A Semantic Extension of a Hierarchical Storage Management System for Small and Medium-sized Enterprises

Axel Schröder, Ronny Fritzsche, Sandro Schmidt, Annett Mitschick and Klaus Meißner

University of Dresden, 01187 Dresden, Germany, 2011
{axel.schroeder, Ronny.Fritzsche, Sandro.Schmidt, annett.mitschick, klaus.meissner}@tu-dresden.de,
WWW home page: http://www.mmt.inf.tu-dresden.de

Abstract. The number of company data deposited in hierarchical storage management systems heavily increases. Thus, new approaches are necessary to keep track of a data pool. This paper introduces a semantic storage extension (SSE) for existing hierarchical storage management systems that allows them to exploit semantic relations between files and use them for a more efficient and more intelligent data management. Our approach enhances traditional hierarchical storage management systems regarding migration, deletion, and retrieval operations by making use of semantic relations between files and contextual knowledge. Thereby a predictive file management is possible, which contributes to an increasing system performance and a better user experience. To this end, the SSE uses extracted features of documents to define relations between them and also offers the possibility to specify additional knowledge by a domain expert.

Keywords: semantic, hierarchical, storage, management

1 Introduction

At present, the amount of digital data stored by companies doubles year by year [15]. To save costs, companies increasingly use (hierarchical) storage management systems (SMSs). Those systems distribute data between different storage technologies and deposit information in a cost-optimized way. A SMS typically divides the storage environment into three tiers (Figure 1). The performance tier utilizes very fast and also expensive storage technologies like solid-state drives or SAS[1]/FC[2] hard disks. The capacity tier usually consists of SATA RAID systems which offer lower costs per gigabyte, but also suffer from an higher access time. The archive tier uses long time archiving technologies (WORM[3]) like optical

[1] Serial Attached SCSI
[2] Fibre Channel
[3] Write Once Read Many

jukeboxes or tape libraries. This tier offers the lowest costs per gigabyte and the highest capacity, but it also has a very high access time. So the SMS has to find a tradeoff between costs, capacity and access time. It has to distribute its data in an optimized way. We concentrate on three typical operations of a SMS: *migration*, *retrieval* and *deletion*. Migration means the movement of a file to a slower storage tier (e.g., from the performance to the capacity tier) and retrieval means the opposite, the movement to a faster storage tier. However, current SMSs treat files individually and do not recognize and use semantic relations (e.g same author, topic, accessdate, ..) between them. Furthermore, about 80 %

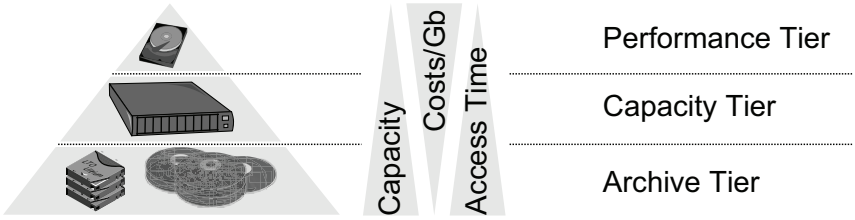

Fig. 1. A typical three tier architecture of a hierarchical storage management system.

of all data in a company is stored unstructured and contentwise unorganized [4], what soon becomes problematic for a useful cost optimization. Also only about 2 % of all stored data is used in daily business [17]. This means only these 2 % of all available data needs to be provided within the SMS's performance tier. The use of inherent and additional semantic information and semantic relations between files is one key to a more efficient and more intelligent way of storage management. There are other approaches to achieve a better performance, like monitoring user behaviour for a anticipatory data management, which are not focussed in this paper. We are concentrating on the aspect of using semantics in SMS. However, current SMS do not or only rudimentarily use those semantic relations. Treating this offers huge potential to improve management algorithms in order to achieve better performance and user experience.

This paper introduces a semantic-aware software component as extension for a SMS that allows the use of semantic information to optimize the way data is stored. We call it *Semantic Storage Extension* (SSE). The SSE can be integrated into existing SMS (see Section 3) and enables them to recognize semantic relations between documents, which may be used for distributing digital information. Initially, Section 2 will illustrate current developments and research projects in this field of study. Hence, requirements for the SSE will be derived, which are fundamental for its design, described in Section 3. This section also illustrates the functionality of the SSE and points out its interactions with other components like the SMS. Section 4 summarizes the results of this paper and illustrates both current and further work in this research area.

