ObjectCube: A Novel Data Model for Multi-Dimensional Media Browsing

Grímur Tómasson, Hlynur Sigurþórsson,
Björn Þór Jónsson, Laurent Amsaleg

School of Computer Science
Reykjavík University
January 2012

TECHNICAL REPORT / RUTR-CS12002

ISSN 1670-5777

ObjectCube: A Novel Data Model for Multi-Dimensional Media Browsing

Grímur Tómasson*, Hlynur Sigurþórsson*,
Björn Þór Jónsson*, Laurent Amsaleg†

Technical Report RUTR-CS12002, January 2012

Abstract: Personal collections of digital media have been growing ever larger, leaving users overwhelmed with data. Learning from the resounding success of multi-dimensional analysis in the business intelligence community, we propose a novel multi-dimensional model for media browsing, called ObjectCube. We describe the data model in detail, demonstrate its performance using a prototype media server applied to large collections of media, and show that the ObjectCube data model enables many interesting and useful browsing scenarios that are not possible with current media browsers.

Keywords: Multimedia; Multi-Dimensional Analysis; Performance Measurements; ObjectCube.

This work is partially supported by Icelandic Research Fund Grant 070005023.

* Reykjavik University, Menntavegi 1, IS-101 Reykjavík, Iceland. bjorn@ru.is
† IRISA–CNRS, Campus de Beaulieu, 35042 Rennes, France. laurent.amsaleg@irisa.fr

ObjectCube: Nýtt gagnalíkan fyrir margvíða skoðun margmiðlunargagna

Grímur Tómasson, Hlynur Sigurþórsson,
Björn Þór Jónsson, Laurent Amsaleg

Tækniskýrsla RUTR-CS12002, Janúar 2012

Útdráttur: Magn af margmiðlunargœgnum í eigu einstaklinga hefur vaxið stœðugt að undanfœrnu að því marki að notendur missa yfirsýn yfir sœfnin sín. Við lítum til viðskiptagreindar, þar sem margvíð greining gagna hefur rutt sér til rúms, og setjum fram nýtt margvítt gagnalíkan fyrir skoðun á margmiðlunargœgnum, sem við kœllum ObjectCube. Við lýsum gagnalíkaninu ítarlega, sýnum að hægt er að útfæra það á hraðvirkan hátt með mælingum á frumgerð að margmiðlunarþjóni sem við beitum á stór gagnasœfn, og sýnum að ObjectCube gagnalíkanið gerir mœrg áhugaverð og gagnlegt skoðunartilvik mœguleg.

Lykilorð: Margmiðlunargögn; Margvíð greining; Afkastamælingar; ObjectCube.

Contents

1 Introduction

Since the introduction of personal computers, and digital recording devices in particular, personal collections of digital media have been growing ever larger. A typical personal computer now contains a multitude of files containing various documents, photos, videos, and music. Keeping track of the location and contents of all these files is turning into a major headache for most people and users find it increasingly difficult to organize and retrieve the contents of their computer. It is therefore increasingly important to provide effective tools for browsing personal media collections.

1.1 Current Media Browsers

The current set of browsers that are shipped with computers, and in fact most media browsers, are geared towards helping users organize their documents in hierarchies of folders. These browsers essentially provide a GUI for the physical layout of files and documents on disks. While this is both simple and familiar, these browsers can be of limited use as soon as we forget where our documents are located.

Assigning a unique location for a particular file can be very counter-intuitive. Where, for example, should one store a particular photo? The choices may include: in a folder associated with the time the image was taken; in a folder associated with the event captured by the image; or in a folder associated with a key person in the image. In essence, there is no *logical* reason to prevent files being found in multiple locations. Instead, however, the alignment of current browsers with the underlying file system structure has many *practical* reasons—with simplicity perhaps the key reason.

The search capabilities of file browsers, based on file names and/or file contents, can help in some limited cases. Furthermore, tags can be attached to documents and used to find documents. Unfortunately, however, tags are treated very poorly as there are no means for structuring sets of tags in any meaningful way, e.g., by grouping tags into different concepts or by defining relationships between tags. Today, users are therefore unable to see the forest for the trees.

1.2 An Analogy from Business Intelligence

Helping users overwhelmed with data has long been studied in the database community. Traditionally, database vendors provided users with tools to get back specific data via SQL-queries. Typically, users knew what data they were looking for and simply retrieved the actual values. For users wishing to *understand* the data, however, by discovering trends and patterns, or simply via an overview of key figures in the ever expanding databases, using this model was particularly painful; specifying SQL queries was very difficult, sometimes even impossible, and the execution cost enormous.

It therefore became crucial to invent an adequate framework to facilitate such on-line analytical processing (OLAP), enabling decision support and business intelligence applications. In order to help users turn their *data into information*, a formal data model was created, namely the multi-dimensional analysis (MDA) model, along with the appropriate set of server and client tools.

The MDA model introduced two key concepts that revolutionized users' perception of data. These two concepts are *dimensions*, including hierarchies, used for specifying interesting sets of data and *facts*, or numerical attributes, which are aggregated for an easy-to-understand view of the data of interest. The MDA model has proven to work extremely well in practice and has largely been accepted as the means for understanding huge (terabyte-sized) data sets [CM99].

1.3 Multi-Dimensional Media Browsing

The MDA model was instrumental in helping users make sense of vast amounts of numerical data. The model is implemented through *browsers* that help users discover their data collections without bothering them with location. These browsers are of great practical interest because they focus on the value of data items as well as on the relationships that exist between data, abstracting physical considerations.

