Semantic Storage: A Report on Performance and Flexibility

Edgar Weippl, Markus Klemen, Manfred Linnert, Stefan Fenz, Gernot Goluch, and A Min Tjoa[1]

Vienna University of Technology, A-1040 Vienna, Austria,
weippl@ifs.tuwien.ac.at, klemen@ifs.tuwien.ac.at
WWW home page: http://www.ifs.tuwien.ac.at

Abstract. Desktop search tools are becoming more popular. They have to deal with increasing amounts of locally stored data. Another approach is to analyze the semantic relationship between collected data in order to preprocess the data semantically. The goal is to allow searches based on relationships between various objects instead of focusing on the name of objects. We introduce a database architecture based on an existing software prototype, which is capable of meeting the various demands for a semantic information manager. We describe the use of an association table which stores the relationships between events. It enables adding or removing data items easily without the need for schema modifications. Existing optimization techniques of RDBMS can still be used.

1 Introduction

It is commonly known that the amount of digital information increases as people store more information on their Personal Computers. The variety of formats used increases because people store their digital images, emails and faxes they sent and received, and documents they created as well as many files from the Internet. There are several approaches to storing *lifetime's worth of articles, books, ...*[1] which we briefly discuss in Section 4.

The basic idea of our system has the same goal, allowing easy access to all data and information, but differs fundamentally in the approach. We focus on the relationship of various data-objects (such as photos, e-mails, graphics or text files) and events (opening a text file, receiving a phone call, sending an e-mail) rather than relying on the names of these objects for retrieval. Our intent is to allow for more human-like retrieval processes by adding semantic metadata to the data collections. For example, instead of finding a text file based on its name, a semantic search would allow a context-aware query, for instance *I don't know the filename but I know I created it when I was talking to Jim on the phone about a week ago.*

Our prototype (Figure 1) collects raw data from multiple sources such as file events of the operating system or user events from Microsoft Outlook via agents.

[1] http://research.microsoft.com/barc/mediapresence/MyLifeBits.aspx

On the level of the file system we receive data on which application has accessed (read or written) which file. The user can select which applications or directories to monitor, typically local office applications and user document folders. For instance, we can store that Word.exe has written a specific file document.doc to the disk at a certain time and date. From Microsoft Outlook we can gather information on emails, contacts and calendar items. We also store which computer has been used (to differentiate between laptops and workstations) and the user ID. Additional data collectors are planned that can integrate incoming and outgoing telephone calls (via CTI or serial printer ports) as well as facsimiles, GPS data and EXIF[2] metadata from digital camera images.

Based on the vast amounts of data accumulated, a semantic enrichment engine (SEE) is implemented which uses the data and derives information from it to build semantic databases for human users. Clearly, the usefulness of the whole system depends on the quality, speed and versatility of the semantic database and on the capability of the semantic enrichment engine. In this paper, we will focus on the underlying database schema and propose a database architecture which is the foundation for the semantic analysis. There are certain requirements for such a database:

- *Flexibility:* A database for semantic storage must be highly flexible. It must be able to store heterogenous data from various sources, including e-mail systems, file systems, date books, telephones or GPS modules. Defining new relationships between existing entities will be a common task.
- *Backwards compatibility:* All enhancements to the database must be backwards compatible. Modification of the database schema should occur only rarely.
- *Speed:* The database must perform well due to the high volume of processed data
- *Scalability:* The database design should allow to scale the database up without significant performance loss.

More specifically, our contribution is to:

- Propose a suitable and flexible schema of storing semantic information on relational database engines (Section 3).
- Compare our database schema to other viable approaches (Section 3.1).
- Provide an analysis of performance in comparison to other approaches (Section 3.2).

