UNIVERSITÄT LEIPZIG

Institut für Informatik

Evaluation of object-relational database systems for fulltext retrieval

Erhard Rahm

Report Nr. 6 (1998)

ISSN 1430-3701

Evaluation of object-relational database systems for fulltext retrieval

Erhard Rahm
Univ. Leipzig
rahm@informatik.uni-leipzig.de

May 1998

Abstract

Object-relational database systems add object-oriented features to relational DBMS and allow the DBMS's functionality to be extended to new application domains. For the important domain of fulltext retrieval and document management, we analyze whether current object-relational DBMS are already able to compete with specialized information retrieval (IR) systems. After discussing the main requirements, we present a comparison of the ORDBMS Informix and several text data blades with the information retrieval system Fulcrum and a relational DBMS extended by IR functionality (Oracle ConText). A qualitative and quantitative evaluation is presented considering the degree of IR functionality, performance, and retrieval quality.

Key words:
Object-relational databases, fulltext DBMS, information retrieval, document management

1 Introduction

Object-relational database systems [St96] extend relational DBMS (RDBMS) by object-oriented features, such as support for user-defined data types and functions, reference attributes, inheritance, and polymorphism. Furthermore, type constructors such as array, set, tuple, etc. are typically provided facilitating the construction of complex objects. Object-relational DBMS (ORDBMS) thus offer a huge increase in expressive power and flexibility over the traditional relational approach. They are particularly appealing for the many advanced database applications, such as CAD/CAM, geographic information systems, document management, multimedia databases, etc., for which relational DBMS have experienced their limitations. In contrast to object-oriented database systems, ORDBMS support all features of relational DBMS so there is no need to change existing RDBMS applications. In particular, they are compatible with the SQL standard. Language support for the new features of ORDBMS will be included in future versions of the SQL standard (SQL3, SQL4).

A main advantage of ORDBMS is their extensible type system permitting to add new application-specific data types and functions. ORDBMS vendors are using different names for such type extensions, e.g., data blades (Informix), extenders (DB2), or data cartridges (Oracle). Typically, extensions are supported for various multimedia data types (video, audio, image, ...), text, spatial data, time series, etc. The functions provided with the type extensions can be used within SQL queries, as in the following example

```
SELECT p.name
FROM person p
WHERE   contains (p.resume, "Java", relevance) AND p.age  < 40
ORDER BY relevance DESCENDING
```

The person table is assumed to have a text attribute "resume" for which the contains function can be used to determine whether it contains a given search term. Thus the query finds all persons having the word "Java" in their resume and having an age of less than 40. It is also assumed that the contains function returns a relevance parameter that is used to rank the results so that the persons with a resume most relevant with respect to "Java" appear first in the result list.

A key factor whether ORDBMS can really succeed is their performance. Despite the much higher complexity of data and queries, users expect short response times for their database operations. In order to achieve acceptable performance, it is imperative that the type extensions are not merely provided by an add-on layer to a RDBMS, but that they are closely integrated with the key DBMS components such as the query optimizer, record and index manager, buffer manager, and external storage manager. Furthermore, new index structures

and intra-query parallelism should be supported. Otherwise, most queries would require sequential table scans with typically unacceptable response times due to enormous data volume. To support such an integration, ORDBMS typically provide special APIs (application programming interfaces) for type extensions that must be used by their developers.

So far, only few information is available on the performance of commercial ORDBMS. In [As97], the so-called Bucky benchmark was proposed to compare the performance of object-relational and relational DBMS. While the benchmark was tailored to utilize object-relational features such as type constructors, navigational access over reference attributes (rather than joins), user-defined functions etc., the performance of the considered ORDBMS was almost twice as slow as for the relational DBMS. The benchmark, however, used a very simple and conventional database application so that the ORDBMS suitability for an advanced application was not addressed.

In this paper, we evaluate the use of ORDBMS for fulltext retrieval. Due to the explosive growth in the number of documents accessible in the internet and within intranets, the need for efficient and effective fulltext retrieval has increased enormously. Fulltext retrieval is a variant of information retrieval (IR) for which many commercial systems exist. While such IR systems typically provide good support for vague retrieval queries on largely unstructured data like texts, they lack the capabilities of DBMS. In particular, they typically do not support queries over structured data (e.g., tables with attributes), transaction processing (concurrency control, logging, recovery), referential integrity, views, parallel query processing, etc. However, there is an increasing need for combined IR and DBMS functionality, in particular for queries over both unstructured and structured data (as in the above example where the resume text is evaluated and attributes like age or name are accessed). The integration of IR and DBMS functionalities can be provided by IR systems extended with DBMS functionality, by RDBMS extended with IR functionality, or by ORDBMS with type extensions for text and IR. The ORDBMS approach promises a seamless integration of the new functionality as well as the possibility to use it in combination with other type extensions, e.g., for multimedia data.