2 Related Work

Modern file systems like NTFS[4] already use a wealth of metadata in documents. However, these are primarily used for describing files. When trying to find similar information between different documents, the limitations of those file systems become apparent very fast. There is a wide range of papers which deal with semantics in documents (e.g. to improve file searching) [6, 8, 9, 12, 14]. It is also possible to enrich a rulebook of a document management system in order to choose appropriate file handling strategies [2]. Other approaches (e.g. [5]) try to separate the metadata from files to achieve a better system performance.

In contrast to all this work, we focus on improving the document management in a SMS. Thus, the SSE tries to help on managing files cost-orientedly according to their actuality and relevance. To achieve this goal, new semantic relations should be used. There are only few papers addressing semantic associations between documents and attempting to use those for managing files. The next section outlines some selected work of this specific research area.

2.1 Semantic Information in File Systems

To overcome the limitations of current file systems and SMSs, TagFS [1] annotates files with keywords. This is done automatically as well as user-controlled. For example, the keywords could contain different metadata, names of folders in a document path or other manually added terms. Subsequently these tags can be used to filter from a set of files. Thereby restrictions of the documents path are avoided. By using tags that reflect folder names of its original path, it is possible to filter directly for files within subfolders, without knowing their exact location (e.g., *../pictures/vacation* or *../vacation/pictures* then means the same).

Another approach for mapping semantic information in file systems was created in 2003 by introducing semantic vectors [7]. Thereby the metadata of files are converted into vectors, which subsequently span a common feature space. This leads to two results: On the one hand, duplicates easily become visible and on the other hand, strong dependencies between several documents can be found through vectors that are very close together.

There is another approach outlined in [16]. It illustrates how to capture external events in an ontology and to link them with the data that is related to this event. By doing so, the data pool is enriched with additional knowledge and files get indirectly linked with each other via events. For example, if files are created or modified during a phone call, they become related to this event inside the ontology. Thus, these documents have an implicit relation to all other files related to this event. The authors clarify that the correct linking of files with events needs a longer training process. But they also underline that this approach finally works very precisely. This idea allows a SMS to cluster data based on their real connections. Another advantage is that the users also benefit from this concept. They get the possibility to retrieve data by recalling specific events

[4] New Technology File System

(meetings, phone calls, ...). So this approach closely resembles human thinking. The navigation through a data pool is no longer based on the place (Where is information stored?), but on events (Why and Whereby?). A disadvantage is, that the relevance of files is not captured. So this approach has great potential for user-based files searched, but still needs additional improvements to support internal SMS operations.

2.2 Semantic Information in Additional Systems

There is a need for more research in the analyzed areas. The generation and processing of additional information creates a greater workload. For example, the approach of [16] introduces a mechanism to gather events. Corresponding to the size of a company, this concept can require extensive upgrades in the existing IT infrastructure. However, a long-term influence on current file systems is only possible, if such an approach is enforced as a well-established practice in a company. Also, it has to be carefully considered, if the achieved advantages justify the higher resource requirements. At this point, most publications only provide theoretical estimations or smaller field tests. In [17], studies about extensive scenarios are realized and performance and flexibility of those approaches are evaluated. They especially indicate, that modern DBMS[5] (in this case MySQL[6]) are not optimized for a very large number of metadata. Also many of the investigated concepts are only partially applicable for ubiquitous use. For example, some approaches require additional computing power to permanently extract metadata and analyze them, which then can be used to derive semantic relations. Usually it is very difficult to realize event gathering on external devices (e.g., fax) and connect them to documents, because there are no standardized interfaces to catch these events. So the integration of the illustrated approaches in current file systems and the combination of different research concepts is tricky. Thus, a solution is needed, which has enough potential compared to conventional methods and also presents an additional value abreast them.