2 The Problem

Evermore data storage enables people to save virtually their whole life digitally in various file formats or databases — photos, videos, e-mail, address databases etc. Available personal programs to store and manage these files usually offer

[2] http://www.exif.org/

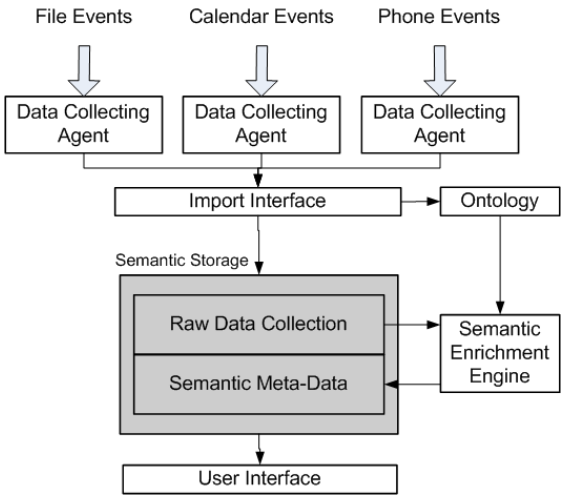

Fig. 1. System architecture: data is collected from different sources and stored in the raw data collection. Subsequently, the semantic enrichment engine (SEE) analyzes the data, adds links between recorded data items and may add additional tables or data items based on the analysis.

searches either via file system hierarchies or via keywords or fulltext search (in case the file contains text data). *Filesystem hierarchies* are not well suited since often precise attributions to a single specific folder are not possible [DEL+00]. *Keywords* are usually either based on the filenames or they need to be typed in manually. Manual keyword input is cumbersome, time consuming and subject to the *Production Paradox* [CR87] — people will simply not do it since they see no immediate advantage. *Fulltext-engines*, on the other hand, are only useful for text-based documents. Integrating photos and music into fulltext-based system is difficult and an area of ongoing research.

Apart from that, people tend to forget names of specific objects. It is often easier to remember the *context* of a situation in which a specific file was created, modified or viewed, especially with regard to a *timeline* ("I remember I just got an e-mail from Mike when I was working on that document"). Semantic enrichment of automated data-gathering processes is a useful tool to complement this human way of thinking in relations rather than thinking in keywords or tags.

Our database schema allows two links to describe the relationship between any two entities — one in each direction. The architecture must also allow to modify the semantic relationships easily and without database schema modifications or reprogramming.

The traditional relational model is considered a basic modeling technique that all computer science students have to know. The model, however, has one important drawback when relationships between already stored data objects need to be modified. Typically, in a semantic information manager we would

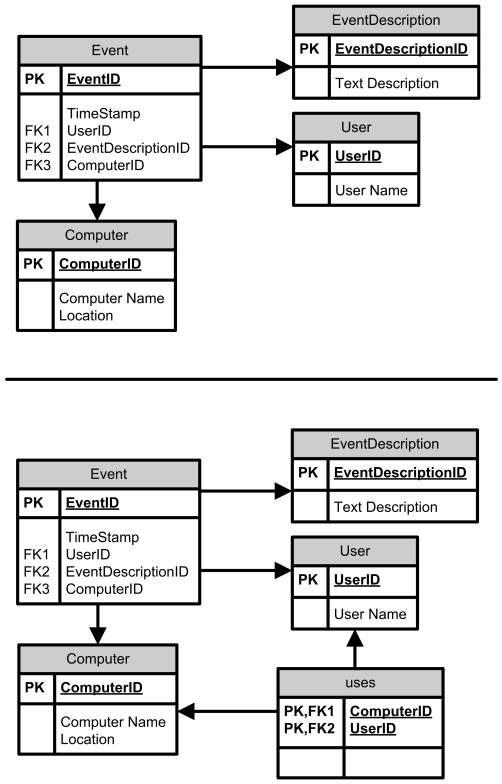

Fig. 2. The upper image shows that for each `event` an `event description`, the `user` who triggered it and the `computer` can be stored. When a new relationship is introduced to record who `uses` a computer the database schema needs to be modified.

have to link many different objects, such as e-mails with people, phone calls with numbers etc. This requires many $n : m$ relationships, which are usually realized in modern database systems by introducing a third table which incorporates a $1 : n$ and $1 : m$ relationship. Links between tables are usually realized using foreign keys.

Figure 2 (upper image) shows a simple example using foreign keys. In order to record which user triggered which event on which computer we need the tables `event`, `user` and `computer`. To add a new relationship, e.g. indicating which user is the owner of which computer, we need a new $n : m$ relationship, which would require adding a new table to handle the $1 : n$ and $1 : m$ relation.

Such a database structure would allow to relate any number of objects (documents, e-mails, pictures, date book entries) with events (e.g. creation of a file, responding to email, viewing a picture, changing a date book entry) if all tables and all relationships are known a priori. However, a database containing semantic information is highly volatile with regard to the relationships, which can vary often and extensively. Additionally, since dominant relationships are

usually not known a priori, optimizations using cluster functions of the DBMS are extremely difficult to achieve.