Our evaluation concentrates on one of the leading ORDBMS, namely Informix Universal Server and various text data blades. For comparison purposes, we have also evaluated a commercial IR system, namely Fulcrum, which already includes some DBMS functionality. In a previous evaluation of various information retrieval systems [Me96] conducted within the German digital library project MeDoc [En97, Me98], Fulcrum has been chosen as the best system. That evaluation, however, did not include object-relational DBMS. In our comparison, we also consider a RDBMS extended with IR functionality, namely Oracle 7.3 with the ConText option.

The rest of the paper is organized as follows. In section 2, we briefly review the main requirements for fulltext retrieval. Section 3 provides an overview of the evaluated systems, namely Fulcrum, Informix Universal Server and several text data blades, and Oracle. This section also contains a qualitative comparison of these systems with respect to the text retrieval functionality. In section 4, we then present a quantitative comparison. Finally, we conclude.

2 Text retrieval

The field of information retrieval [SG83, Sa89] is relatively mature since it has been investigated for more than 30 years. Traditionally, the document search is based on bibliographic data (author, title, year, etc.) and the key words and index terms associated with the documents. Unfortunately, this approach requires a time-consuming manual classification and indexing of documents which is no longer adequate in the case of digital libraries in the internet or within intranets. The sheer data volume which is rapidly increasing renders the manual approach impracticable. Fulltext retrieval is a partial solution to the problem as it can consider all words found in a document to answer retrieval queries without prior manual classification*.

The goodness of an IR system is determined by many factors, including retrieval quality, performance, functionality, ease of use and administration, cost, etc. A good *retrieval quality* requires that the most relevant documents are found for a given query. This quality is typically measured by the recall and precision metrics. For a given text collection and query set, *recall* determines the fraction of the relevant documents that are found by the system, while *precision* indicates the fraction of the found documents which are relevant. If r indicates the number of relevant documents for a query set, f the number of documents found for the query set, and fr the number of found and relevant documents, then it holds that

$$recall = fr / r, \quad \text{and} \quad precision = fr / f.$$

It is difficult to achieve a high value (close to 1) for both metrics together. If only save candidates are found (high precision), then many relevant documents may be missed. On the other hand, if many potentially relevant documents are returned precision may be poor.

Performance refers to the response times and processing overhead for retrieval queries. Furthermore, the system should scale to very large document collections and many users. In addition, indexing time and the storage overhead for index information should be low etc. *Functionality* can directly impact retrieval quality and performance, e.g. depending on the used retrieval model, indexing, and ranking approaches. Other important features include

* In general, it cannot be guaranteed that all relevant documents are found due to the use of specific vocabularies, synonym/homonym problems etc. These problems are also apparent for internet search engines.

relevance feedback and text operations like stemming, similarity search (e.g., based on soundex metric), proximity search, wildcards/truncation, thesaurus, etc.. Desirable features not directly impacting retrieval quality and performance include support for many text formats, multiple languages, many hardware/software platforms, interfaces (SQL, Z39.50, HTTP, ODBC, ...), result highlighting etc.

Definition and discussion of the various features is beyond the scope of this paper but can mostly be found in standard text books on information retrieval. We only stress some particularly important aspects.

