Extending database technology: a new document data type

Stefania Leone

Departement of Informatics, University of Zurich
Binzmuehlestr. 14, 8050 Zurich, Switzerland
leone@ifi.unizh.ch

Abstract. Our research explores a new database extension. Our aim is to see how far it is possible to support collaborative editing and document provenance with database technology, and extend the technology to fully support the data type document. We present an overview of techniques relevant in this area, some of the research challenges, and present a research plan which focuses on the issues of storage, query and manipulation of documents within a database. The current state of experimental work is summarised. Then a plan for further work is presented in detail.

1 Introduction

Large numbers of text documents are produced in companies, universities and government offices every day and contain a significant part of our knowledge. These documents are most commonly stored in hierarchical folder structures on file servers and are difficult to find and manage. Most commonly, documents are produced, edited and read by more then one person. Tools for collaborative document editing, however, are very rarely deployed.
In contrast to such weakly structured data, structured data, e.g. account data or customer data are most commonly stored in a database. A DBMS provides uniform data access, security, consistency and integrity, concurrency control and recovery by default. Data storage, retrieval and manipulation is totally managed by the DBMS.
Documents are in our opinion as important as structured data and contain a lot of valuable business knowledge, but they are not taken care of appropriately. We therefore propose to bring document data to the same level of importance as structured data, i.e. into the database (giving it full database support).
A recent community report [1] voices the requirement that text should be integrated into the database as a 'first-class' data type. This means that text documents should be stored in the database in an appropriate way using an appropriate data structure accessible via a text interface supporting storage, retrieval and manipulation of document data.
We strive for a robust implementation of a document data type and its integration into SQL, so that documents are stored in the database and can be retrieved and manipulated by extended SQL statements. The goal of this thesis

is the specification, design and implementation of a robust document data type and its integration into SQL.
The paper is structured as follows. First, we present preliminary information and related work. Then, we outline the problem areas and present the approach that will be taken.

2 Preliminaries

We define a document as a set of objects (characters, tables, figures), their structure and associated meta data. Structure includes layout information, structural information in terms of chapters, sections and paragraphs, business process information, security information, semantic information and notes. Meta data consists of creator, author(s), creation date, size etc. We assume that meta data are gathered automatically during the document creation process.
The document creation process (text editing) is by its nature a collaborative process where a number of editors work, often simultaneously, on the same document. Text editing involves the following operations: writing and deleting characters, copying and pasting, adding layout, inserting images and tables etc., as regularly carried out by a word-processing user.
Compared to an ordinary data type, like a character string, a document is much more complex. Whereas a string can be said to store information in one dimension (content), a document, which in our definition consists of content, structure and meta data, is not only stored in one, but in multiple dimensions.
As mentioned before it is desirable that documents be fully supported by the database [1]. This includes the storage of content, structure and meta data and operations for accessing, retrieving and manipulating the document in the database, preferably all within a collaborative environment. This means that SQL will be extended in order to support data type specific operations such as creating and deleting documents, inserting and deleting parts of the document (e.g. characters, sentences, paragraphs) and adding layout and structure.
To summarize the proposed new features, a document as a 'first-class citizen' of a database should have the following properties:

- It should be stored entirely and comprehensively within the database including content, structure and meta data.
- It should be accessed and manipulated by the use of extended SQL statements.
- It should be accessed and manipulated simultaneously by more than one user, i.e. the new data type should be collaborative.

3 Related Work

To enable text handling in database systems, the SQL99 Standard [12] introduced, among others, Large Objects, data types for storing objects such as images and text (BLOBs and CLOBs). The SQL/MM Standard [16] includes several new data types, such as 'Full-Text' which can be used for full-text search.

The SQL/XML Standard [11] introduces the new data type XML. Several commercial database products natively implement this XML data type and offer XML data access by both SQL and XQuery [21][15][19].
Transact-SQL [10] implemented by Sybase and Microsoft is an extension to SQL that offers two text data types, TEXT and NTEXT, for storing large text and data type specific statements. These statements enable text editing functionality, i.e. insertion and removal of characters. Document structure, however, is not stored.
DB2 Text Extender [9] and Oracle Text [20] are database extensions for storing documents within a database. DB2 Text Extender enables search and linguistic operations on documents. Plain text documents are stored either inside the database in BLOBs or outside, and indexes (precise, ngram, linguistic) can be created on TEXT-enabled columns. Text is not by itself a data type and standard SQL functions can not be applied to TEXT columns.
New techniques have also been developed, based on textual query languages, including queries on content and structure and text retrieval in databases [3]. In [18] and [17] Navarro and Baeza-Yates present a model for querying documents by content and structure. Structure is represented in a hierarchical way as known from mark-up languages like SGML and XML. Text search is enriched by conditions relating to the structure. Content and structure are kept entirely separate. The language provides only query facilities, and data manipulation operations are not supported. SuperSQL [23] supports publishing and presentation of textual data stored in the database in different formats, depending on the query. Document structure is not persistently stored in the database but dynamically created when executing the query.
Collaborative editing is one of the core requirements of document processing. Several research projects have been conducted in this area. The REDUCE project[1] developed complex collaboration algorithms [22]. Also, a large number of collaborative editing products are on the market (e.g. SubEthaEdit[2], Writely[3]). In all cases, collaboration algorithms are always based on file-based documents.
Gathering meta data automatically during document processing has been the subject of research in the field of text retrieval and personal information management. Dumais and colleagues present a system [5] that indexes all data accessed by a user (documents, web pages, emails etc.) to facilitate data re-access. Google Desktop Search[4] and MSN Desktop Search[5] index all data on one's computer and allow queries on content and file meta data. Meta data about the data creation process is found not to be sufficient to support rich query functionality.
Data provenance [4] is related to meta data. Systems have been developed that store not only data but also linage and accuracy [24] of that data. To our knowledge data lineage for document data has not been the subject of research so far.