Our goal is to apply the concepts of the MDA model to media browsing. This allows us to define in a very clear manner the concepts according to which media items can be grouped, the way tags attached to media files are organized, typically in well-defined hierarchies, or the way users switch from one set of documents to the other when navigating between concepts, hierarchies, or levels of detail. While the MDA model is not immediately suited to media browsing, we believe that the similarity of the problems is significant enough to use the concepts of that model as the foundation of a powerful and flexible solution to media browsing.

1.4 Contributions

In this paper we therefore define a new multi-dimensional model for media browsing, called ObjectCube, based on the MDA model used in OLAP applications. In Section 2 we describe the ObjectCube model and how it manages complex and interesting media browsing scenarios.

A key consideration for any data model is whether it can be implemented efficiently—whether queries accessing large amounts of data are fast enough for practical use. In Section 3 we therefore describe a prototype media server implementing the model and evaluate its performance in a realistic context, using three different underlying data-stores and image collections containing up to one million photos.

Of course, the key strength of a data model lies in the browsing scenarios it enables. In Section 4 we consider a detailed browsing scenario and its realization in the ObjectCube model. We also describe related projects and models and discuss why they fail with the proposed scenario.

ObjectCube	**MDA**	**Correspondence**
Object	Fact	Approximate
Tag	Member	High
Tag-set	Dimension	High
Hierarchy	Hierarchy	High
Hypercube	Hypercube	High
Cell	Cell	Approximate
Filter	Selection	Approximate
State	Subcube	Approximate

Table 1: Comparison of ObjectCube and MDA concepts.

In parallel, we have been working on a graphical browsing interface suitable for the extensive user studies that will be necessary for this project; the initial user satisfaction results are presented elsewhere (citation removed for double-blind reviewing). In this paper, however, the focus is squarely on the underlying data model and its efficiency, which are worthy research contributions in their own right.

2 The ObjectCube Model

In this section, we define the ObjectCube model, which is a generic multi-dimensional analysis model for media files. The model is based on the MDA concepts which have been successfully used in OLAP applications to view and analyze data. The remainder of this section develops the ObjectCube concepts, contrasting them with corresponding MDA concepts; the correspondence is summarized in Table 1.

2.1 Core Concepts

In this section we describe the two core concepts of *objects* and *tags*, which apply to the media items themselves.

Object: An object is any entity that a user is interested in storing information about, typically a media file of some sort. Objects correspond to *facts* in MDA, as both represent information users are interested in analysing and both have associated meta-data that describe them further. Unlike facts, however, objects are typically quite complex, making operations such as aggregation difficult.

Tag: A tag is any meta-data that can be associated with objects. There is no limitation on how many objects a tag can be associated with, nor are there limitations on how many tags can be associated with a single object. Tags correspond closely to *members* in MDA. Both are associated with the objects/facts, providing additional information about them.

2.2 Multi-Dimensional Concepts

In this section we define four concepts that concern categorization and grouping of data, and are at the heart of the multi-dimensional nature of our model.

Tag-set: A tag-set is a set of tags that the user perceives to be related; we can think of the tag-set as a category of tags representing a concept. In a tag-set instance named 'People' the tags can, e.g., be names of people or names of subcategories, say, 'Children' or 'Class Mates'. Tag-sets are mathematical sets, since the elements (tags) are distinct and the order of the elements is irrelevant.

Tag-set are very similar to *dimensions* in MDA. In both cases the user perceives the members/tags as being strongly related to each other, and in both cases hierarchies can be built to organize their contents.

Note that the tag-sets may differ significantly in the method used for their generation. Some tag-sets may be entirely user generated, e.g., an 'Event' tag-set. Others may be based purely on the meta-data associated with each object, e.g., a 'Creation Date' tag-set. Yet others may partially use automated content analysis, e.g., face generation for the 'People' tag-set. Any implementation of the ObjectCube model must therefore support automated media analysis methods.

Hierarchy: A hierarchy is a tree structure that adds structure and order to a subset of the tags of a tag-set. More informally, it serves to categorize some of the tags of a tag-set. A hierarchy is derived from a single tag-set and only contains tags from that tag-set. Each tag-set, however, may have zero, one, or more associated hierarchies. The 'People' tag-set, e.g., could have one hierarchy called 'Friends' and another called 'Family', which would typically be largely disjoint but might share some tags.

While tag-sets are not ordered, the vertices of a hierarchy are explicitly ordered. Hierarchies can therefore be used to enforce the order of tags and create a tag sequence, e.g., listing the months of the year in their natural order.

The hierarchy concept is highly similar to the corresponding concept from MDA. Both can represent either level or value based hierarchies and in both cases a vertex is the aggregation of its children. In MDA aggregation is typically based on mathematical functions, such as sum or average, while in ObjectCube aggregation takes some form of grouping.

A node in a hierarchy may optionally have a title that applies to its children, called a *child category title*; the children are then instances of the category that the title names. To use the example above, 'Month' could be a child category title, while the months themselves are the children. A minor difference between the two models is thus that in MDA a column name supplies a level name in a hierarchy, but in ObjectCube the child category name is applied to a hierarchy node and only applies to the children of that node.

Hypercube: A hypercube is created by selecting and storing information about one or more tag-sets, or hierarchies, which the user wishes to browse her objects by.

In MDA implementations, the hypercube is typically stored in a specialized data structure for efficiency. This data structure stores both base facts and pre-calculated aggregations of facts for higher levels of the hierarchies. Due to the differences between objects and facts, however, it is possible that only base data can be stored for the ObjectCube model, and that the hypercube will therefore only be conceptual.