- *Retrieval model*
  Mostly, a simple *Boolean retrieval model* is supported where queries consist of atomic expressions connected with the three logical operators "and", "or", and "not" (e.g. find documents with "author=Gray" and "subject=databases"). This model results in a binary decision whether or not a document qualifies for a query and only the documents which fully qualify are included in the result. Typically, much better results are achieved with a so-called *vector space model* [SG83, Sa89]. This model represents both the documents and the queries as weight vectors. Given a query, the documents are ranked according how "similar" their corresponding vectors are to a given query vector. More recent models like *probabilistic retrieval* [Fu93] are not yet supported by commercial systems.
- *Ranking* is a must for any IR system to order the documents of a result list according to their relevance. This is to allow quick access to the most important documents even for large result lists, obtained due to the vagueness of queries. The key issue is how the system determines the relevance of documents for a query. Simple approaches just consider the frequency of index terms within a document, while sophisticated schemes rely on comprehensive statistics and other factors (e.g., by giving higher weight to term occurrences within title of documents, abstract etc.). The weight vectors of the vector space model directly support ranking, but ranking is also possible for boolean retrieval models.
- *Relevance feedback* allows the user to iteratively refine a query based on the relevance of previous query results. This manual interaction often leads to significantly improved retrieval quality.
- *Indexing*
  All relevant terms and key words of the documents are maintained in an index, represented as an inverted list. The index allows to quickly locate all relevant documents for a given search term. For fulltext retrieval, all words of a document are considered, possibly with the exception of words maintained in a *stop list.* Some systems maintain statistics for indexed terms like their frequency and their exact position within a document in order to improve retrieval quality.
  Indexing is important for retrieval performance and quality. On the other hand, it re-

quires a substantial storage overhead and update cost. It should be possible to index new documents incrementally without having to re-index the entire document collection.

In addition to these requirements, we demand support for both IR and DBMS functionality. In particular, the SQL query language should be supported with the possibility to access both structured data (attributes) and unstructured data (text). Furthermore, it should be possible to store documents not merely in the file system but within the database in order to protect them with transaction mechanisms (concurrency control, recovery), to enable parallel processing on them, etc. It is also desirable to support an internal structuring of documents in order to refine queries and query results to document parts, such as individual chapters.

3 Overview of fulltext DBMS and qualitative comparison

For our comparison, we consider the IR system Fulcrum, the object-relational DBMS Informix with four text data blades, and the relational DBMS Oracle 7.3 with an integrated fulltext component. The described functionality refers to the versions as of 1997.

3.1 Fulcrum

Fulcrum is a well-established information retrieval system with some DBMS functionality [Fu96]. In particular, it follows the relational data model as all information is organized in tables and an SQL dialect is supported. Each row in a document table refers to exactly one document. However, the documents are not stored within the database but are kept in their original format as separate files. The names of these files are kept in a system-defined column of a document table. Additional columns (attributes) can be specified, e.g. to maintain bibliographic information (author, title, year, etc.) to support queries on both these structured data as well as on the document contents. Access to the document files is performed by text readers which are provided for more than 150 formats. For indexing, Fulcrum reads the documents with the help of the text readers and converts them into an internal format. Indexing can be performed immediately for every new document or in batch mode. Documents can be divided into several "zones" which allow separation of text portions like title, abstract, main text, and references.

Fulcrum supports three retrieval models and four different ranking algorithms that can be specified by the user. The retrieval models are strict boolean, fuzzy boolean, and vector space model. The fuzzy boolean model also considers documents that only satisfy a subset of and-connected conditions, similar to an or-operator. This can lead to an improved recall compared to strict boolean, while the ranking should help keep an acceptable precision for the most relevant documents. Queries are formulated with an SQL dialect containing proprietary extensions for document retrieval. In addition to boolean query expressions it is

possible to specify queries with weighted search terms as well as natural language queries. Fulcrum supports a large spectrum of text operators (wildcards, proximity search, ...) and other features (relevance feedback, result highlighting, ...) as summarized in Table 1. Furthermore, multiple languages are supported. The ranking algorithms use information like the frequency of terms within documents, the frequency of different terms per document, and the frequency of terms in the entire document collection.

On the other hand, the considered Fulcrum version (V3.5) has significant SQL limitations as only a small subset of SQL-92 is supported. Moreover, not even joins are supported between tables! Furthermore, there is no support for user administration in the version considered.

3.2 Informix Universal Server

Informix Universal Server represents one of the leading object-relational DBMS in the market. It was developed by extending the relational Informix DBMS with object-oriented features and the data blade technology pioneered by Illustra, which was acquired by Informix in 1996. Compared to other ORDBMS, Informix already supports many object-oriented concepts, in particular inheritance, type constructors, overloading, encapsulation, etc. Data blades allow the incorporation of new application-specific data types and functions into Informix which can then be used in combination with the built-in DBMS functionality as well as with other data blades. Currently, more than 20 data blades are available, mostly developed by third-party companies. A DataBlade Developers Kit (DBDK) helps developing and integrating data blades. This is a complex task as a data blade must cooperate with several Informix components, in particular the SQL parser, query optimizer, query and function executor, access methods, and storage manager. It is also possible to specify parallel processing for data blade functions [Ol96].