[1] http://www.cit.gu.edu.au/∼scz/projects/reduce
[2] http://www.codingmonkeys.de/subethaedit/index.html
[3] http://www.writely.com
[4] http://desktop.google.com
[5] http://desktop.msn.com

TeNDaX stands for TeXt Native Database extension and stores document data natively in the database [7]. The TeNDaX system is a collaborative database-based real-time editor [13]. Every text manipulation operation is ultimately represented as a interactive transaction [6] and all changes made by an editor are immediately stored persistently in the database and propagated to all clients working on the same document [14]. Document creation meta data is automatically gathered during text editing and stored along with the document in the database [8]. It can be used for sophisticated document management, information retrieval and data lineage functionality. TeNDaX is our current prototype for storing documents in the database, as specified in section 2. However, the document model as well as performance needs to be improved. We will use this prototype as a benchmark for measuring the new implementation to be undertaken as part of PhD work.

4 Problem Areas/Gaps

None of the approaches presented in section 3 store document data in the database in an optimal fashion, nor do they offer rich functionality for retrieving and manipulating these documents.
We identified the following problem areas:

- Text is not stored entirely in the database yet. We need to store a document entirely and comprehensively, including content, structure and meta data. It is not clear what the optimal text storage should look like.
- Text stored in the database is most commonly not only queried but also updated. Therefore, statements for text manipulation should be provided: SQL should be extended in order to provide statements for creation (and removal) of documents, insertion and removal of characters and changes to document structure.
- Document meta data, i.e. data about the document creation process, is currently not gathered automatically (even in systems presented by Dumais [5] there is too little meta information). Current (non-database based) word-processing applications gather a very limited and incomplete record of document creation process meta data. By storing document data in the database, meta data can be stored as part of the document and can later be used for functionalities like data lineage or document management.
- Text retrieval should not only be based on content but also on structure and meta data [2].

5 Approach - Research Plan

The aim of this work is the design, specification, implementation and evaluation of a document data type in one of the available database systems. There are several goals to achieve. The first goal consists of a detailed definition and

specification of the data type DOCUMENT (consisting of content, structure and meta data). Based on that, an appropriate document model will be elaborated. The second goal is the implementation of the specified data type DOCUMENT in a database system and the provision of a query language for that data type (SQL Extension). Then, the model and the implementation will be evaluated.

Approach To achieve the goals, the following steps are planned: a conceptual modelling of the new data type; the specification of the SQL extension in terms of primitives and more complex text based operations; the implementation of the data type and the SQL extension in a database system; and evaluation in terms of testing, performance measurement and user satisfaction evaluation.

6 Collaborative Datatype DOCUMENT

The storage of documents consisting of content structure and meta data which should be accessible concurrently by a number of users in a database brings along a lot of unanswered questions. The most appropriate document model should define the optimal granularity of text objects, connections, relationships and dependencies between these objects as well as between content, structure and meta data. The model should be scalable and extensible and provide optimal access for fast retrieval and concurrent manipulation.
We will examine the TeNDaX data model and in-memory document models used by word-processing applications. Based on that, a native document model will be defined which stores a document including content, structure and meta data in an optimal way supporting fast, efficient, and concurrent access in terms of query and manipulation.

7 SQL Extension

SQL has to be extended to support the new data type DOCUMENT. The extension should provide statements that cover text editing functionality such as inserting and removing characters and structuring the document.
We first need to extend the SQL Data Definition Language. Statements for creating, altering and deleting a document, i.e. an extension of the common SQL CREATE, ALTER and DELETE statements. We will have to define what options are allowed at creation and alteration time. Those will refer to what we see as constituent parts of a document (meta data such as structure, layout, notes, semantics and security).
Secondly, we will provide extensions to the Data Manipulation Language (DML). Statements for inserting and deleting characters (INSERT CHARACTERS and DELETE CHARACTERS) and for applying structure (CREATE ZONE) will be needed. We will also extend meta data manipulation functions in DML.
Thirdly, we will extend the Data Retrieval Language. Statements for retrieving or searching for a document or parts of documents based on content, structure and meta data (i.e. an extended SELECT statement) are required. These will

have to support a variety of pattern matching functions combined with structure matching.