Let us assume that based on all the information stored in the system an improved mechanism of semantic analysis can now automatically add descriptions (qualifiers) for events and add relationships automatically.

For instance, if we realize later that it is important for Powerpoint presentations to be associated with people (`users`) attending an Outlook `meeting` we would need to add a new table `watched` to resolve the new $n : m$ relationship. The obvious drawback is that we have to modify the database schema and introduce new indices.

What we were looking for was a way to circumvent these problems using relational databases without being forced to use vendor-specific features not generally supported by SQL standards. We made our system more or less independent from any specific database vendor, so that users can chose to use MySQL, Oracle or MS-SQL[3].

3 The Semantic Database Schema

Our idea, which is based on an architecture of a workflow management software project is to use a relational database, but not to link tables to others *directly* with foreign keys or by using $n : m$ intermediary tables but via a single, generic association or *link table*.

Figure 3 (top image) shows two tables. The table `event` is used to store all events. The table `link` is used to create links/association of tupels of different tables to each other.

If the service that records the event is now improved so that it now records also the applications used, a new table is added (Figure 4 (bottom image)). The table *application* stores applications that exist such as Word or Excel. When an event is related to an applications — such as the event `save` in Word — an entry in the `link` table is created.

Step	SQL Command
1	`INSERT INTO event VALUES (1, currenttime)`
2	`INSERT INTO event_description VALUES (1, 'save')`
3	`INSERT INTO link VALUES (1, 'event', 1, 'event_description')`
4	`INSERT INTO link VALUES (1, 'event', 42, 'application')`
5	`INSERT INTO link VALUES (1, 'event', 87, 'computer')`

Table 1. Inserting events and links to applications. We assume that (1) application 42 is Word and (2) that the user currently uses computer with the `ComputerID` 87.

[3] We are obviously aware of the major differences between these RDBMs but we wanted to develop a system which would allow the user to choose between various databases, depending on the number of users and volume of stored information.

In the classical schema, adding a new table with an $n : m$ relationship to another table (e.g. `event`) would require adding the table itself an the intermediate table resolving the $n : m$ relationship. Using our approach, only one new table has to be created. Our link table is basically an intermediate table resolving all $n : m$ and $1 : n$ relationships.

Using our approach it is easier to find other tables that a certain table has relationships with. In the classical schema this information is clearly stored in the data dictionary from which all meta information on tables such as foreign key constraints can be retrieved. Data dictionaries, however, are not standardized for all RDBMSs. Therefore, applications need to be modified when using different RDBMSs.

Moreover, using our system it is easily possible to synchronize data from different databases by merely combining all entries from the link table[4].

3.1 Advantages to other approaches

We built the prototype of the data collecting agents to work with two different database schemas, one based on $n : m$ intermediary tables and one based on link tables.

Figure 3 shows the main tables of a prototype we built to record events triggered by the file system. For each event the system may record the `computer`, the `location`, the `application`, the involved `files` and an `event description`. In addition, parts of the tables required to include Outlook calendar entries (`meeting`) and contacts (`persons`) are shown. We omitted many tables as not to make the diagram larger than required to understand the concepts. As previously described, the drawback of this classical approach is that semantic enrichment engine (or other data collecting agents) cannot easily add new tables and relationships.

To avoid this drawback we also used a schema that uses the `link` table (Figure 4). In this case new relationships can easily be added. Changing an existing $1 : n$: to $n : m$ relationship needs no changes at all, the same applies to new relationships between two objects which had no prior link. Obviously our approach lacks any constraints on referential integrity or cardinality constraints. While such constraints are certainly important for many applications, it is not essential in our semantic setting. If required we can still implement most constraints by adding customized (functional) constraints on the `link` table. In an environment which basically allows relating any two objects, referential integrity is not really necessary. In addition it is possible to merge existing databases into one, for instance when merging the events recorded on an employee's desktop and her laptop computer.

In a semantic model, it is difficult to predict the quantity of relationships between the main tables and subsequently the number of intermediary tables. Given a schema with n tables, the number of intermediary tables has an upper

[4] The IDs that are the primary keys in each table have to be globally unique; this can be achieved by using GUIDs.

