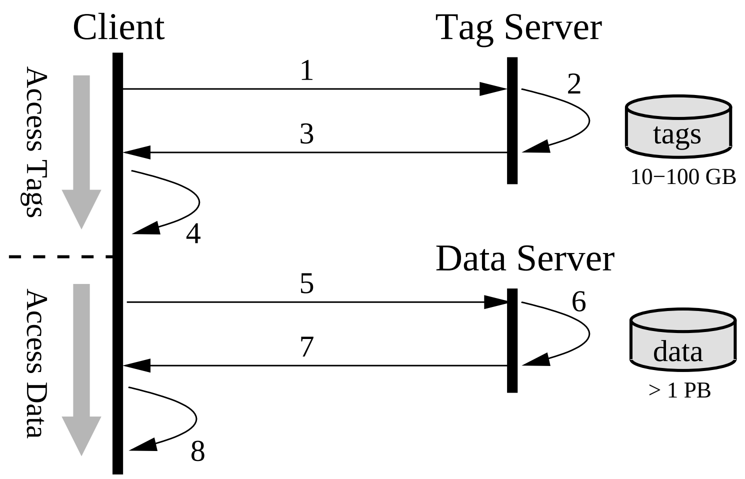

Figure 2: Data selection process

Furthermore, the 2-level indexing schema presented here offers a very powerful and flexible way of applying various preselections allowing to reduce both the amount of accessed data and the network traffic. The complete selection process is shown in Figure 2.

The steps involved in the selection process are the following :

1. The client selects a tag collection and sends a SQL query to be applied on tags from

this collection. The usage of a specific collection is actually a first preselection made by the physicist.

2. The query is processed on the server side.
3. Only matching tags are sent back. This minimizes the network load.
4. A second selection may be applied on the client side, for example for queries that cannot be formulated in SQL and which require a procedural language such as C^{++}.
5. Once the selection on tags is complete, requests for the corresponding data items are sent to the data server.
6. Data items are retrieved (from files, in the LHCb experiment).
7. Retrieved data items are sent to the client.
8. A last selection may be performed on the full data items, in the case that some information was missing in the tags which did not allow to perform this more narrow selection in a previous step.

Note that the separation between client and servers (a tag server and a data item server) on Figure 2 allows for example to replicate the tag server while keeping the data item server at a single location .

5 Performance evaluation

Let us evaluate the performance of our indexing schema. It is hard to compare our proposed schema to existing indexing techniques since we don't know of other indexing techniques except linear scanning which are able to meet our requirements.

Two of the main high-dimensionality indexing schemas are X-trees[2] and VA-file[12]. The X-Tree method is outperformed by a sequential scan for a dimension exceeding 6 (see [13]) and VA-files are only applicable to similarity-search queries. Thus, we only compare our performances with the performances of the sequential scan method.

5.1 Some approximations

Let us make some simplifications and approximations in order to create a model of the proposed indexing schema.

Type of data : We only consider one data type (integers). The cost of a comparison between two values is therefore always the same. This is not the case in real life, where data typically consist of numbers, booleans and strings. However, it is always possible to express the comparison cost of a data item type as a factor of a single integer comparison.

Optimizations : No optimization of the query processing on tag collections are taken into account. This means that tag collections are searched sequentially. Thus, the gain obtained by querying tag collections is really the minimum we can expect from the new schema.

Data transfers : No optimization of data transfers are taken into account. Especially, we do not consider pipelined schema where the data transfer of a given item could be realized during the computation of the previous one.

Size of tag collections : For the performance analysis we consider only a single tag collection with a fixed number of tags. The number of tags and the size of the tags may be considered as an average among the different values of a real life example.

Complex queries : We do not take into account complex queries that could only be processed by a dedicated program. In other words, step 4 of the selection process (Figure 2) does not occur here.

5.2 Theoretical model

Let us adopt the following notations :

N	is the number of items in the whole database;
n	is the average number of items in a given tag collection;
D	is the number of values in a data item i.e. its dimension;
d	is the number of values in a tag i.e. the dimension of the tag; we assume that all these values are tested;
d'	is the number of values that are not contained in the tag but still need to be tested (step 8 in Figure 2);
T_{lat}	is the latency of the network which is used to transfer the data;
T_{tr}	is the time used to transfer one value through the network; in second per value;
T_{IO}	is the time needed to load one value from disk into the memory;
T_{CPU}	is the time to compute one value, i.e. to compare it with another value;
q	is the number of matching tags for the query we are dealing with;
t_{seq}	is the duration of the query using a sequential scan;
t_{tag}	is the duration of the query using the new indexing schema.