Cell: A cell in the hypercube is the intersection of a single tag from each of the dimensions (tag-sets or hierarchies) in a hypercube. A cell can contain zero or more images, unlike the MDA model which simply aggregates all the facts corresponding to each cell.

2.3 Retrieval Concepts

So far, we have defined concepts that describe and organize media objects, but now we turn to the retrieval of the objects. Unlike typical search applications, the retrieval is based on browsing, where the retrieved object sets are defined incrementally, based on the user's interest. In the ObjectCube model, different *filters* can be applied to the various tag-sets or hierarchies, resulting in a *browsing state*. We now define these concepts in more detail.

Filter: A filter is a constraint describing a sub-set of objects that the user wished to retrieve. Each filter applies to a single tag-set, but many filters may be applied to the same tag-set. All objects that are associated with tags that satisfy the constraint of a filter, are said to pass through the filter.

Note that a filter can be applied to any dimension, regardless of whether that dimension is visible in the user interface or not. Furthermore, a filter continues to restrict the retrieval until it is explicitly revoked by the user. A browsing session thus consists of applying filters and retrieving the objects that pass through all the applied filters.

The filter concept corresponds to two MDA concepts. First, it serves the same purpose as *selection*, as both can be used to define a filter to restrict the data retrieved. Second, the filter also corresponds to the *page dimension* concept, as both can be used to restrict the data being retrieved in a dimension not in the cube.

There are three different filter variants:

Tag Filter: The tag filter is a filter that selects a single tag, which must exist in the tag-set being filtered. It is used to retrieve only objects associated with that particular tag.

Range Filter: A range filter defines a value range by two boundary values, where both boundary values are included in the range. The boundary values themselves need not exist as tags in the tag-set the filter is applied to. Since a filter only applies to a single tag-set, only tags from that tag-set can be considered to be within the range. A range filter is used to retrieve objects that are associated with one or more tags that fall within the range of the filter.

Hierarchical Filter: A hierarchical filter selects a single node in a hierarchy. The entire sub-hierarchy of that node is said to pass through the hierarchy filter. A hierarchical filter on the root of a hierarchy thus returns the entire hierarchy.

Browsing State: As mentioned above, a browsing session consists of applying filters and retrieving the objects that pass through all the applied filters. The browsing state therefore contains information about filters, tags or sub-hierarchies that pass through the filters, and objects that are associated with these.

Note that tags and sub-hierarchies are included even when there are no corresponding objects, as their existence is highly useful information for the user. Objects are only returned, however, if they pass through *all* the applied filters. It is sufficient that one tag that the object is associated with passes through each filter, and it is possible that different tags may pass through different filters.

Informally, we can think of each filter as selecting a sub-set of the objects. The objects in the browsing state are then selected by the intersection of the sets passing through the filters. Tag-sets with no filters, however, are excluded from the intersection. If no filters are in effect, the browsing state therefore includes all objects in the system.

The browsing state corresponds loosely to a *sub-cube* of the hypercube in the MDA model, as the browsing state contains enough data to build such a sub-cube. Since the hypercube cannot be precomputed, as in the MDA model, it is necessary to submit all this information to the user interface.

2.4 Summary

In this section we have defined the ObjectCube model and contrasted it with the well-known MDA model. All their differences stem from the fact that the MDA model is applied to simple facts, which are typically numerical data items, while the ObjectCube model is applied to complex media objects which cannot be as easily manipulated or aggregated. In the following, we first evaluate the efficiency of the model (Section 3) and then compare its effectiveness to the state of the art in media browsing (Section 4) .

3 Experimental Evaluation

We have developed a prototype media server implementing the ObjectCube model, for the purpose of demonstrating the efficiency of the model as well as for the future development of browsing interfaces based on the model; the server will be made publically available. In this section we first describe the overall architecture of the prototype and then describe the experimental setup which

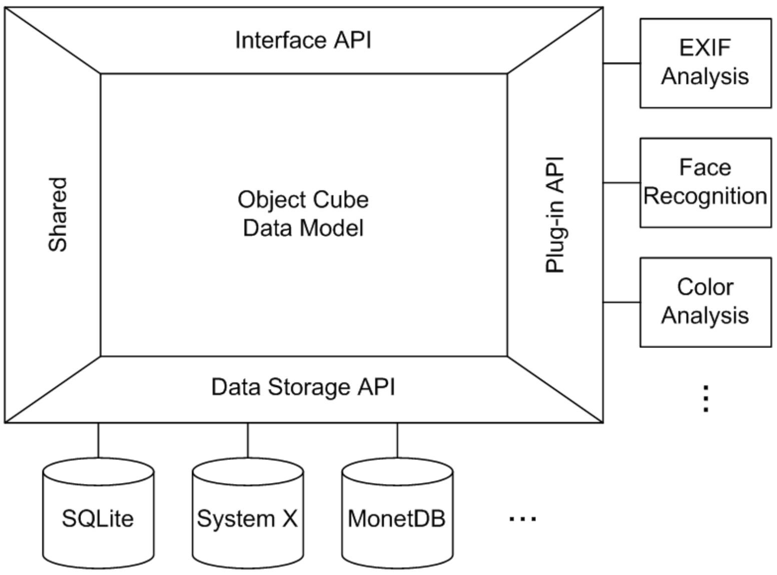

Figure 1: Architecture of the ObjectCube prototype.

applies to all the experiments reported here. We then analyze the results of three experiments, one for each filter type. These experiments were designed as a “stress-test”, i.e., they are designed to showcase any scalability weaknesses of the prototype and/or data-stores. It is also of interest to see how the prototype performs using different data-stores as they have different underlying architectures (row-store vs. column-store), especially since the workload is analytical in nature. We conclude the section by analyzing the performance overhead of the prototype and summarizing the performance results.