Several text data blades are available to support fulltext retrieval. Some of them use the Informix LOB (large object) features for storing documents in the database. Similar to SQL3 [PB97], Informix distinguishes between binary LOBs (BLOBs), e.g., for storing multimedia data like images or video, and character LOBs (CLOBs) for text documents. For our evaluation, information was available for four text data blades*, for which the main features are summarized in Table 1. All data blades support boolean retrieval and ranking. Furthermore, each data blade provides a special text access structure to support indexing and proprietary language extensions for retrieval. The documents can be stored either in the database or in the file system. Unfortunately, the information available in manuals and internet documents is partially limited, in particular with respect to the details of indexing, ranking, etc.

* There are several additional text data blades available or announced, including a data blade developed by Fulcrum. Unfortunately, there is not yet sufficient information available.

Some specific remarks on the selected data blades follow.

- *TDB Text data blade*
  This was the first available text data blade already provided by Illustra [Ill94]. It offers only limited functionality as can be seen from Table 1, and supports only few text formats (Ascii, Postscript, SGML, Tex). About 30 additional formats are available by the combined use of a separate Text Conversation data blade.
- *Verity Text Search data blade* [Ve97]
  This data blade is provided by the US company Verity Inc. It supports more retrieval functions than TDB (thesaurus, similarity and proximity searches, etc., see Table 1). Retrieval queries are possible for both indexed and non-indexed text attributes. Indexing requires that documents are converted to Ascii at first. Currently, 11 languages are supported but only few (<10) text formats (support for about 130 formats is planned).
- *Excalibur Text Search data blade* [Ex97]
  The data blade is developed by the US company Excalibur Technologies [Ex97] and offers a similar functionality than the Verity data blade. Currently, only few text formats are supported.
- *PLS Text Search data blade* (developed by Personal Library Software Inc.)
  In contrast to the previous data blades, PLS Text Search [PLS96] supports a vector space retrieval model as well as natural language queries. Indexing is always immediate for new documents; only canonical forms of words are maintained in the index. Ranking is based on comprehensive statistics, in particular term frequencies per document, inverse document frequency (higher relevance for documents containing rare terms), and breadth of match (higher relevance for documents with many distinct query terms).

3.3 Oracle with ConText option

Oracle's fulltext support is closely integrated with the DBMS by a so-called ConText option. This option was already available for the relational DBMS Oracle 7.3, which we will consider here, and will also be supported by future object-relational Oracle versions.

Oracle ConText provides comprehensive support for IR functions (Table 1) in the form of specific SQL operators. Additional flexibility is gained by supporting two-step queries, where the results (hit list) of an initial query are kept within a temporary table that is then processed by a second query. This also allows relevance feedback to be achieved. Currently, only few text formats are directly supported, but there is the possibility for the user to provide filters for additional formats. Stemming is provided for six languages, while many other systems only support English in this respect.

Several indexing options exist (based on bit vector indexing) which can be specified by the user. The position of terms within documents can be recorded in the index in order to enable

fast phrase and proximity searches. Furthermore, parallel indexing is possible to reduce the indexing time for large collections. Ranking is primarily based on the frequency of terms within the documents. Documents are either stored in the database or accessed via file names or URLs if kept separately in the file system or in the intranet/internet. Documents can linearly be partitioned, e.g., to distinguish between individual chapters.

ConText features special linguistic capabilities to automatically classify English text documents and identify key themes. For instance, a document frequently mentioning interest rates and money supply could be classified as a financial document. Furthermore, so-called theme summaries can be generated which are supposed to indicate the main topics covered within a document that can be used for querying.

3.4 Qualitative comparison

The feature lists in Table 1 allows for a qualitative comparison of the six systems. As could be expected, Fulcrum excels by a strong IR functionality including relevance feedback, flexible ranking algorithms, and support for vector space model. Moreover, queries can be posed in natural language permitting a much simpler user interface than with a boolean retrieval model. Furthermore, Fulcrum currently is the only system supporting many text formats and a Z39.50 interface, which is an important standard for bibliographic databases [Z39]. Oracle also has a rather comprehensive IR functionality despite its DBMS origin. The considered text data blades, on the other hand, still miss important IR features. Relevance feedback is not supported at all, a vector space model and natural language queries are only offered by the PLS data blade. This data blade thus comes closest to the functionality of Fulcrum and Oracle.