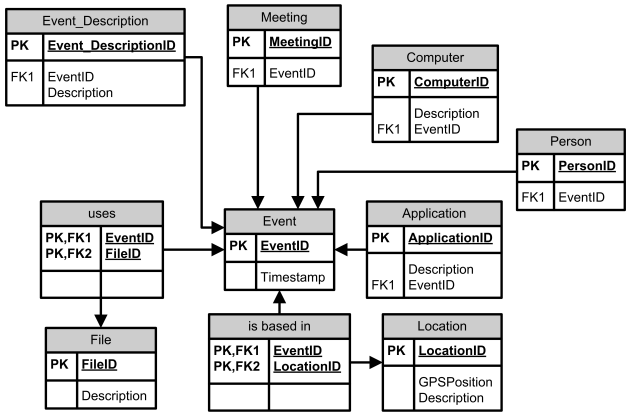

Fig. 3. To store events recorded by the `file data collecting agent` we would need this classical schema; we simplified it by selecting a subset of entities to provide a clear print.

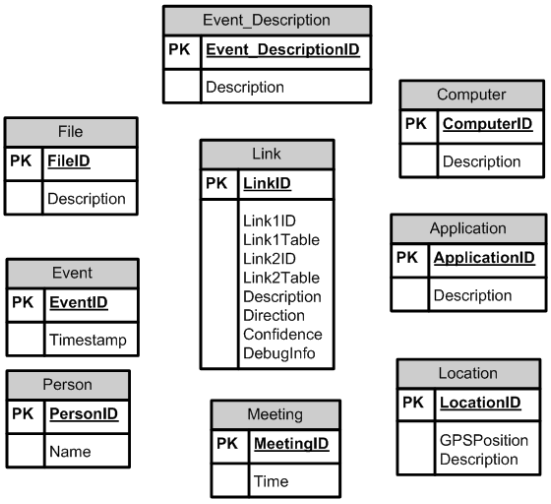

Fig. 4. Instead of using the classical schema (Figure 3) we propose to use this (simplified) schema.

bound of $\frac{n\times(n-1)}{2}$, while in our link-based database schema remains constant with one table. This is an advantage with regard to the complexity and performance of `select` statements and to the overall software complexity.

Since in a semantic setting the number of types of relationships between tables may vary widely, it is difficult to make use of cluster functionality. In our model, the link table is always the most important table, therefore, `clustering` joins with this table will have a significant positive impact on performance which

is virtually impossible with traditional approaches without manually tuning the database.

With the link-based approach adding new relationships can be performed within *transactions* whereas schema modifications such as adding new $n : m$ tables are usually performed outside a transaction. Transaction-safe operations are essential because we anticipate many different semantic enrichment engines to work in parallel.

Creating new relationships and deleting existing relationships, both in two directions, should be embedded in a proper transaction procedure. Since in normal operation mode no new tables are added or removed, this is easy to implement. This will ensure that two objects are always linked bi-directionally and removed in pairs.

3.2 Performance Comparison

To test the performance of out database schema on modern databases, we prepared two prototypes, one incorporating a traditional approach and on based on our new architecture on an Oracle 9 database system. Apart from using indeces we did not optimize the database any further - we will do this in our ongoing work on various databases such as SQL Server and MySQL to compare performance and optimization features such as cluster functions. The database was filled with random data. In the traditional design, we created six N:M relationships, each containing 175.000 entries, adding up to 1.05000 entries. In the new design, we created 1.926.064 entries (our architecture uses two entries to describe a bi-directional relationship) in the link table.

We compared three queries on both systems. Due to the difference of the architecture, the queries would vary between the traditional and the new design.

A typical query – and one we used for comparism (queries 1 and 2) – in the traditional form would look like this:

```
SELECT DISTINCT cl_event_description.description
FROM cl_event_description
INNER JOIN cl_event ON cl_event_description.event_descriptionid
= cl_event.event_descriptionid)
INNER JOIN cl_fileitem ON cl_event.activityid=cl_fileitem.activityid
WHERE description = 'qjdym.pdf'
```

A query with the same effect in the new schema would look like this:

```
SELECT DISTINCT event_description.description FROM event_description
FROM(((link l1 INNER JOIN fileitem ON l1.link1id=fileitem.fileid)
INNER JOIN event ON l1.link2id=event.eventid)
INNER JOIN link l2 ON event.eventid=l2.link1id)
INNER JOIN event_description
ON l2.link2id=event_description.event_descriptionid
WHERE fileitem.description = 'qjdym.pdf'
AND l1.link1table='event_file'
```

```
AND l1.link2table='event'
AND l2.link1table='event'
AND l2.link2table='event_description'
```

Query	Classic Design	New Design
1	1.300ms	2.600ms
2	300ms-600ms	15ms-30ms
3	800-900ms	900ms-1.400ms

Table 2. Results of our preliminary performance tests.