3.1 Prototype Architecture

Figure 1 shows the architecture of the ObjectCube prototype. The prototype consists of a central logic module implementing the data model, as well as APIs for data storage, user interface development, and automated media analysis using a *plug-in* architecture (based on Markus Ewald’s concepts, see `http://www.nuclex.org/articles/5-cxx/`). Plug-ins are software components which are called upon to analyze media objects during insertion and generate new tags or attach objects to existing tags. Currently, three plug-ins are implemented that: extract the EXIF meta-data associated with media files; extract faces from photos; and analyze the color composition of photos.

The ObjectCube prototype is written in C++ and makes extensive use of its standard library, as well as the TR1 C++ library extensions. The current implementation, without plugins, consists of ap-

Data Set	photos	Tags	Per Photo
1K	1,000	3,252	13.90
10K	10,000	20,819	13.68
100K	100,000	113,185	13.50
1M	1,000,000	201,277	13.62

Table 2: Statistics for the experimental data sets.

proximately fourty thousand lines, and runs on Mac OS X 10.6 and Ubuntu Linux 10.04. Data-store implementations exist for three different relational database systems: SQLite, a widely deployed public domain row-store; MonetDB, an open-source column-store from CWI, that is known for its focus on performance; and "System X," a widely used commercial row-store. Currently, filters and tags are cached for performance, while objects are retrieved upon request by the user.

3.2 Experimental Setup

In the following we describe our experimental setup, which applies to all the experiments reported in this paper.

Data Sets: In order to examine the scalability we used four data sets of photos downloaded from Flick'r, each one an order of magnitude larger than the other. The bulk of the tags and taggings in this set are automatically generated by the EXIF plug-in; of the individual tags in the collection, about 85% are different time tags. Additionally, we created randomly assigned tags and hierarchies with specific selectivities, for use in the individual experiments. The properties of the photo collections are shown in table 2.

Experimental Platform: The experiments were run on a Dell Precision T5400 Workstation, which was equipped with two Intel E5430 quad core CPU's running at 2.66GHz, a 12MB L2 Cache for each CPU, 4GB of memory, and a 500GB ATA hard drive, 7200 RPM. The operating system was a standard 32bit Ubuntu 10.04,allowing to use only 3.2GB of memory.

Our experiments were performed using a virtual machine, facilitating deployment regardless of the underlying hardware. Although the virtual machine used has been shown to lose approximately 15% of CPU performance and 30% of I/O performance [DAS09], this does not affect our conclusions significantly, as our goal is the comparison of different data-stores and an analysis of the overhead of the prototype.

In all cases, default settings for the three supported data-stores were used and no tuning performed, but all tables were heavily indexed. Recall that MonetDB is a column-store, which has been shown to perform well in analytical workloads, while the other two are traditional row-stores.

Methodology: In our experiments, we focus on the time required to retrieve the browsing state, both tags and objects (the photo location). We have chosen to ignore the time to retrieve the photos themselves, as that time is independent of the method used to retrieve the state. We are interested in all aspects of scalability, i.e., in terms of the data set size, the selectivity of the browsing state, and the complexity of the queries required to retrieve the browsing state. As we observe, different data stores react differently to different scalability aspects.

Each experiment was run for all the data sets, from the smallest to the largest, using a single data-store. To avoid interference and, we restarted the workstation (hence the VM) before each experiment, loaded the relevant tables in memory, ran the experiment once, before running the actual measurement a minute later. This simulates well the long-term operation of the system, where buffers are fully loaded. The experimental workstation did not run any other tasks, and was not connected to a network to avoid interference.

Measures: We focus on the time, measured at a granularity of milliseconds, to retrieve each browsing state. In all cases, we consider retrieval time below one second to be satisfactory; runs taking more than 90 seconds were aborted.

When considering scalability with respect to data set size, we compare the scaling of individual data-stores to linear scaling. For each data store, we assign a value of 100% to the time required for the 1K data set. Then we scale the retrieval time for each of the other data sets with the time for the 1K data set and the relative size of the data set. If the system scales linearly, this will result in a straight and horizontal line, while sub-linear scaling results in a downward slope.

3.3 Experiment I: Tag Filtering

In this experiment, a single tag filter is used to retrieve all photos that have a particular tag associated with them. We created five different tags with selectivities equal to 0.01%, 0.1%, 1% 5% and 10%, and tagged photos randomly in order to create a worst case scenario without any groups of tagged photos. The underlying query to the data store, however, is independent of the selectivity.

Impact of Selectivity: Figure 2 shows the time required to retrieve the browsing state for the 10K data set. The x-axis shows the selectivity (percentage of objects retrieved), while the y-axis shows the retrieval time of the browsing state in milliseconds. Note the logarithmic scale of both axes. Generally, the performance is satisfactory, as the retrieval time is in most cases well under our limit of one second. We observe that System X suffers most from the increased selectivity, while MonetDB is the stand-out performer, in all cases requiring less than 60% of the time required by SQLite.

Figure 3 shows the corresponding results for the 1M data set. As expected, all data-stores perform worse for this data set, which is 100 times larger than the 10K data set. In particular, SQLite appears incapable of handling such a large collection, as it requires 8.6 seconds even for the 0.1% selectivity. As before, MonetDB performs the best, but at 1% selectivity, even MonetDB requires 2.3 seconds to retrieve the browsing state. It should be noted, however, that at this point the browsing state contains 10K objects, which represents far more photos than any user can digest on screen; they would also require much longer time to load from disk.