3 The Semantic Storage Extension

This section illustrates the design for the SSE in detail. In Section 3.1, necessary requirements for the SSE are described. They lead to a software architecture outlined in Section 3.2. The following sections show the functionality of a semantic service component (Section 3.3) which is needed for feature extraction, illustrate necessary modifications inside the SMS (Section 3.4) and finally describe the structure of the SSE (Section 3.5).

[5] Database Management Systems
[6] http://www.mysql.com/

3.1 Requirements

As shown in the introduction, the data pool of SME[7] grows exponentially. Thus, it is important that a semantic extension for a storage management system (SSE) works performantly even after years. In contrast to traditional SMS which just relocate files in regarding their own context, the SSE should offer a foundation to enable the SMS to preemptively relocate related files as well. Furthermore, metadata should be managed centralized and independently from their documents. Similar to the approach of [5], who suggests separating metadata, two advantages follow. Metadata can be accessed faster and the number of read accesses on the actual storage media decreases, which means a longer lifetime [11]. In respect of the limitations of DBMS, metadata should be stored in an ontology [17]. Current DBMS often only support data mining and clustering. An ontology allows a more expressive description of semantic relations and metadata. It offers additional possibilities for reasoning mechanisms to infer semantic knowledge that is not explicitly modelled. Furthermore, the approach of gathering and processing events offers a huge potential [16]. The SSE should link documents with external knowledge to not only manage data about their structure and content, but about their origin and meaning. By doing so, the SMS would be able to not only provide relevant data for fast access, but presenting other data that is semantically close as well.

In order to function as an extension, the SSE has to work autonomously. It should enhance the underlying SMS with minimal modifications, but never be able to interfere with the SMSs basic functionality. This means the SSE should be available for the SMS through its own interface as an independent component.

Another requirement is a centralized metadata storage [5]. It can be used to look up information about the data pool in one place and use it to quickly get a link to related files. The metadata should also be managed centrally inside the SSE. Here it is especially necessary to pay attention to the consistency of information regarding the data pool in the SMS.

The heterogeneity of the data pool requires various and complex extraction algorithms. Metadata should not only be extracted from current file formats, but new file formats should be supported in the future. The complexity and extensibility of those extraction processes requires a complex treatment and was not focused yet. A semantic service component (SSC) has to provide these extraction features. The SSC has to be able to extract all necessary metadata, information from the file system, semantic information of single documents and save them into an ontology. In addition, a mutable set of policies inside the SSE is needed. Through these policies it should be possible to capture additional knowledge and connect it with the ontology. This additional knowledge is not implicitly available and can not be derived through extraction algorithms from the data pool itself. To avoid sophisticated, technical modifications as described by [16], a domain expert should become the informal interface between the company's processes and the SSE. So the SSE operates semi-automatically, whereby existing

[7] Small and Medium-sized Enterprises

information are extracted automatically and additional knowledge is generated manually. An advantage of this procedure is the possibility to consider different processes and requirements of a company.

Another requirement for the SSE is, that it just provides advisory functions for the SMS. The SMS may consult the SSE, but must never lose control over its tasks and responsibilities. The SMS consults the SSE by requesting semantically related documents to a given source document or a set of source documents. Furthermore, it should be possible to inform the SSE about which tasks should be performed with the source file within a request (e.g., deletion, migration, ...), in order to support a decision. To achieve these requirements the SMS needs the ability to query the SSE and correctly interpret its answer.

3.2 Software Architecture

The requirements in Section 3.1 lead to a software architecture that is shown in Figure 2. Our approach concentrates on hierarchical SMSs, but is also usable for otherSMS that use equal operations on files (e.g., migrating them between different storage tiers). The figure shows that the SMS communicates with the SSE over a dedicated interface. Thereby, requests for documents are sent to the SSE, which searches for existing semantic relations to other documents. This interface is also used to inform the SSE about every modification in the data pool, to ensure consistency to its ontology. Additionally, the SSE communicates with the SSC, which extracts all relevant information and manages them in an ontology. In the context of this analysis, a controlled data access to the SMS's files takes place, where the SMS stays in charge and provides access only to files, that are needed for the current update. Furthermore, the system policies for getting semantic associations are administrated in a decentralized way. The following sections describe the realization of this architecture in detail.