Query No. 1 was the initial query. Query number two had the same search arguments and was executed immediately after the first, showing a significant better performance due to the database' internal caching strategies. The third query had a different search argument. Our next steps are to perform more elaborate tests; these will also take into account database-vendor specific behavior and optimization potential .

4 Related Work

Vanevar Bush's vision of the Memex [Bus45] has recently become a paper that almost everyone cites when writing about semantically enriched information storage; projects such as Microsoft's *MyLifeBits* [GBL+02] or the *SemanticLIFE* project [AHK+04] build on this vision.

They strive to build a personal digital storage that records all documents, emails, photos, videos, etc. a person works with. MyLifeBits focuses on storing the content in a database; unlike SemanticLIFE it does not strive to semantically enrich the stored data. It seems that MyLifeBits builds on improving search engines and desktop search solutions. The focus of SemanticLIFE seems to be building ontologies and finding new relationships between existing documents.

Haystack [AKS99] — an Open Source system — is a platform to visualize and maintain ontologies. The system is designed to flexibly define interactions and relationships between objects. The focus lies on the quality of the retrieval process.

The main difference between MyLifeBits, SemanticLIFE and our approach is that all the aforementioned systems are built on the concepts of *documents* or *files*. The approach we described relies primarily on events. These events are then related to existing resources and they allow to build more elaborated links between data items. A technical but yet important difference is that we store all data in relational databases that are known to work stably even with huge amounts of data.

5 Conclusion

We proposed an improved way of organizing a database schema so that relationships between tables can easily be added without schema modification. The advantages of our approach are:

1. Relationships can be added without schema modifications. This allows to easily perform operations within transactions.
2. Tables and indices can be clustered to improve the speed of joins with the central `link` table. In the classical model many $n : m$ relationships exist and it is a priory not clear how to cluster.
3. Our approach allows retrieving relationships from the link table without accessing the data dictionary. Since the data dictionary is vendor specific, the classical approach requires modifying the application for each database system.
4. If n entities exist and $n : m$ relationships are to be established between all entities the number of additional tables is $O(n^2)$, whereas our approach is $O(1)$ Of course this applies only to new relationships, not new tables.

Our prototype is – in the first stept – designed as a single user system where users can choose which content is kept and which is not. An enhanced prototype will also support multi-user environments, which pose additional challenges with regard to security, privacy and collaboration features. We will discuss these aspects in a later paper in detail.

References

[AHK+04] Mansoor Ahmed, Hoang H Hanh, Shuaib Karim, Shah Khusro, Monika Lanzenberger, Khalid Latif, Michlmayr Elka, Khabib Mustofa, Nguyen H Tinh, Andreas Rauber, Alexander Schatten, Nguyen M Tho, and A Min Tjoa. Semanticlife — a framework for managing information of a human lifetime. In *Proceedings of the 6th International Conference on Information Integration and Web-based Applications and Services (IIWAS)*, September 2004.

[AKS99] Eytan Adar, David Karger, and Lynn Andrea Stein. Haystack: Per-user information environments. In *Proceedings of the Conference on Information and Knowledge Management.*, 1999.

[Bus45] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(7):101–108, 1945.

[CR87] John M. Carroll and Mary Beth Rosson. *Paradox of the active user*, chapter 5, pages 80–111. Bradford Books/MIT Press, 1987.

[DEL+00] Paul Dourish, W. Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Michael Salisbury, Douglas B. Terry, and James Thornton. Extending document management systems with user-specific active properties. *ACM Trans. Inf. Syst.*, 18(2):140–170, 2000.

[GBL+02] Jim Gemmel, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. Mylifebits: Fulfilling the memex vision. In *ACM Multimedia '02*, pages 235–238, December 2002.