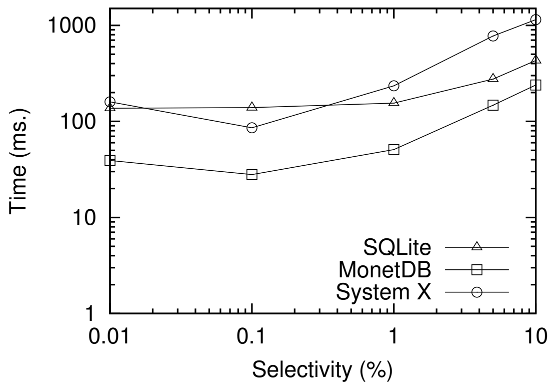

Figure 2: Performance of state retrieval.
(tag filter; 10K set; varying selectivity.)

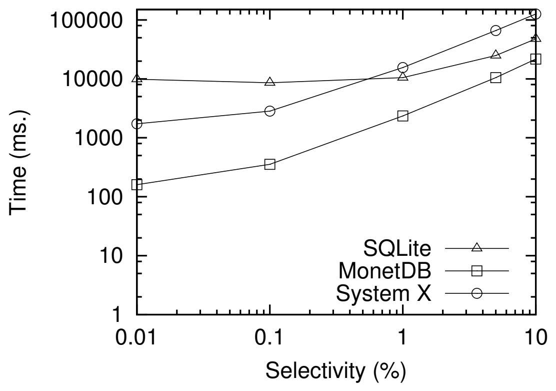

Figure 3: Performance of state retrieval.
(tag filter; 1M set; varying selectivity.)

Impact of Scale: Figure 4 shows the scalability of the data-stores with respect to the size of the data set, for a selectivity of 1%. The x-axis shows the data set, while the y-axis shows the performance relative to the 1K data set, scaled by the size of the data set. Note that linear scaling would result in a horizontal line.

The figure shows that while all three data-stores scale quite well with data set size for this low selectivity, MonetDB scales best in this experiment. Recall that the underlying query to retrieve the browsing state is simple, so this experiment focuses on the impact of the data set size.

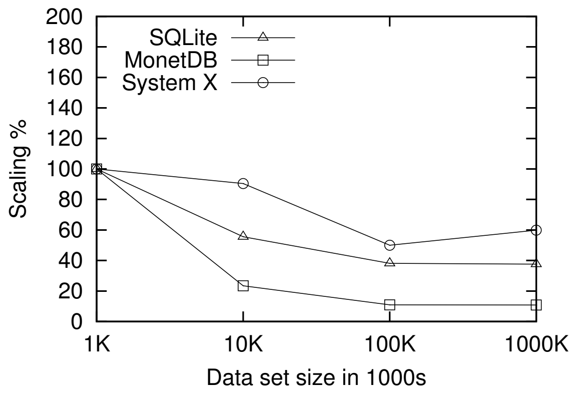

Figure 4: Scalability of state retrieval.
(tag filter; 10% sel.; varying data set size.)

For the 1M collection (not shown), MonetDB continues to show sub-linear scaling due to its column-store architecture.

3.4 Experiment II: Range Filtering

In this experiment we evaluate the performance of our prototype using a single range filter. We use contiguous time ranges in the tag-set representing the creation time of photos, so no data was added or edited for this experiment.

Impact of Selectivity: Figure 5 shows the results for the 10K data set. On the one hand, both MonetDB and System X perform relatively well across the range, although System X requires about two seconds for 10% selectivity. The difference between System X and MonetDB is considerable, however, as MonetDB only needs about 20% of the time used by System X for the largest selectivities. SQLite, on the other hand, performs significantly worse than before.

Figure 6 shows the results for the 1M data set. For this very large set, only MonetDB comes close to matching our definition of usability, and only for small selectivities. As before, System X performs significantly worse for range filters, and SQLite is extremely slow, even for 0.01% selectivity.

Impact of Scale: Figure 7 shows the scalability of the data-stores with respect to the size of the data set, for a selectivity of 1%. The x-axis shows the data set, while the y-axis shows the performance relative to the 1K data set, scaled by the size of the data set. Recall that, with this methodology, linear scaling would result in a horizontal line.

Given the poor performance of SQLite in this experiment, its superlinear scaling is not surprising. The other two data-stores show similar or even better relative scalability than in the tag filter experi-

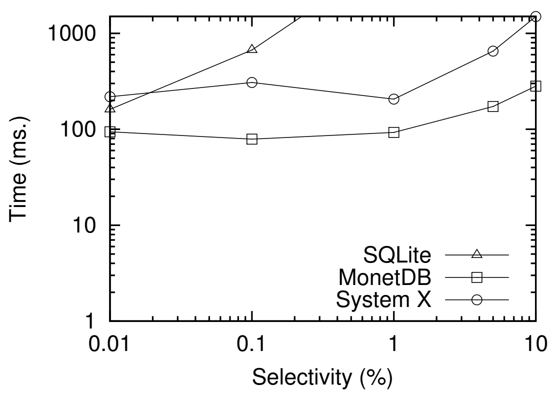

Figure 5: Performance of state retrieval.
(range filter; 10K set; varying selectivity.)