All systems offer a WWW (HTTP) interface to their data. Moreover, they all support SQL and the combination of text retrieval and attribute queries. However, Fulcrum does not provide full-level DBMS functionality as Informix and Oracle (no joins, no user administration, etc.). Furthermore, documents cannot be stored in the database but only in the file system. This may be sufficient for many applications since it allows documents to be easily displayed in their original format and avoids extra storage overhead (except for indexing). On the other hand, the documents and their storage cannot be controlled by the DBMS. As a result they can be deleted, modified, renamed, etc. by users without DBMS intervention so that file names kept in the documents table may become invalid (no referential integrity). Furthermore, there is no transactional protection for documents (logging, locking, ...), no intra-query parallelism, no specific storage structures etc. These benefits are supported by both Oracle and Informix.

		Informix Universal Server 9.12				Oracle 7.3/
	Fulcrum	TDB	Verity	Excalibur	PLS	ConText
Version	3.5	1.2	1.1	1.1	1.1	1.1.2
Homepage	www. fulcrum.com	www.infor- mix.com	www. verity.com	www. excalib.com	www. pls.com	www. oracle.com
retrieval model	boolean (strict/fuzzy), vector space	boolean	boolean	boolean	boolean, vector space	boolean, "thematic"
ranking	+	+	+	+	+	+
indexing	+	+	+	+	+	+
relev. feedback	+	-	-	-	-	+
natural language queries	+	-	-	-	+	-
thesaurus	+	-	+	+	-	+
stemming	+	+	+	-	+	+
wildcards	+	?	+	+	+	+
similarity queries	-	-	+	+	+	+
proximity search	+	-	+	+	+	+
phrase search	+	-	+	+	+	+
stop word lists	+	+	+	+	+	+
result highlighting	+	-	+	+	-	+
document location	file system	file system or DB	file system or DB	file system or DB	file system or DB	file system or DB
document structure	+	-	-	-	-	+
combined text/ attribute queries	+	+	+	+	+	+
support of many text formats	yes	limited	limited	limited	limited	limited
interfaces	Z39.50, SQL, WWW	SQL, WWW	SQL, WWW	SQL, WWW	SQL, WWW	SQL, WWW

Table 1: Qualitative comparison of fulltext DBMS (as of 1997)

Compared to relational fulltext systems like Oracle 7.3, the text data blades have the advantage that they can be used in combination with other data blades, e.g. for images, geographical data, etc., as well as with the object-oriented features of the underlying ORDBMS (here Informix Universal Server). Object-oriented features like reference attributes and object identity are helpful to better model the internal structure of complex documents than with a linear document partitioning currently provided by Fulcrum and Oracle. This not only relates to the hierarchical chapter structure but also to cross-references within and between documents. It has been shown that the arbitrary complex structure of SGML documents can adequately be modeled by object-oriented DBMS [Bö97] so that it is to be expected that ORDBMS are similarly well suited in this respect. Currently however, such an internal struc-

turing is not yet supported by the available text data blades but would have to be modeled by the user.

A general problem for all systems is that they use proprietary SQL extensions for text retrieval. This is because there is not yet an SQL standardization for this important area. Hence, there is no compatibility between different vendors making it even more important to select the system that meets the requirements best.

4 Quantitative comparison

In this section, we present results of a quantitative evaluation for some of the systems discussed in the previous section. Results are given for Fulcrum and Informix with two data blades (Verity, Excalibur) which we could install in our department. From a related study [Or97], we can also use some comparable results for Oracle ConText. Our measurements have been performed in single-user mode on a Sun Ultra system, 200 MHz, 128 MB main memory, under Solaris 2.5.

We first look at the storage requirements of the fulltext DBMS themselves. Then we compare results for the so-called Time suite with respect to indexing time, index size and retrieval quality. Finally, we discuss additional observations for other text collections.

4.1 Storage requirements of fulltext DBMS

Table 2 shows the storage requirements for the fulltext DBMS. Fulcrum is by far the "leanest" system, while the Informix installations are three to four times as large. Significantly more resources are needed for Oracle/ConText consuming already 230 MB under NT [Or97]; Unix requirements are even higher. Of course, these large differences are mainly because of the huge resource demands of the DBMS components of Informix and Oracle. However, since most companies need a full-fledged DBMS anyway, the additional resources for the text components are less critical (the data blades typically require less than 20 MB). A specific text retrieval system like Fulcrum, on the other hand, is better suited if no or only little DBMS functionality is needed.