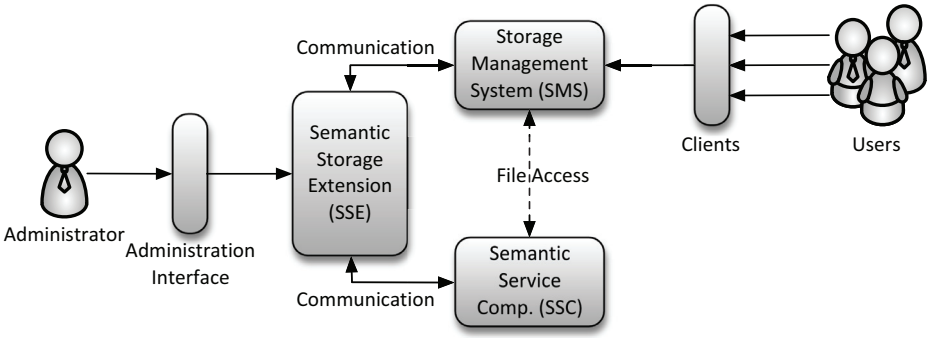

Fig. 2. The software architecture

3.3 Functionality of the Semantic Service Component

The SSC needs access to the data pool to analyze documents in the SMS. To decide which data should be analyzed and stored, the storage system has to specify these documents, whereby two procedures are possible. The first one is a complete analysis. This is normally triggered when the SSE is activated for the first time or if a full consistency check should take place. The second procedure is a partial analysis, which is called at runtime, whereby only modified, removed or new documents of the SMS are examined.
During the analysis, the SSC has to extract all available information regarding files and save them in an ontology. This information can be categorized as follows:

1. File system information (e.g., resident attributes like filename or -type)
2. Metadata (e.g., ID3[8], TEI[9], EXIF[10])
3. Implicit semantic knowledge (e.g., CBIR[11], face recognition or audio analysis)

Another important requirement is the extensibility of the SSC to integrate new extraction algorithms for future file formats or metadata standards.

The available information about documents is very heterogeneous regarding their attributes and parameters. Different identifiers sometimes got the same meaning (e.g., *author/creator*, *creation/date of creation*). This leads to a lack of interoperability. Therefore, navigation or search in the knowledge base is very expensive. To achieve a good performance with an increasing data pool, we broke down all semantic information to four fundamental dimensions: *Person*, *Place*, *Topic* and *Moment*. Figure 3 illustrates the structural layout (schema) of information for a specific document inside the ontology. All extracted information is reduced to these four semantic concepts which are instantiated at runtime and stored in the ontology. So they represent a very compact document knowledge base. The ontology schema allows to ask about the *who*, *where*, *what* and *when* in context of a document.

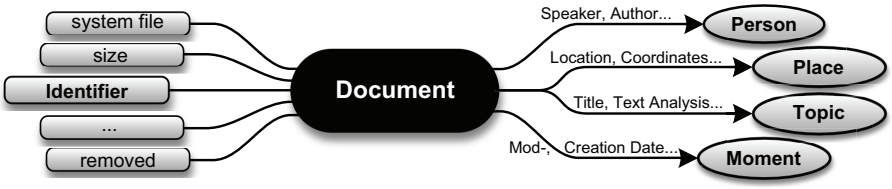

Fig. 3. Used ontology schema to describe a document's context information

[8] Identify an MP3 (metadata for audio files), http://www.id3.org
[9] Text Encoding Initiative (description of text documents), http://www.tei-c.org
[10] Exchangeable Image File Format (metadata for images), http://www.exif.org
[11] Content Based Image Retrieval

Figure 3 shows the semantic concepts and below them extracted feature (*Speaker*, *Location*, ...) which led to the specific concept. For example, an extracted *speaker* leads to a designated *person*and a document *title* leads to a specific *topic*. Furthermore, each document has basic properties which are derived from the file system (*system*, *size*, ...). To differentiate between removing and irrecoverable removing, the property *removed* marks a document as removed. So it can recovered until is was irrecoverably deleted.
We use the K-IMM[12] system [10] as a demo SSC. It especially offers most of the functionality we specified. In particular the extraction mechanism for file system information, metadata and content based information retrieval. Additionally, K-IMM has a modular structure, which easily allows the extension with new extraction mechanisms. Furthermore, K-IMM stores generated knowledge in an ontology and uses the ontology schema described in Figure 3.