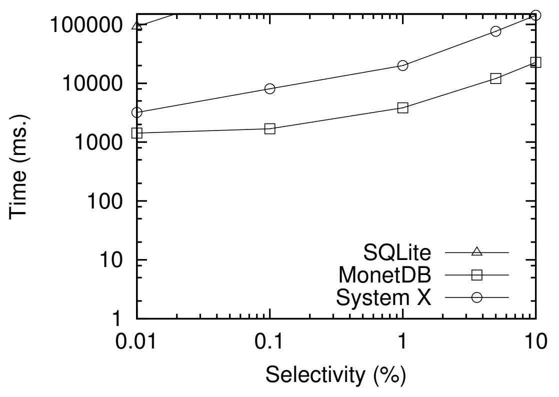

Figure 6: Performance of state retrieval.
(range filter; 1M set; varying selectivity.)

ment, and both are significantly sublinear. For large collections and selectivities, however, this is not sufficient for good performance, as we have seen.

3.5 Experiment III: Hierarchical Filtering

We now evaluate the performance of our prototype using a single hierarchical filter. We have constructed a hierarchy specifically for this purpose, described in the following.

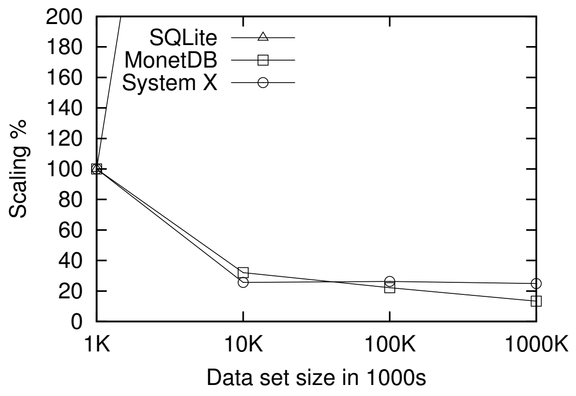

Figure 7: Scalability of state retrieval.
(range filter; 10% sel.; varying data set size.)

Hierarchy Setup: We created a worst case scenario for hierarchical filtering, where all the objects in the system are associated with one or more tags included in the hierarchy. This is quite relevant, since it is reasonable, e.g., that all photos reside in a date hierarchy. We created a new tag-set, populated it with 400 different tags, and then created a four level hierarchy where the root and each child, aside from the leaves, has seven children. This leads to a hierarchy with 400 nodes, where each node encapsulates a single tag from the tag-set. We then associated every photo in each data set with a random node in this hierarchy, so the selectivity percentages vary slightly between the test sets; the detailed selectivities can be seen in Table 3. Note that, unlike previous experiments, the selectivity goes to 100%.

Impact of Selectivity: Figure 8 shows the performance of state retrievals when using hierarchical filtering. For comparison, it also shows the time needed when applying tag and range filtering, which have been discussed earlier. Note that this figure focuses on the performance of the MonetDB datastore, ignoring the two other datastores, as we already demonstrated that MonetDB clearly outperforms both. Overall, the figure shows that the performance of state retrieval is very much related to the selectivity of the filters, and that hierarchical filtering has a rather similar performance to tag and range filtering. We observed that the performance of all the filter types is highly dependent on the number of images in the state retrieved, much more than it depends on the size of the data-set. Fetching a browsing state containing meta-data for 1,374 photos from the 10K dataset takes 328ms while it takes 399ms to fetch the browsing state containing meta-data for 2,023 photos from the 100K dataset (almost 50% more elements increase the time by 20% while the dataset is 10 times larger).

Hier.			**Data Set**		**Rough**
Level	**1K**	**10K**	**100K**	**1M**	**Select.**
1	2	23	302	2,878	0.25%
2	22	184	2,023	20,225	2%
3	143	1,374	14,283	142,903	14%
4	1,000	10,000	100,000	1,000,000	100%

Table 3: Objects in browsing state for each data set in the hierarchical filtering experiment, varying selectivity.

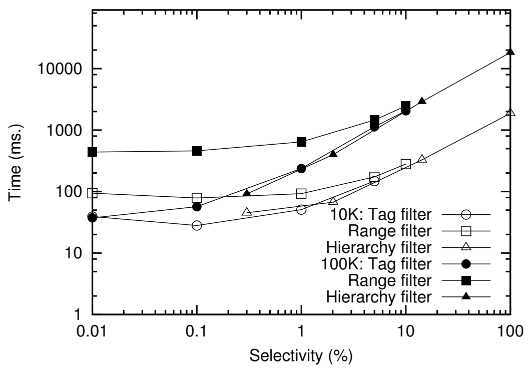

Figure 8: Performance comparison of the different filters.
(MonetDB; 10K and 100K sets; varying selectivity.)

3.6 ObjectCube Overhead

The overriding factor in the performance of ObjectCube is the performance of the data-store queries. We ran extensive experiments, however, to determine the overhead of the ObjectCube prototype and briefly summarize the results here. For a state of 1,000 objects the overhead for layer conversion and adding tags is around 3 ms. The overhead for each browsing state hierarchy is 15–30 ms; we expect to see 1–3 state hierarchies in a typical browsing state. Finally, the overhead of the Python interface is 6–7 ms for every 1,000 objects. The total overhead is thus less than 100 ms for a state of 1,000 objects with three state hierarchies. This overhead is quite acceptable.

3.7 Key Lessons

We note first that MonetDB outperforms the other systems significantly, due to its column-oriented architecture, and can handle quite large data sets. The first key lesson is therefore to select the appropriate data-store; the remaining lessons refer solely to the MonetDB results.

The second key lesson is that the performance of all filter types (tag, range, and hierarchical filters) is primarily dependent on the number of images in the state retrieved, rather than the size of the data-set, and that performance is quite acceptable for moderate to large selectivity ranges.