In the case of a sequential scan, the time needed to process a query is simply the time needed for querying one data item multiplied by N. Each data item is read from disk, transfered through the network and processed.

$$t_{seq} = N\left(T_{lat} + D\left(T_{tr} + T_{IO}\right) + \left(d + d'\right) T_{CPU}\right)$$

It is independent of the size of the result.

The time needed to process a query using the new indexing schema is slightly more complicated to compute. Using the architecture depicted in Figure 2, we can divide it into two parts : the duration t_1 of the query on tags and the duration t_2 of the query on data items. The query on tags is carried out on the server. Matching tags are transfered to the client. The query on data items is similar to the sequential scan method.

$$t_{tag} = t_1 + t_2$$

$$t_1 = n\,(d\,T_{IO} + d\,T_{CPU}) + q\,(T_{lat} + d\,T_{tr})$$
$$t_2 = q\,(T_{lat} + D\,(T_{IO} + T_{tr}) + d'\,T_{CPU})$$

Finally :

$$t_{seq} = N\,T_{lat} + N\,D\,(T_{IO} + T_{tr}) + N\,(d + d')\,T_{CPU} \tag{1}$$

$$t_{tag} = 2\,q\,T_{lat} + (n\,d + q\,D)\,T_{IO} + q\,(d + D)\,T_{tr} + (n\,d + q\,d')\,T_{CPU} \tag{2}$$

The query duration is dependent on the number q of matching tags. Note that the assumption that tags are transfered one by one to the client corresponds to the worst case. This could be improved by sending tags by groups.

5.3 Interpretation

The terms in equations (1) and (2) can be divided into three parts : processing time (T_{CPU}), network transfer time (T_{lat} and T_{tr}) and data retrieval time (T_{IO}). Let us consider them separately here.

Processing time : the processing time ratio between tag collection access and the default sequential scan is :

$$r_{CPU} = \frac{n\,d + q\,d'}{N\,(d + d')} = \alpha\,\frac{d + \gamma d'}{d + d'} \tag{3}$$
$$where \qquad \alpha = \frac{n}{N} \qquad \gamma = \frac{q}{n}$$

Since $\gamma \leq 1$ (comes from $q \leq n$), we can be sure that $r_{CPU} \leq \alpha$. This demonstrates that the CPU time ratio is less than (but of the order of) the ratio between the number of tags in a collection and the number of data items.

Network transfer time : the network transfer time ratio between tag collection access and the default sequential scan is :

$$\begin{aligned} r_{NET} &= \frac{2\,q\,T_{lat} + q\,(d + D)\,T_{tr}}{N\,T_{lat} + N\,D\,T_{tr}} \\ &= \frac{q}{N}\,\frac{2\,T_{lat} + (d + D)\,T_{tr}}{T_{lat} + D\,T_{tr}} \end{aligned}$$

Since $d \leq D$, we finally have :

$$\begin{aligned} r_{NET} &\leq 2\,\frac{q}{N} \\ r_{NET} &\leq 2\,\alpha\,\gamma \end{aligned} \tag{4}$$
$$where \qquad \alpha = \frac{n}{N} \qquad \gamma = \frac{q}{n}$$

Since $\gamma \leq 1$, the network transfer ratio is at least of the order of the ratio between the number of tags in a collection and the number of data items. In practice, we even have $\gamma \ll 1$ (we foresee $\gamma \sim 10^{-2}$ for LHCb) and thus $r_{idle} \ll \alpha$.

Data retrieval time : the data retrieval time ratio between tag collection access and the default sequential scan is :

$$r_{DR} = \frac{n\,d + q\,D}{N\,D} = \alpha\,(\beta + \gamma) \tag{5}$$
$$where \qquad \alpha = \frac{n}{N} \qquad \beta = \frac{d}{D} \qquad and \qquad \gamma = \frac{q}{n}$$

Usually, $\beta \ll 1$ and $\gamma \ll 1$. Thus $r_{DR} \ll \alpha$. This means that, in respect to data retrieval time, we gain far more than just the gain obtained by the preselection on data items.

Let us estimate γ. By definition, γ is the proportion of matching tags in a tag collection for a given query. Let us consider a very simple case where every part of the query is a comparison and is fulfilled by half of the items. In addition, let us suppose that the data is uniformly distributed. This leads to :

$$\gamma = \frac{1}{2^d} \qquad and \qquad \frac{r_{DR}}{\alpha} = \frac{d}{D} + \frac{1}{2^d}$$

Figure 3 gives the behavior of this ratio against the dimension d for different values of D.