3.4 Modifications in the Storage Management System

To interact with the SSE, the SMS needs some modifications. Basically it has to be able to request the SSE for semantically related documents. Also it is necessary that the SMS interprets the answers. To ensure consistency of the ontology it is important that modifications in the data pool are immediately delivered to the SSE. Last but not least, a common interface is essential to realize those tasks. The following subsections illustrate the modifications which have to be done.

Requesting the SSE According to the requirements (Section 3.1) the SMS can use two different types of requests. We call them *simple request* and *directed request*. Simple requests only pass the identifier (ID) of a file in the SMS. Each file that has stored metadata records got an ID (see Figure 3) that is used to link a file to its metadata inside the ontology. Simple requests are used to get a list of semantically related documents to a source document without any additional knowledge. The second type are directed requests which have a second parameter. This informs the SSE about the planned action for the source file (inside the SMS). Currently our concept supports the three core operations: migration, retrieval and deletion. The aim of a directed request is not only to receive a list of semantically related documents, but also to receive a recommendation for a given action.

Another important aspect are concurrency issues. To fulfill the requirements, the SMS should never wait for a response from the SSE. It has to be guaranteed that the SMS can process its tasks without depending on the SSE. The SMS just gets the advice to request recommendations from the SSE before starting a planned task and then to integrate the response to its workload to optionally re-schedule future actions.

Between all files in the storage system, there are at least weak semantic associations. For example, all files are stored in the same file system, that are

[12] Knowledge through Intelligent Media Management

managed by the same SMS. Since the SSE also considers weak semantic bindings, a response list could be very large. In the worst case, the response set contains all files of the storage environment ordered by their relevance. At this point another parameter for requests is introduced. This one is optional and is used to limit the size of the response list to a request. This threshold parameter represents the maximum disk space consumption in megabyte for related files. For example, if only 2 GB of data could be retrieved from a specific storage tier, the value of the parameter has to be 2048. So the response list only contains as many related documents as the SMS can use.

Response Interpretation Responses of the SSE basically contain a sorted list of file identifiers (FIDs). The order and their interpretation is influenced by the type of request. Usually, first listed FIDs have a closer semantic relation to a source document than FIDs with a lower rank.

In case of a directed request the SMS initially examines the recommendation of the SSE. This is constructed as a boolean value. *True* stands for an approval and *false* for a rejection. The detailed interpretation of the response can be categorized as follows:

Directed request for a migration: As long as the SSE approves the planned migration (true), it can take place. In this case no sign has been found that the source file belongs to the active data pool. So it would be recommendable for the SMS to migrate other documents from the response list, which are also most likely redundant. This helps to predictively move unused data into a slower storage tier to save costs. If the SSE responds with a rejection (false), the response list has to be interpreted in the opposite way. The source document (and its semantically related documents) seems to be relevant and active in the company. Thus, a migration to a slower storage tier is not advisable.

Directed request for a retrieval: A file retrieval works in an opposite way. For example, a rejection (false) of a specific document means that this one does not have (many) active or relevant relations to other files. Therefore it is unnecessary to move this file to a faster storage tier. Also semantically relevant documents most likely do not need to be retrieved.

Directed request for a deletion: This case requires an additional treatment by the SMS. Also the significance of the response list has to be handled with care. If the SSE calculates a high relevance or actuality for the requested file, its response contains a rejection (false). Like the other requests, this recommendation counts for all files in its response list as well.

Modifications in the Data Pool As described in Section 3.3, we differentiate a partial and a complete analysis. At the first activation of the SSE and also at the regular consistency check of the ontology, the SMS initiates a complete

analysis, whereas the modification of single informations just invokes a partial analysis.

For a well-structured delivery of necessary data to the SSE, the SMS needs an additional module. This module provides all affected documents as separate data streams. Because of the complexity of the data pool, all data streams should be processed iteratively during the analysis. Every data stream contains a reference to following data stream (chained list). The SSE can process them step-by-step until all modified documents, which are affected by this update, are processed. To clearly identify a data object, an FID has to be embedded into the data stream.

