From: *Proc. 3rd Int. Conf. on Information Integration and Web-based Applications & Services iiWAS 2001,* Linz, Austria, W. Winiwarter, St. Bressan and I.K. Ibrahim (eds.), Österreichische Computer Gesellschaft, 2001, pp. 389-394.

FOOLPROOF QUERY ACCESS TO SEARCH ENGINES

Bert Bakker and Johan ter Bekke [1])

Abstract

In order to support queries on the content of HTML documents and their interconnections, we propose to extend search engines with a visible data model of their index accessible via the Xplain language. Contrary to SQL this language only accepts queries strictly respecting the structure of the addressed data model. This language does not allow the specification of join operations. Consequently, malicious users are not enabled to execute extremely complex queries, possibly leading to a severe performance degradation of the attacked system. We show why in open systems as the WWW - with so many users - queries should not be specified in terms of SQL, but in terms of the Xplain language.

1. Introduction

Since its invention in 1989, the World Wide Web has expanded to more than 1300 million HTML-documents and many search engines where developed since 'Mosaic' was developed in 1993 [10]. In order to offer an acceptable quality of service, search engines apply some kind of indexing, skip irrelevant words, calculate term weights depending on their appearance in titles, subtitles, URL's, meta tags or anchor text [11]. Designers have made certain decisions on the kind of intelligence and the degree of end-user support they wanted to offer by their software product. For example, the relevance of documents can be calculated on the basis of relative term frequency per document, on the basis of the number of references to a document, or both. Because of the built-in intelligence, search engines do not enable users to really manipulate their search criteria except that they can specify search terms, possibly combined with operators such as "AND", "OR" and "NEAR".

[1]) Faculty of Information Technology and Systems, Delft University of Technology,
P.O. Box 356, 2600 AJ Delft, The Netherlands
E-mail: {J.A.Bakker, J.H.terBekke}@its.tudelft.nl

Although built-in intelligence and a presentation of retrieved documents ordered by their calculated relevance is essential for a world-wide usability of the WWW, most search engines do not offer sufficient support for users wanting to retrieve documents on the basis of their own, possibly more complex criteria. This need was recognized already by others and has lead to SQL-like [9] languages as WebSQL [1] and Squeal [13]. The last approach offers more functionality than the first one because it applies a more detailed model. However, SQL-like languages also enable malicious users to specify multiple join-queries with a complexity possibly leading to a 'denial of service' situation. In order to illustrate this problem, section 2 proposes a model for the index of a search engine, whereas section 3 shows two examples of index-queries as a preparation to intelligent document retrieval.

2. A data model for the index of search engines

A data model of the index of a search engine is the basis for supporting queries about documents, their location, links between documents and words occurring in documents. Because of weight calculations, the sort of a term is also important: is the term present in a title, subtitle, anchor text, or after a HTML meta tag (for example 'keywords' or 'description')? This leads us to propose the model of an index, shown in figure 1 that should be visible to end-users.

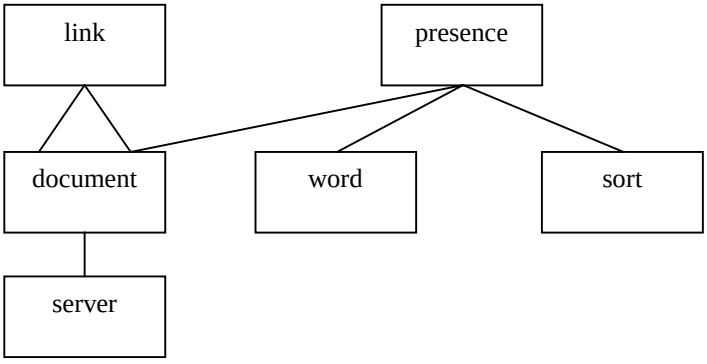

Figure 1. An index model for a search engine

Figure 1 shows an abstraction hierarchy derived from the following semantic definition:

type server =	IP_address.	{instances of 'server' identified by a server name}
type word =	string.	{instances of 'word' identified by an integer}
type sort =	description.	{instances of 'sort' identified by an integer}
type document =	server, doc_name.	{URL: concatenation of 'server name' and 'doc_name'}
type presence =	word, document, sort, relative_frequency, letter_size, font.	
type link =	from_document, to_document.	

