# DBDOC: Querying and Browsing Databases and Interrelated Documents

Carlos Garcia-Alvarado
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Carlos Ordonez
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Zhibo Chen
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

## ABSTRACT

Large collections of documents are commonly created around a database, where a typical database schema may contain hundreds of tables and thousands of columns. We developed a system based on SQL code generation and User-Defined Functions that analyzes document-to-metadata links by extracting a basic set of relationships at different levels of granularities: coarse, medium and fine. Such relationships are then stored and queried in the DBMS, allowing the user to explore, query, and rank how columns and tables are related to users and applications. At the same time, our system provides typical information retrieval capabilities for querying medium-sized document collections of interrelated documents in the DBMS, with an acceptable performance.

## Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Data warehouse and repository*

## General Terms

Management, Documentation, Design

## 1. INTRODUCTION

The information retrieval (IR) world and the database (DB) world have focused their research on different areas. A database schema contains structure and hierarchy. Hence, the data stored in a database is associated with an explicit meaning, as opposed to the data spread throughout in the unstructured world. The semi-structured data, in relation to a database schema, has become valuable because of the ability to describe this structured data. Managing semi-structured sources, such as documents, text files, web pages and spreadsheets is challenging when compared to working with the self-described structured data in a DBMS. As a result, linking this information is also challenging, but obtaining such relationships can be widely used to describe and obtain hidden knowledge. We decided to build a novel model integrating the approaches from [3, 1], in a system that explores such relationships and ranks or retrieves them using in-database IR capabilities and speedup techniques (Rank-Join approach for top-k querying). We believe that


http://doi.acm.org/10.1145/1557670.1557688

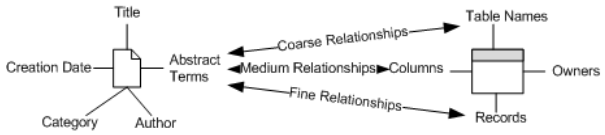


**Figure 1: Relations.**

using a relational database as a central repository for storing the metadata is the right approach for integrating these two worlds, and allows the user to find unknown relationships that are not the result of a simple keyword matching. We tested our application with a scientific database containing water pollution data and a collection of documents from the Texas Commission on Environmental Quality to allow the user to answer non-trivial questions such as, "Which tables are more related to arsenic?" or "Which authors use table X more?" Our motivation for developing this novel approach is to find a reduced set of relationships in complex database schemes which can be stored, used, and efficiently managed in a DBMS, instead of using the original collection.

## 2. SYSTEM TECHNICAL DESCRIPTION

Structured data is represented as tables, columns and fields in a database schema, where the fields represent the content of the columns in a database. The unstructured or semi-structured data represents all the words contained in any external source of the database. All the words are then treated without any hierarchy, and represent just a "bag of words" related to a source. Additionally, the document's metadata, such as author and title, was extracted, for matching additional relationships with this information. All of this information was then stored in the database and ranked based on the results of its relationships using the vector space model (VSM). The metadata found in the structured and the semi-structured data were matched, and three types of relationship were generated: coarse-relationships, medium-relationships, and fine-relationships. These relationships differ in the level of scope in which they match these terms (see Figure 1). The coarse-relationships are based on describing a relationship between a document and the title of a table in the schema. The medium-relationships describe the level of relationship between the terms in the document and the columns in the schema. The fine- relationships are the most complex type of relationships, and they are the result of matching the content of the structured data and the content of each semi-structured source. Once