3.5 Design of the Semantic Storage Extension

In this section, we introduce the structure of the underlying policies and show how to formalize explicit semantic knowledge. Next, the process of reasoning will be explained, particularly, how the SSE determines related files by semantic bindings. Concluding, this subsection will show how response lists are generated and structured.

Policies With policies, the domain expert should be able to specify any knowledge about the processes and the structure of a company. A DBMS is used to store this knowledge. Below, the structure of the database tables is illustrated by the following example: "If documents were modified at Computer7 by Mrs. Schulz or Mr. Meyer, they belong to the accounting." To express this knowledge, the domain expert can use two types of policies: *basic policies* and *relational policies* (Figure 4).

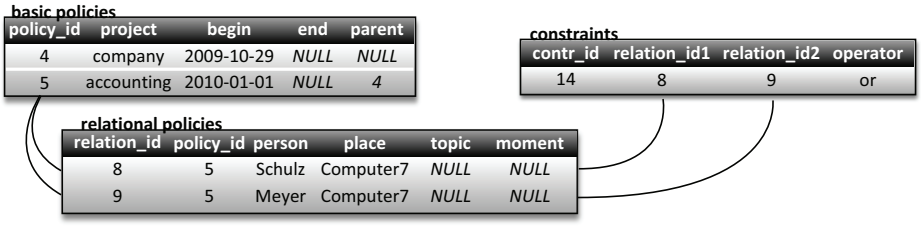

Fig. 4. A simple policy example

Basic policies gather all processes which have temporal boundaries. These processes are called *projects*. *Policy 4* implies that the project *company* started on 2009-10-29. Furthermore, *Policy 5* defines the project *accounting* which started on 2010-01-01. Both projects have no defined ending. Also the *accounting* is part of *company*.

The second type of policies are relational policies which are stored in a separate table (Figure 4). They associate existing projects with persons, places,

moments and topics. *Policy 8* declares that all documents which contain the person *Schulz* and were created or modified on *Computer7* belong to accounting (*policy id 5*). *Policy 9* describes the same for *Meyer* and *Computer7*.

To minimize the amount of policies, a third table (*constraints*) is introduced (Figure 4). In this table, dependencies between relational policies can be expressed by logical operators. In our example, constraint 14 describes a relation between policy 8 and 9 and combines them by an *OR*-operator. This means that only one of those policies needs to be fulfilled for a document to be matched to the accounting.

Procedure for Getting new Semantic Information Figure 5 shows how the SSE tries to find semantically related documents and which communication is necessary between the involved components. The SMS requests the SSE and inform it about the ID, and optional about the planned operation, of a source file. Initially this ID is used to query the SSC for all information on this file. The request itself is created by the SSE and formulated in SPARQL [13]. How the returned knowledge is structure, is described in Figure 3. Next, the SSE checks if existing relational policies match and if projects may be associated. For example, if a source document is related to a place *Computer7* and a person named *Meyer*, then already one of the two policies matches. As both policies are *OR*-linked, the file would be assigned to the project *accounting*.