This semantic model can be translated into an equivalent relational model:

server (server-name, IP-address);
word (word#, string);
sort (sort#, description);
document (doc#, server-name, doc-name);
presence (pres#, word#, doc#, sort#, relative-frequency , letter-size, font);
link (link#, from-doc#, to-doc#);

In order to be comprehensive, this model ignores links incorporated in pictures and does not enable us to distinguish servers by domain ('.com', '.org', etc.) or country ('.at', '.nl', etc.). We do not show the value domain (INT, CHAR, etc.) of attributes and types because they are not relevant here. Information about Xplain, its concepts, language and applications can be found elsewhere [2, 3, 6, 7, 12, 14-19].

3. Index queries

Query 1:

Select the documents where the word "Tjoa" occurs less frequently than the word "Wagner".

Xplain:

```
extend document with T_number =  total presence its relative_frequency where word its string = "Tjoa"
                                 per document.
extend document with W_number = total presence its relative_frequency where word its string = "Wagner"
                                 per document.
get document its server, doc_name where T_number < W_number.
```

SQL:

```
CREATE VIEW tjoa (doc#, T-number) AS
SELECT doc#, SUM (relative-frequency)
FROM document d, presence p, word w
WHERE d.doc# = p.doc# AND p.word# = w.word# AND string = "Tjoa"
GROUP BY doc#;
UNION                                    {otherwise the view does not contain docs without "Tjoa"}
SELECT doc#, 0 FROM document
WHERE doc# NOT IN ((SELECT doc# FROM presence
                    WHERE word# IN (SELECT word# FROM word WHERE string = "Tjoa" ));

CREATE VIEW wagner (doc#, W-number) AS
SELECT doc#, SUM (relative-frequency)
FROM document d, presence p, word w
WHERE d.doc# = p.doc# AND p.word# = w.word# AND string = "Wagner"
GROUP BY doc#;                           {here, because of the search condition, a union is superfluous}

SELECT server_name, doc-name FROM tjoa t, wagner w
WHERE t.doc# = w.doc# AND T-number < W-number;
```

The applied union avoids a pitfall of SQL: if an empty subset participates in a join this leads to disappeared information [16]. An outer join cannot solve this problem because in the absence of data this join does not produce an integer ‘0’, but ‘NULL’ [8].

Query 2:

Select the documents, which are referenced more than 30 times directly and more than 800 times indirectly via two links.

Xplain:

```
extend document with number1 =    count link per to_document.
extend document with number2 =    total link its from_document its number1 per to_document.
get document its server, doc_name where number1 > 30 and number2 > 800.
```

SQL:

```
CREATE VIEW direct (doc#, number1) AS
SELECT doc#, COUNT (*) FROM document d, link l
WHERE d.doc# = l.to-doc#
GROUP BY doc# HAVING COUNT (*) > 30;          {condition reduces the view cardinality}

CREATE VIEW indirect (doc#) AS
SELECT doc# FROM direct dir, link l
WHERE dir.doc = l.to-doc#
AND l.from-doc# IN (SELECT doc# FROM direct
                    GROUP BY doc# HAVING SUM (number1) > 800;

SELECT doc#, server-name, doc-name
FROM document WHERE doc# IN (SELECT doc# FROM indirect);
```

More complex queries could combine some of the conditions shown above.

4. Discussion

Due to the enormous number of data about HTML-documents stored in an index system, a relational implementation of index-queries will be difficult to optimize. We cannot enforce that users specify queries with a good performance, for example by applying well-chosen views. However, a more serious problem is that malicious users can attack search engines by extremely expensive queries such as:

```
SELECT doc#, server-name, doc-name
FROM document d, presence p, word w, sort s, link l
WHERE d.doc-name = s.description;          {additional problem: this condition ignores the structure of the index}
```

These problems can be avoided by the Xplain approach [19]: based on inherent structuring through attributes connecting two types (*type* attribute = composite_type, type, kind.), the '*its*' construct is available [14]. Therefore it is easy to enforce that queries only apply existing paths in data models such as 'presence *its* document *its* server *its* IP-address'. This has two consequences:

1. Joins are superfluous and forbidden in Xplain.
This implies a defense against malicious users. The only alternative is to traverse attribute paths. For example: *extend* document *with* number1 = *count* link *per* to_document.
Here the applied path is the attribute 'link *its* to_document'.

2. Queries in Xplain have an almost linear complexity.
For example, the complexity of deriving the attribute 'document *its* number1' in query 2 is proportional to the sum of the cardinalities of the composite types 'link' and 'document'. This also results into a defense against malicious users: they cannot specify any query leading to a denial of service.