Roughly, $\frac{r_{DR}}{\alpha}$ goes down from 1 to a minimum for dimensions between 0 and $d_m \sim 8$ and linearly goes up afterwards until it reaches 1 again for dimension D. Clearly, we can approximate r_{DR} by $\alpha\frac{d}{D}$ if $d \geq d_m$. This is exactly our goal since the data retrieval time becomes proportional to the loading time of the tags.

For the LHCb experiment, the dimension of a data item is $D \sim 20000$. The minimum I/O time is reached for $d \sim 18$ and $r_{DR} < \frac{\alpha}{1000}$.

6 Conclusions

We presented a new way of indexing and selecting data in huge datasets having a high index dimensionality. The method avoids linear scanning of the whole data set. Only a minimal set of data is scanned, namely the values stored in tag collections. The selected tags point to the data items that are then retrieved for applying a more narrow selection.

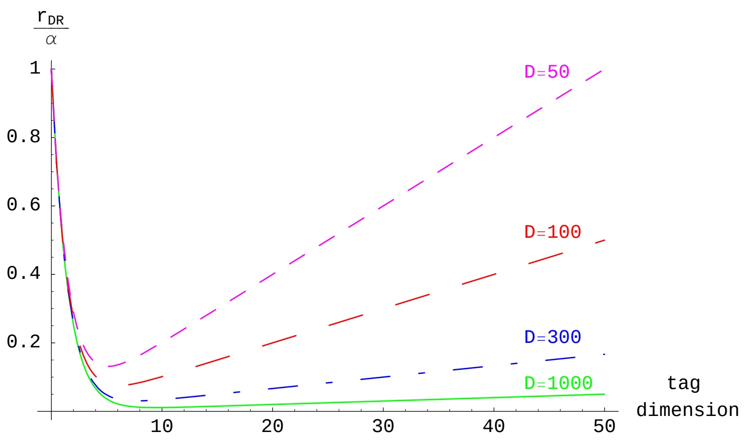

Figure 3: Evolution of a majoration of the data retrieval ratio divided by α in function of the dimension of the tag

By scanning tags in tag collections instead of a flat scan of all data items, the minimal gain is proportional to the ratio between the number of data items and the number of tags within the selected tag collections. In many cases, the effective gain is the minimal gain multiplied by the ratio of the dimension of data items and the dimension of tags.

The proposed data items selection and retrieval schema was implemented at CERN, in the context of the LHCb experiment and seems very promising. No enhancements have been tested at this time but an implementation of a computer assisted parallelization is planned.

References

[1] C. C. Aggarwal and P. S. Yu. The IGrid index: reversing the dimensionality curse for similarity indexing in high dimensional space. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 119–129, August 2000.

[2] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree: an index structure for high-dimensional data. In *VLDB'96, Proc. of 22th International Conference on Very Large Data Bases*, pages 28–39, 1996.

[3] A. Guttman. R-trees : A dynamic indexing structure for spatial searching. In *SIGMOD'84*, pages 47–57, 1984.

[4] H. V. Jagadish, N. Koudas, and D. Srivastava. On effective multi-dimensional indexing for strings. In *Proc. 2000 ACM SIGMOD on Management of data*, pages 403–414, May 2000.

[5] N. Katayama and S. Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 369–380, May 1997.

[6] B. S. N. Beckmann, H-P. Kriegel R. Schneider. The R*-tree : An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD'90*, pages 322–331, 1990.

[7] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The subspace coding method: a new indexing scheme for high-dimensional data. In *Proceedings of the ninth international conference on Information knowledge management CIKM 2000*, November 2000.

[8] K.-T. Song, H.-J. Nam, and J.-W. Chang. A cell-based index structure for similarity search in high-dimensional feature spaces. In *Proceedings of the 16th ACM SAC2001 symposium on on Applied computing*, pages 264–268, March 2001.

[9] C. web pages. A matter of symmetry. URL : http://lhcb-public.web.cern.ch/lhcb-public/html/symmetry.htm.

[10] C. web pages. Experiments in B-physics. URL : http://lhcb-public.web.cern.ch/lhcb-public/html/bphysicsexpts.htm.

[11] C. web pages. What is CP-violation? URL : http://lhcb-public.web.cern.ch/lhcb-public/html/introduction.htm.

[12] R. Weber and S. Blott. An approximation-based data structure for similarity search. Technical Report 24, ESPRIT project HERMES (no. 9141), October 1997. Available at http://www-dbs.ethz.ch/∼weber/paper/HTR24.ps.

[13] R. Weber, H.-J. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In A. Gupta, O. Shmueli, and J. Widom, editors, *VLDB'98, Proc. of 24th International Conference on Very Large Data Bases*, pages 194–205. Morgan Kaufmann, 1998.