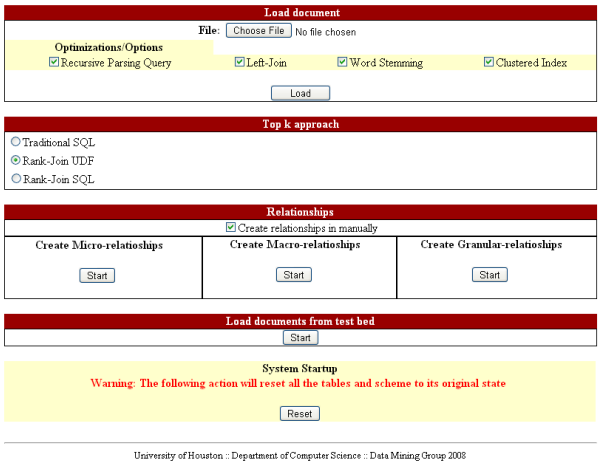


**Figure 2: Relationship discovery.**

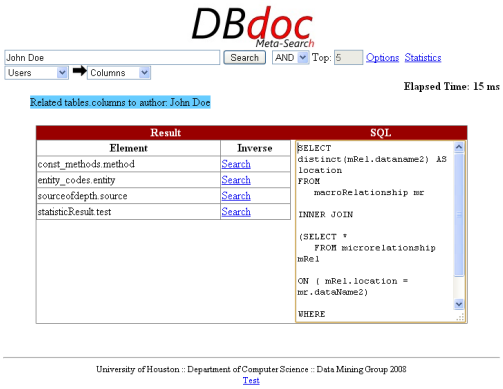


**Figure 3: Query results.**

we computed these relationships, we had to adjust the traditional document VSM to work with our relationship model, but we kept the VSM for traditional IR searches in the documents' abstracts. All the computations, such as the cosine similarity formula, the inverse frequency, or the document frequency are related exclusively to a relationship type, and this implies having separate term weights for each. The VSM framework in SQL for traditional searches is detailed in [1]. The application was developed using SQL statements and in-database extensions, such as User-Defined Functions (UDFs) for stemming, SQL statements for cleaning, storing and obtaining the relationships, and recursive queries for parsing and ranking [1]. The ranking function retrieves the top-k relationships using the Rank-Join algorithm [2].

## 3. SYSTEM DEMONSTRATION

The system was built as a Web Application, and developed entirely within the DBMS. The implementation in SQL is justified because it allows the system to be portable to multiple DBMSs [1], has the ability to manage these relationships efficiently, and takes advantage of optimized queries and algorithms to manage fast retrieval, indexing, and clustered storage. The system works in a client-server architecture, where all processing is performed in the server, and the front-end is used exclusively for retrieving the information to the end-user, and querying. Therefore, the front-end is a thin layer of SQL generation, which connects to the DBMS using ODBC. We also used recursive queries for parsing documents, a UDF for stemming, and indexed the term tables. We clustered the term tables with secondary indexes on the term name to speedup the discovery of relationships. The front-end was developed with the ASP .NET framework. The user interface allows searching documents utilizing the relationships involving Users, Tables, Documents, Columns and Fields, as well as the document abstracts. The output shows the results that are part of the database schema (e.g. columns), the results that are part of the semi-structured world (e.g. authors) and the SQL associated to obtain those results. The front-end also allows the user to define the settings of the application, such as the document loading options, on-demand relationships search, and the top-k implementation (traditional, UDF, SQL).

For the demonstration, our application will be tested using a pre-processed collection of 1000 documents. It will be separated into two parts: (1) creation and search of relationships and (2) navigating interrelated documents, top-k selection, exploration of the relationships, and tracing performance. In the first part (see Figure 2), the user will see how relationships are created between unstructured documents and structured database tables. Once the relationships are created, the user can query these relationships with terms to find the best matching documents. The users are provided the option of searching for multiple terms conjunctively or disjunctively. In the second part of the demonstration (see Figure 3), the user will be able to navigate and explore all types of relationships to gain information about the interrelated documents and the database not available through searching. It will be shown that by forming relationships between the unstructured and structured data, it is possible to obtain additional knowledge that was previously unavailable. Starting with the results of a search, it is possible to navigate to other relationships. As an example, the user will first search for a keyword in our document collection. With the top-k ranked results, it is possible to find information such as all documents written by the same author, all documents that reference the same database table, all authors who used a particular table, and all tables referenced by a particular document. Now, the user will have experimented with the type of information available from coarse relationships. It is also possible to combine the different levels of relationships and obtain more information. With such cross referencing, the user can find all columns used by a particular author, all documents referencing a specific column, and all columns referenced in a specific year. As we can see, such knowledge cannot be obtained using a typical search.

## 4. REFERENCES

[1] C. Garcia-Alvarado and C. Ordonez. Information retrieval from digital libraries in SQL. In *ACM WIDM Workshop*, pages 55–62, 2008.

[2] I.F. Ilyas, G. Beskales, and M.A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):1–58, 2008.

[3] C. Ordonez, Z. Chen, and J. García-García. Metadata management for federated databases. In *ACM CIMS Workshop*, pages 31–38, 2007.