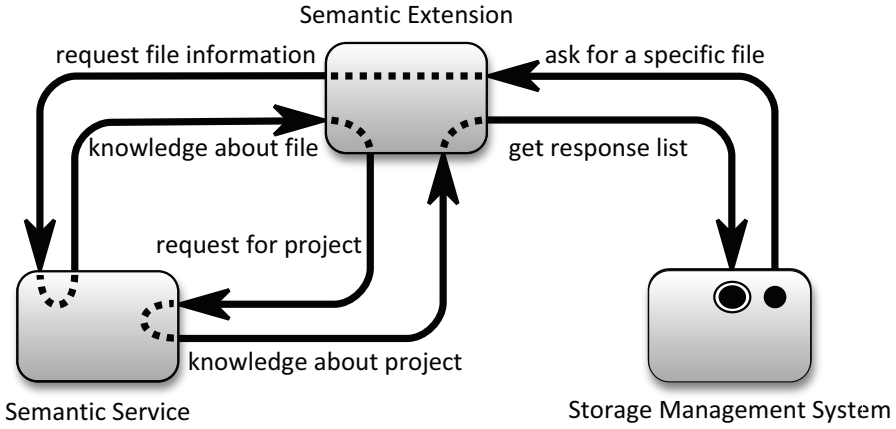

Fig. 5. Workflow for getting semantically related documents

If all projects which can be assigned to a given document were found, the SSE executes another SPARQL query against the SSC. This query determines the IDs of all files which can be associated with the found projects. For the project *accounting* this means that all IDs of files are requested which are related to at least *Meyer* and *Computer7* or *Schulz* and *Computer7*.

Thereafter the results are converted into data objects which are separated into different sets. A data object not only contains the ID of an affected file, it also contains different properties (file size, ...), attributes (system, ...) and the last access time. Each set represents documents with a simple semantic relation to the source file.

The next step is a calculation of the degree of relationship of the determined files. All data objects of the different sets are merged into one response set. Data objects that contain the property "removed" (Figure 3) are skipped. If a data object is not already available in the response set, it is added with a counter initialized with 1. This counter represents the number of determined semantic relations (degree of relationship). If a data object is already available, it is not repeatedly added. Instead the counter is incremented by 1. The result of this calculation is a set of data objects that contains different degrees of relationships for documents regarding their binding to a source file.

Before the response list can be generated, a recommendation has to be prepared, if requested. If the request of the SMS contains a deletion as planned action, the attributes of the source document are checked again. If there is a "system" attribute, the recommendation will be *false*. Otherwise, all associated projects are considered. Those projects have a specified period of time. If one of them is active (the end point is in future), the source document is also classified as active. In this case the recommendation for deletion or migration is set to *false*. Otherwise, if all related projects are inactive (the end point is in the past) the recommendation is set to *true*, which means a migration or deletion of related files is possible. If the planned action is a retrieval, the recommendation is inverted (Section 3.4).

Generating Response Lists If the request of the SMS contains a threshold parameter (Section 3.4), the SSE makes sure that the sum of all file sizes in the response list does not exceed this value. A response list is constructed as a dual sorted list of data objects with an optional recommendation flag (Figure 6). Each data object represents a semantically related document to a source file. The dual sorting offers the SMS an additional benefit on its processing. Furthermore, the order depends on the planned action. Figure 6 illustrates an example for a directed response list in case of a planned migration. The data objects themselves are labeled from *FID1* to *FID9*. The primary order is accomplished in accordance with the number of semantic bindings for each data object (curved brackets) and in the case of a migration the secondary order is accomplished with the file size, starting with the biggest file. So in case of a migration, if documents own the same semantic binding, the bigger ones are migrated first. The advantage is that just a few operations are necessary to create enough space and that a lot of small files can remain on the fast storage tier. This is particularly useful if no threshold value was submitted.

Response lists for simple requests are sorted the same way but do not contain a recommendation bit. In case of a retrieval, the secondary order is based on the file size, starting with the smallest one. This is done to retrieve as many actual

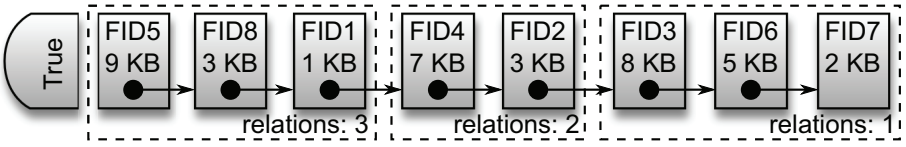

Fig. 6. A sample response list for a planned migration

documents as possible. However, on a deletion, the SSE does not consider file size. Here, the secondary order is based on the last-access-time-property of data objects. Thus, older files are listed first.

4 Conclusion and Further Work

A great amount of data will force software designers to implement more efficient and scalable algorithms to optimize storage solutions. One way to reach this goal is given by semantic web technologies.

As we have showed in Section 2, there is a great lack of publications about using semantic technologies in SMSs. Current approaches do not regard relations between documents. Only simple hierarchies are used for classifying files and folders.