8 Choice of Platform and Evaluation Strategy

We have so far experimented with the Caché Database System[6] [7]. The selection of the future experimental platform will be based on a comparison of existing DBMSs with regard to extensibility functions offered by each of the systems. The implemented data type and the SQL extension will be tested and evaluated regarding performance of concurrent data access and manipulation, and user satisfaction.

9 Conclusion

We here presented a proposal for extending database technology. We outlined the need of storing document data in a database and presented requirements and possible approaches for the storage, retrieval and manipulation of text documents within a database. The new collaborative document data type will store documents entirely including content, structure and meta data. SQL will be extended to support efficient data definition, manipulation and retrieval of document data stored in the database. Doing so, document data will be brought to the same level as structured business data by providing enhanced support through database technology.

References

1. S. Abiteboul et al. The lowell database research self-assessment. *Commun. ACM*, 48(5), 2005, 111-118.
2. Ricardo Baeza-Yates and Gonzalo Navarro. Integrating contents and structure in text retrieval. *SIGMOD Rec.*, 25(1), 1996, 67–79.
3. Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
4. Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. *ICDT 2001*, 316–330.
5. Susan Dumais et al. Stuff I've seen: a system for personal information retrieval and re-use. *SIGIR 2003*, 72–79.
6. Thomas B. Hodel and Klaus R. Dittrich. A collaborative, real-time insert transaction for a native text database system. *IRMA 2004*.
7. Thomas B. Hodel and Klaus R. Dittrich. Concept and prototype of a collaborative business process environment for document processing. *Data & Knowledge Engineering, Special Issue: Collaborative Business Process Technologies, Elsevier Science Journal*, 2004.
8. Thomas B. Hodel, Roger Hacmac, and Klaus R. Dittrich. Using text editing creation time meta data for document management. *CAiSE 2005*, 105–118.

[6] http://www.intersystems.com/cache/

9. IBM. DB2 Text Extender Administration and Programming. http://www-306.ibm.com/software/data/db2/extenders/text, 2000.
10. David Iseminger. *Microsoft SQL Sever 2000 Reference Library - T-SQL LANGUAGE REFERENCE*. Microsoft Press, 2000.
11. ISO/IEC 9075–14:2003. *Database languages – SQL – Part 14: XML-Related Specifications (SQL/XML)*, 2003.
12. ISO/IEC 9075-2:1999. *Database Language – SQL – Part 2: Foundation(SQL/Foundation)*, 1999.
13. Stefania Leone, Thomas B. Hodel, Michael Boehlen, and Klaus R. Dittrich. TeNDaX, a collaborative database-based real-time editor system - a word-processing Lan-Party. *EDBT 2006*, 1135–1138.
14. Stefania Leone, Thomas B. Hodel, and Harald Gall. Concept and architecture of a pervasive document editing and managing system. *SIGDOC 2005*, pages 41–47.
15. Zhen Hua Liu, Muralidhar Krishnaprasad, and Vikas Arora. Native Xquery processing in Oracle XMLDB. *SIGMOD 2005*, 828–833.
16. Jim Melton and Andrew Eisenberg. SQL Multimedia and Application Packages (SQL/MM). *SIGMOD Rec.*, 30(4), 2001, 97–102.
17. Gonzalo Navarro and Ricardo Baeza-Yates. A language for queries on structure and contents of textual databases. *SIGIR 1995*, 93-101.
18. Gonzalo Navarro and Ricardo Baeza-Yates. Proximal nodes: a model to query document databases by content and structure. *ACM Trans. Inf. Syst.*, 15(4), 1997, 400-435.
19. Matthias Nicola and Bert van der Linden. Native XML support in DB2 universal database. *VLDB 2005*, 1164-1174.
20. Oracle Corporation. Oracle Text Application Developer's Guide 10g Release 2 (10.2). http://download-east.oracle.com/docs/cd/B19306_01/text.102/b14217.pdf, 2005.
21. Shankar Pal et al. XQuery implementation in a relational database system. *VLDB 2005*, 1175-1186.
22. Chengzheng Sunet et al. Reduce: A prototypical cooperative editing system. *HCI International 1997*, 89–92.
23. Motomichi Toyama. SuperSQL: an extended SQL for database publishing and presentation. *SIGMOD 1998*, 584-586.
24. Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. *CIDR 2005*, 262–276.