		Informix Univ. Server		
	Fulcrum	Verity	Excalibur	Oracle 7.3 / ConText
storage requirements	33 MB	115 MB	98 MB	>= 230 MB

Table 2: Storage requirements of fulltext DBMS

4.2 Time suite results

The Time suite is a small text collection that has been used in several IR benchmark studies, including [Me96] and [Or97]. It consists of 423 articles from the magazine Time just encompassing 1.5 MB.

We first look at the time required to build an index for this collection as well as at the index size compared to the collection size (Table 3). We see that Fulcrum is clearly the best system for both metrics. Its index size is only 54% of the collection size which is a good value for an IR system considering all words in the index (except stop words). Indexing only takes about 7 s while the text data blades require more than 1 minute (Excalibur) or even more than 1 hour (Verity)! On a slower hardware, Oracle required about 3 minutes for building the index [Or97] indicating a comparable performance to Informix with the Excalibur data blade. The very fast indexing time of Fulcrum is favored by the fact that this system is rather small and optimized for IR tasks. The larger Informix incurred a certain setup time; further, the cooperation between the data blades and the DBMS core for I/O and buffer access introduces overhead. In addition, the generated index is about 2 to 3 times as large as for Fulcrum. The very poor results for Verity are not fully explainable; however, they were confirmed for other text collections (see below). Oracle generates an even larger index than the data blades with more than 3 times the size as for Fulcrum.

		Informix Univ. Server		
	Fulcrum	Verity	Excalibur	Oracle [Or97]
index time	7,3 s	1 h : 20 min. : 20 s	1 min : 3 s	-
index size	54 %	152 %	97 %	168 %

Table 3: Indexing performance for Time suite

The Time suite defines 83 queries to be processed. Query response time was fast in all cases (about 1 s or below) and all systems. Hence, the object-relational DBMS were competitive in this important aspect as they were able to use the index thus avoiding scanning the entire document collection. Since the relevance of all documents is known for the query set, the Time suite allows measuring the retrieval quality of a system. The recall and precision metrics are determined for the n documents that are ranked first in the result list. In this way, the ranking quality is also evaluated.

All queries are formulated in English so that only Fulcrum supporting natural language queries did allow a direct evaluation of recall and precision values. For the systems based on the boolean retrieval model, the queries were approximated by a boolean expression using an Or-connection of all query words. Unfortunately, only Oracle produced meaningful results with this approach. Hence, Table 4 only shows the recall and precision values for Fulcrum

and Oracle for the first 5 and 10 results according to the ranked result list. Fulcrum achieves in all cases about 20% better results than Oracle indicating a superior retrieval quality, despite its smaller index.

	Fulcrum	Oracle [Or97]
n = 5	0.55 / 0.37	0.46 / 0.30
n = 10	0.69 / 0.26	0.60 / 0.22

Table 4: Time suite recall / precision values for the first n documents

The Verity and Excalibur data blades, on the other hand, obtained very large result sets for the queries with an identical relevance rating for the first 100 hits making it impossible to specify recall and precision for the first 5 or 10 hits. This indicates a poor ranking approach of the two data blades. Together with the other IR limitations (Table 1) we conclude that the retrieval quality of the data blades is likely to be significantly worse than for Fulcrum.

4.3 Further results

We have additionally tested the systems for two larger text collections, namely a collection of news group articles (183 MB) and for 1000 HTML pages from our WWW server. For both cases the superior performance of Fulcrum compared to the data blades was confirmed. While the index sizes for Fulcrum and the data blades were comparable for these collections (40-55%), Fulcrum's indexing times were again at least one magnitude of order better. Furthermore, Excalibur's indexing performance was again much better than Verity's. Fulcrum's indexing time for the 183 MB collection was about 22 minutes underlining its ability to scale to larger collections.

Our results indicate that fast query response times are obtained by all systems, including the ORDBMS, due to the use of an text index. In terms of retrieval quality and indexing performance, Fulcrum outperforms the two data blades by far. While the indexing performance of the Verity data blade was intolerable, Excalibur's indexing time and index size are comparable or better than for Oracle.