In this paper, we introduced the so-called Semantic Storage Extension (SSE) (Section 3). The SSE is a software component which can be integrated in an existing SMS with just minimal effort. To enable a semantic information extraction, this architecture is completed by a SSC which handles extracted information by using a specialized ontology schema to describe documents in a semantic way. As K-IMM is used as a SSC, it is easy to add new extraction plug-ins to analyze future document formats and enable information retrieval in the way it is necessary for the SSE. Through inference algorithms that are provided by the SSC new relations between indexed documents can be found which cannot be derived by methods like data mining and clustering. Using this knowledge, the SSE can advise the SMS on planned actions for a collection of files. To fulfill the requirements of different domains, we developed an additional policy-based approach to enable a domain expert to describe the application domain in detail. With this architecture a SMS can decide on actions like migration, retrieval and deletion by using semantic knowledge.

For future work we plan to improve the implementation of this architecture in the project HSM [3]. Thereby, we want to perform tests to proof our theoretical evaluation and to show that the benefit and the performance for a SMS increases. Also, a detailed analysis need to be made on security issues (like authentication and authorization), sorting of given answers and the way files are weighted in response lists. At least the handling of concurrence issues between the SMS operations and operations of the SSC and SSE needs to be improved.

References

1. Bloehdorn, S., Görlitz, O., Schenk, S.: TagFS - Tag Semantics for Hierarchical File Systems. Proceedings of the 6th International Conference on Knowledge Management I-KNOW'06 (2006)
2. Chang, W., Masinter, L.: System and method of determining and recommending a document control policy for a document (2008), http://www.freepatentsonline.com/y2008/0059448.html
3. Fritzsche, R.: HSM-Projekt (2011), http://www.mmt.inf.tu-dresden.de/Forschung/Projekte/HSM/
4. Gnasa, M.: Fraunhofer IAIS: Text Mining und Information Retrieval (2009), http://www.iais.fraunhofer.de/4862.html
5. Hackl, G., Pausch, W., Schönherr, S., Specht, G., Thiel, G.: Synchronous Metadata Management of Large Storage Systems. Proceedings of the Fourteenth International Database Engineering & Applications Symposium pp. 2–7 (2010)
6. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Prentice Hall (2008)
7. Mahalingam, M., Tang, C., Xu, Z.: Towards a semantic, deep archival file system. The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems pp. 115–121 (2003)
8. Mierswa, I.: Beatles vs . Bach : Merkmalsextraktion im Phasenraum von Audiodaten. Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivitat (2003)
9. Mitschick, A.: Ontologiebasierte Indexierung und Kontextualisierung multimedialer Dokumente für das persönliche Wissensmanagement. Dissertation, University of Dresden (2010)
10. Mitschick, A.: Ontology-based Indexing and Contextualization of Multimedia Documents for Personal Information Management Applications. International Journal on Advances in Software 3(1), 31–40 (2010)
11. Neuroth, H., Oß wald, A., Scheffel, R., Strathmann, S., Huth, K.: nestor Handbuch - eine kleine Enzyklopädie der digitalen Langzeitarchivierung Version 2.3 (2010), http://nestor.sub.uni-goettingen.de/handbuch/nestor-handbuch_23.pdf
12. Orio, N.: Music Retrieval: A Tutorial and Review. Foundations and Trends® in Information Retrieval 1(1), 1–96 (2006)
13. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2008), http://www.w3.org/TR/rdf-sparql-query/
14. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
15. Troopens, U., Erkens, R., Müller-Friedt, W., Wolafka, R., Haustein, N.: Storage Networks Explained. John Wiley & Sons Ltd., Mainz, 2 edn. (2009)
16. Weippl, E.R., Klemen, M., Linnert, M., Fenz, S., Goluch, G., Tjoa, A.M.: Semantic Storage : A Report on Performance and Flexibility. Lecture Notes in Computer Science 3588, 586–595 (2005)
17. Xiong, M., Jin, H., Wu, S.: FDSSS: An Efficient Metadata Management Scheme in Large Scale Data Environment. Fifth International Conference on Grid and Cooperative Computing Workshops pp. 71–77 (Oct 2006), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031532