Returning extremely large browsing states, however, is quite slow. We believe such large browsing states make little sense, however, as they will drown users with so much data that it becomes impossible to properly visualize that many pictures. Therefore, the third key lesson is that it is imperative to develop methods helping users to reduce beforehand the number of objects retrieved in a manner that is perceived as intuitive, seamless, and non-intrusive.

Finally, we must account for the additional time to load and present on screen the selected pictures (or other media objects) from the retrieved browsing state. One potential solution is to pre-calculate aggregates of hypercube cells in some fashion, for example using collages or slide-shows. This is a key subject of our future work.

4 Data Model Expressiveness

In this section we consider a detailed browsing scenario to demonstrate the expressiveness of the ObjectCube model. We have chosen a photo browsing scenario in Section 4.1, for two reasons. First, as mentioned before, we have in parallel been working on a graphical user interface for photo browsing and can therefore support the scenario with an illustration from the interface. Second, most of the advanced browsing work so far has focused on photos, as photos are more amenable to visual presentation than other media forms; in Section 4.2 we discuss alternative approaches in some detail and describe why they fail with this scenario.

4.1 A Photo Browsing Scenario

Assume that a user has tagged the people in her photo collection, partially using the automated face recognition plug-in, and created a hierarchy for her family, with children, parents, and grandparents in different subhierarchies. She has also tagged animals and objects, as well as the location where the photos were taken. Furthermore, all the Exif data has been automatically extracted, as well as color information. The user is sitting down with her children, to show them photos from recent trips.

The user first selects the year 2008 from the ‘creation date’ hierarchy as this was the time of the trip abroad that started the discussion. She wishes to discover all the photos containing the children and the parents and therefore selects the ‘children’ node of the ‘family’ hierarchy on one display axis and the ‘parents’ node of the ‘family’ hierarchy on another. The user is also interested in where the pictures were taken and decides to add the ‘location’ hierarchy as the third display axis. Note that in the ObjectCube model, all these actions require the application of a hierarchical filter; currently four such filters are in effect.

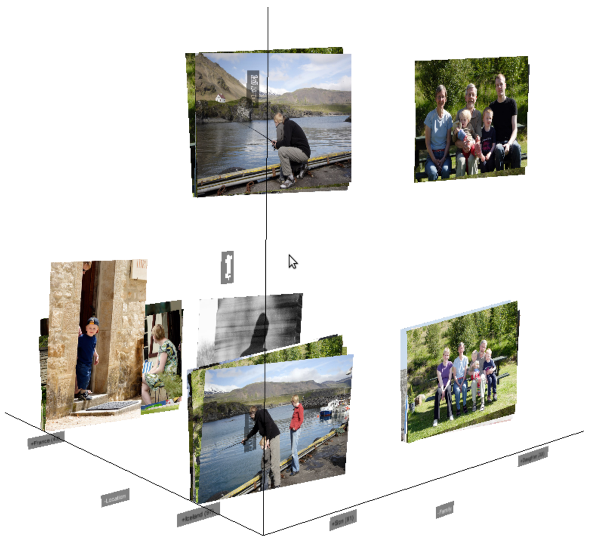

Figure 9: An example three-dimensional browsing state.

The resulting browsing state is shown in Figure 9. On the horizontal axis, photos containing the son are found in the left hand stacks, the daughter in the right hand stacks. On the vertical axis, photos of the mother are found in the bottom stacks, the father in the top stacks. On the 'in' axis, photos from their home country are found in the front stacks, photos from their last trip abroad in the back stacks. Note that the filter on the 'creation date' is not immediately apparent in the figure, but is shown elsewhere.

From the figure, we can observe that photos containing all four will show up in four stacks (no picture, however, can be taken simultaneously in two countries); this is an important feature of the model as the photos belong logically in all these stacks. We can also observe that the father does not appear in any photos abroad, and neither does the daughter; hence there is only one stack abroad. As it turns out, the daughter wasn't born at this time, but since the father was in fact on the trip, the user decides to focus on pictures of parents regardless of the children to see whether there are any photos of the father (who, as it turns out, shot most of the photos). She therefore removes the hierarchical filter on children. Then the user decides to look at more recent photos also and replaces the front axis with the timeline, focusing on the granularity of years. This results in several stacks of photos, one stack per parent, year, and country. The user might then go on to consider animals or objects, rotate the cube for better viewpoints, and so on.

This scenario demonstrates several common operations in the ObjectCube model, namely filtering, drilling down into hierarchies, and pivoting (replacing browsing dimensions). It also shows how well the model uses the screen to indicate why each photo is shown in the result. Of course, it is more difficult to present more than three browsing dimensions at once, but one can imagine color coding, photo size, and photo transparency as additional presentation dimensions. Finally, the scenario illustrates that the model is very well suited to storytelling and discovery, something that we have experienced very vividly in our demonstrations.

4.2 Related Work

It does not take much thought to understand why the simple file and media browsers of today fail completely at handling this scenario that is so natural in the ObjectCube model. These browsers are missing the separation of tags into tag-sets, the relationships between tags in a particular tag-set, and the ability to use these relationships for browsing.[1] Furthermore, when using the limited search capabilities of these browsers, they display the photos in a single-dimensional view, thus completely losing the reason why each photo is present in the result. This is a *dimensionality reduction* of sorts and hampers the discovery of the media collection.