Xplain further reduces the costs of many queries by using attribute files (transposed files [4]). It does not apply further optimization; it only uses one primary index (B^+-tree [5]) per composite type (containing identifiers of instances of a composite type). Still, the Xplain-DBMS executes up to 1000 times faster than a commercial relational system. This performance difference increases with query complexity and database size. Due to the allowed paper size, we cannot deal with improvements, such as the application of RAID-systems [12], or semantics based data distribution [2, 3].

It is also possible to avoid SQL pitfalls by applying an automatic transformation of semantic specifications (data definition and data manipulation) into SQL specifications [6, 7]. Further, the proposed model can be extended and/or modified; for example more details enclosed in URL's and documents can be extracted and made visible in the model.

5. References

[1] AROCENA, G.O., A.O. MENDELZON and G.A. MIHAIILA, Applications of a Web Query Language, in Proceedings of the Sixth Int. World Wide Web Conference, Santa Cruz (1997).
http://www.cs.toronto.edu/~websql/www-conf/wsql/PAPER267.html

[2] BAKKER, J.A., A Semantic Framework for the Design of Data Distribution Schemes, in Proceedings 11th International Workshop on Database and Expert System Applications, DEXA 2000, Greenwich, UK, A.M Tjoa, R.R. Wagner and A. Al-Zobaidie (eds.), IEEE Computer Society, 653-660 (2000).

[3] BAKKER, J.A., Semantic Partitioning as a Basis for Parallel I/O in Database Management Systems, Parallel Computing, 26, 1491-1513 (2000).

[4] BATORY, D.S , On Searching Transposed Files, ACM Transactions on Database Systems, 4, 531-544 (1979).

[5] BAYER, R. and K. UNTERAUER, Prefix-B-trees, ACM Transactions on Database Systems, 2, 11-26 (1977).

[6] BOER, Berend de, and J.H. TER BEKKE, Applying semantic database principles in a relational environment, in Proceedings IASTED International Conference on APPLIED INFORMATICS, Symposium.1, Artificial Intelligence and Applications, Innsbruck, M.H. Hamza (ed.), ACTA Press, 400-405 (2001).

[7] BOER, Berend de, Xplain2SQL: http://www.pobox.com/~berend/xplain2sql/index.html (2001).

[8] CONNOLLY, T.M., C.E. BEGG and A.D. STRACHAN, Database Systems, A Practical Approach to Design, Implementation and Management, Addison-Wesley, 428-431 (1996).

[9] ELMASRI, R. and S.B. NAVATHE, Fundamentals of Database Systems, 3rd edition, Addison-Wesley (2000).

[10] HTML-history: http://www.w3.org/MarkUp/#historical

[11] LU, G., Multimedia Database Management Systems, Artech House (1999).

[12] MELNIK, S., S. Raghavan, B. Yang and H. Garcia-Molina, Building a Distributed Full-Text Index for the Web, in Proceedings of the Tenth International World Wide Web Conference, Hong Kong (2001). http://www10.org/cdrom/papers/275/index.html

[12] ROLLAND, F.D., The Essence of Databases, Prentice Hall (1998).

[13] SPERTUS, E. and L.A. Stein, Squeal: A Structured Query Language for the Web, in Proceedings of the Ninth International World Wide Web Conference, Amsterdam (2000). http://www9.org/w9cdrom/222/222.html

[14] TER BEKKE, J.H., Semantic Data Modeling, Prentice Hall (1992).

[15] TER BEKKE, J.H., Meta Modeling for End User Computing, in Workshop Proceedings 6th International Conference on Database and Expert Systems Applications, DEXA'95, London, N. Revell and A. Min Tjoa (eds.), 267-273 (1995).

[16] TER BEKKE, J.H., Can We Rely on SQL?, in Proceedings 8th International Workshop on Database and Expert Systems Applications DEXA'97, Toulouse, R.R. Wagner (ed.), IEEE Computer Society, 378-383 (1997).

[17] TER BEKKE, J.H., Advantages of a Compact Semantic Meta Model, in Proceedings Second IEEE Metadata Conference, Metadata 97, Silver Spring, Maryland (1997). http://www.computer.org/conferen/proceed/meta97/list_papers.html

[18] TER BEKKE, J.H., Manual of the Xplain-DBMS, version 5.8 (in Dutch), Delft University of Technology (1999). http://is.twi.tudelft.nl/dbs/terBekke.html

[19] TER BEKKE, J.H., Semantic requirements for databases in casual environments, in Proceedings SAICSIT'99: Prepare for the New Millennium, Johannesburg, P. Machanick (ed.), Electronic extension of the South African Computer Journal, 24 (November, 1999). http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT~99/papers_ideas.html