5 Conclusions

There is a large and increasing need for fulltext DBMS supporting both comprehensive database management and information retrieval functionality. These systems must be capable of efficiently processing queries on both structured and unstructured data. They must be able to scale to many users and large volumes of documents stored either directly in the database or externally in the file system. Object-relational DBMS are an attractive approach to satisfy these and other requirements by exploiting their extensibility to support new application domains. New functionality like for text retrieval can be added to the DBMS and be used in combination with other type extensions and built-in object-oriented features such as inheritance, complex object support, etc. Unfortunately, the text retrieval functions have not yet been standardized so that DBMS vendors offer proprietary and incompatible SQL extensions.

We have evaluated the functionality, performance and retrieval quality of the ORDBMS Informix and various text data blades in comparison with an established information retrieval system (Fulcrum) and an relational DBMS extended with an information retrieval component (Oracle/ConText). All text data blades provide basic IR support, in particular they are able to utilize a text index for fast retrieval response times. However, at present IR functionality, performance and retrieval quality of the considered data blades are substantially worse than for the Fulcrum system. These rather disappointing results are consistent with the findings for the Bucky benchmark where ORDBMS were outperformed by relational DBMS.

On the other hand, there are large differences between the various data blades and future versions should improve on the main shortcomings (ranking, vector space support, relevance feedback, natural language queries, support for many text formats, indexing performance, etc.). For instance, the Fulcrum data blade can be expected to provide similar functionality than the stand-alone Fulcrum system. Still, it will be difficult to reach the performance of specialized IR systems which are thus the primary choice for applications where no or only limited DBMS functionality is required. Parallel database processing can help ORDBMS to overcome the performance problems, e.g. for indexing, and support superior scalability to many users and very large document collections.

6 References

As97 Asgarian, M., Carey, M.J., DeWitt, D.J. et al.: The BUCKY Object-relational benchmark. Proc. ACM SIGMOD conf., 1997

Bö97 Böhm, K. et al.: Structured Document Storage and Refined Declarative and Navigational Access Mechanisms in HyperStorM. VLDB Journal 6(4), 296-311, 1997

En97 Endres, A.: Digital Libraries and Electronic Publishing Activities of German Professional Societies. Proc. IEEE Forum on Research and Technology Advances in Digital Libraries (ADL), 1997

Ex97 Excalibur Text Search DataBlade Module Users Guide Version 1.1, 1997

FB93 Frakes, W.; Baeza-Yates, R.: Information Retrieval: Data Structures and Algorithms. Prentice Hall 1993

Fu93 Fuhr, N.: A Probabilistic Relational Model for the Integration of IR and Databases. Proc. SIGIR conf., 309-317, 1993

Fu96 Fulcrum SearchServer Version 3.5 SearchSQL Reference 1996

Ill94 ILLUSTRA Text Conversation DataBlade Guide Version 1.2, 1994

Me96 Meyer, J.: Evaluation of fulltext DBMS for MeDoc. Internal report (in German), OFFIS, Univ. of Oldenburg, 1996. Available via [Me98]

Me98 MeDoc-Homepage: http://medoc.informatik.tu-muenchen.de/english/medoc.html

Ol96 Olson, M.A. et al.: Query Processing in a Parallel Object-Relational Database System. Data Engineering Bulletin 19(4): 3-10, 1996

Or97 Evaluation of Oracle 7.3 with ConText (in German). Internal reports by Grasser, I.; Papenfuß, C., FH Augsburg; and by S. Kanstinger, FH Karlsruhe, 1997. Available via [Me98]

PB97 Pistor, P., Blanken, H.M.: The SQL3 Server Interface. In: Multimedia Databases in Perspective (Eds.: P.M.G. Apers et al.), 101-116, Springer 1997

PLS96 PLS Text DataBlade Module 1.1. 1996. Description available under www.pls.com/products/dblade/wp_dblade.pdf

Sa89 Salton, G.: Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley 1989

SG83 Salton, G., McGill, M. J. Introduction to Modern Information Retrieval, Mac-Graw-Hill 1983

St96 Stonebraker, M.: Object-Relational DBMS - The Next Wave. Morgan Kaufmann 1996

Ve97 Verity Text Search DataBlade Module Users Guide Version 1.1, 1997

Z39 Z39.50 Maintenance Agency : http://lcweb.loc.gov/z3950/agency