Some research projects have considered media browsing, typically focusing on one media type at a time, and have proposed many interesting methods for browsing. While the literature is far too vast to be all cited here, we describe some interesting techniques. PhotoMesa [Bed01] provides a zoomable interface to multiple directories of images at once, grouping the images from the folders into clusters for maximal use of the screen. With PhotoMesa, however, the browsing scenarios is impossible, as the browser operates solely on the folder structure and only allows filtering by people tags. Harada et al. [HNS$^+$04] proposed a browser focusing on automatically generated event assignments of photos; this method could well be implemented as a plug-in for our prototype. Several researchers have considered photo spreadsheets; in one of the most recent works, Kandel et al. focus on a biological application [KPT$^+$08]. Turning to other media, Knees et al. [KSPW07] propose a visual presentation of music that uses clustering to create ‘islands of music’ which the user can then navigate in 3D, and a survey of video techniques is found in [HaLZM11].

A related approach is that of logical information systems (LIS). The Camelis photo browser uses co-occurrences of tags in images to deduce relationships and uses those relationships to facilitate browsing [Fer07]. Camelis may potentially be set up to return the required set of images, but constructing the query would be very complex. Furthermore, images are presented linearly without categorization, thus losing information about why they are present in the result.

The approach most similar to our approach, however, is that of *faceted search* which has been applied to many domains, including photos [YSLH03, BC09] and audio [DMRS10]. Faceted search uses a single tag-set, but proposes to build multiple hierarchies (or even DAGs) over that tag-set, one for each aspect that could be browsed. These hierarchies are then traversed to interactively narrow the result set, until the user is happy. Item counts or sample queries are typically used to present the result while it is very large; when it is sufficiently small it is presented in a linear fashion. The faceted approach suffers from two main problems. First, the single tag-set is a limitation; although the hierarchies do help users somewhat to disambiguate the different uses of an ambiguous tag, it is

[1] An industrious user can try to get around some of the limitations of current tagging approaches by encoding the hierarchy of friends within the textual tags. Such attempts, however, are labor-intensive and rarely sustainable.

more logical to place distinct tags in different tag-sets. Second, the linear presentation results in the same dimensionality reduction as the simple browser suffer.

Scenique [BC09] is a faceted photo browser that breaks from tradition; it is conceptually the browser most similar to our proposal. Scenique allows image browsing in 3D browsing rooms, where each dimension corresponds to a facet.[2] Scenique can not, however, show different parts of the same hierarchy of different browsing dimensions and in fact it is not clear how it handles the case when constraints are given for 1–2 or 4+ dimensions. We believe that the underlying concepts of the ObjectCube model, in particular the cell and the hypercube, are the key differentiating factor.

5 Conclusion

Personal collections of digital media are growing ever larger, leaving users overwhelmed with data. We have therefore proposed a new multi-dimensional data model for media browsing, based on the highly successful multi-dimensional analysis model from the business intelligence community. We have described the ObjectCube data model in detail, demonstrated its performance using a prototype media server applied to large collections of media, and shown that the ObjectCube data model enables interesting and useful browsing scenarios that are not possible with current media browsers.

There are many interesting avenues for future work. The data model allows for content-based dimensions, but that support is missing from the prototype. We plan to add support for dynamic browsing dimensions, e.g., based on key-word search or content-based similarity, thus integrating browsing and searching into a single framework. Much of the interesting work then lies in applications of the ObjectCube model. We have already implemented a photo browsing application with a face recognition plug-in, but other applications of interest, which may require specialized plug-ins, are in music browsing and multi-dimensional file-system browsing. We believe that the ObjectCube model proposed in this paper can radically change the media browsing experience.

References

[BC09] Ilaria Bartolini and Paolo Ciaccia. Integrating semantic and visual facets for browsing digital photo collections. In *Proc. SEBD*, Camogli, Italy, 2009.

[Bed01] B. B. Bederson. PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps. In *Proc. UIST*, Orlando, FL, USA, 2001.

[CM99] Richard Connelly and Robin McNeill. *The Multidimensional Manager*. Cognos, 1999.

[2] In addition to tag-based facets, Scenique also offers facets based on content-based descriptors. The ObjectCube model does not prevent this in any way and we plan to add support for content-based browsing to the server prototype.

[DAS09] Patricio Domingues, Filipe Araujo, and Luis Silva. Evaluating the performance and intrusiveness of virtual machines for desktop grid computing. In *IPDPS*, Rome, Italy, 2009.

[DMRS10] Mamadou Diao, Sougata Mukherjea, Nitendra Rajput, and Kundan Srivastava. Faceted search and browsing of audio content on spoken web. In *Proc. CIKM*, Toronto, Canada, 2010.

[Fer07] S. Ferré. Camelis: Organizing and browsing a personal photo collection with a logical information system. In *Proc. CLA*, Montpellier, France, 2007.

[HaLZM11] Weiming Hu, Nianhua Xie andLi Li, Xianglin Zeng, and Stephen J. Maybank. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 41(6), 2011.

[HNS$^+$04] Susumu Harada, Mor Naaman, Yee Jiun Song, QianYing Wang, and Andreas Paepcke. Lost in memories: Interacting with large photo collections on PDAs. In *Proc. JCDL*, Tucson, AZ, USA, 2004.

[KPT$^+$08] Sean Kandel, Andreas Paepcke, Martin Theobald, Hector Garcia-Molina, and Eric Abelson. Photospread: A spreadsheet for managing photos. In *Proc. CHI*, Florence, Italy, 2008.

[KSPW07] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. Exploring music collections in virtual landscapes. *IEEE MultiMedia*, 14(3), 2007.

[YSLH03] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proc. CHI*, Ft. Lauderdale, FL, USA, 2003.

Technical Report RUTR-CS12002
ISSN 1670-5777

School of Computer Science
Reykjavík University
Menntavegi 1
101 Reykjavík, Iceland
Tel. +354 599 6200
Fax +354 599 6201
www.ru.is