Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features

Kyu-Young Whang, Min-Jae Lee, Jae-Gil Lee, Min-Soo Kim, and Wook-Shin Han
Department of Computer Science and Advanced Information Technology Research Center (AITrc)
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
{kywhang, mjlee, jglee, mskim, wshan}@mozart.kaist.ac.kr

Abstract

We propose the notion of tight-coupling [8] to add new data types into the DBMS engine. In this paper, we introduce the Odysseus ORDBMS and present its tightly-coupled IR features (U.S. patented). We demonstrate a web search engine capable of managing 20 million web pages in a non-parallel configuration using Odysseus.

1 Introduction

Conventional ORDBMS vendors provide extension mechanisms for adding new data types and operations to their own DBMSs. Examples are the Cartridge [7] for Oracle and the Extender [6] for IBM DB2. In these mechanisms, new data types are added by using user-defined types, and their operations by using user-defined functions. Here, user-defined types and functions are implemented through the high-level interface provided by the DBMS. We call this mechanism *loose-coupling*.

In the loose-coupling architecture, the high-level interface employed causes the following problems. First, communication overhead is incurred because operations on new data types are performed outside the core DBMS engine. Second, not all data types and operations can be implemented because the high-level interface provided is not 100% general. Third, concurrency control and recovery in fine granularity are hard to perform because low-level functions of the DBMS engine cannot be fully utilized for new data types due to the high-level interface.

In this paper, we propose the tight-coupling architecture [8] to solve these problems. In the *tight-coupling architecture*, a new data type and its operations are implemented directly into the core of the DBMS engine. This architecture has the following advantages over the loose-coupling architecture: communication overhead is minimal; no limitation on data types and operations exists; concurrency control and recovery can be done in fine granularity.

The tight-coupling architecture has been used to incorporate information retrieval (IR) features into the Odysseus ORDBMS[1] [9] that has been under development at KAIST/AITrc for 14 years. We first introduce Odysseus and the tightly-coupled IR features. We then demonstrate excellence of the tightly-coupled IR features through a web search engine implemented using Odysseus. Our demonstration system (a non-parallel configuration) stores and manages approximately 20 million web pages.[2]

[1] It consists of approximately 450,000 lines of C and C++ codes.

[2] With a parallel configuration, we can increase it to 2 billion web pages with data partitioning in 100 Linux machines. A prototype using five machines has been implemented and successfully tested.

For more details about Odysseus, refer to the full version of this paper available at http://cs.kaist.ac.kr/research/technical/Archive/CS-TR-2004-204.pdf.

2 Overview of Odysseus

Odysseus is an object-relational database management system (ORDBMS) designed to support new applications such as large-scale multimedia, information retrieval, GIS, OLAP, and data mining. Odysseus has the following features:

- support for large-scale databases
- fast bulk loading and bulk deleting
- concurrency control and crash recovery (fine or coarse granularity)
- tight coupling of IR features
  - an SQL-based query language extended for tight coupling with IR features
  - fast immediate update capability
  - fine or coarse granularity concurrency control and crash recovery on IR contents

3 Tightly-Coupled IR Features

Odysseus stores text documents such as web pages and find those that contain given keywords in support of IR. Here, text documents are stored in the form of the text type, and text IR indexes are used for finding those documents. Users can specify the database schema using these text types and text IR indexes at the same level of specifying nontext types and their indexes. Figure 1 shows the physical structure of a data record for a schema defined involving the text type, a text IR index, the integer type, and a B^+-tree index. As shown in the figure, the text type is specified just in the same way as the integer type is specified. Likewise, a text IR index is specified in the same way as is a B^+-tree index.

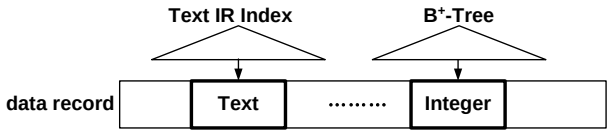

Figure 1. The structure of a data record involving the text type and a text IR index.

3.1 Fast Immediate Update Involving Text IR Indexes

Fast immediate update is the most important among the tightly-coupled IR features of Odysseus. Immediate updates are processed rapidly in Odysseus by exploiting the inverted index structure using *large objects* [2] and

subindexes. We will elaborate on this structure (patented [8]) below.

Figure 2 shows the structure of the text IR index that Odysseus uses. The text IR index is similar to the traditional inverted index structure [3], but differs in using 1) large objects to store posting lists and 2) subindexes to index postings in each posting list. We store a posting list as a large object and manage the storage space of the posting list by using the large object tree proposed by Biliris for Exodus [2]. The advantage of this method is that it is easy to insert a new posting into or remove it from a posting list. A subindex is a B^+-tree created on each large object that stores a posting list. The subindex is used for locating a specific posting with a given document identifier within a posting list. Using subindexes, we can quickly find the location of a new posting to be inserted or an existing posting to be deleted or modified.

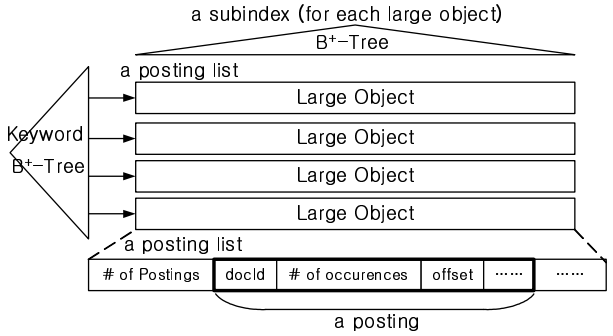

Figure 2. The text IR index structure using large objects and subindexes.

3.2 Fast Query Processing

Odysseus can efficiently evaluate 1) multiple-keyword queries and 2) queries involving both keyword and attribute conditions as well as single keyword queries. We propose the IR index join and attribute embedding techniques for this purpose [9].

IR index join is a query processing technique used to speed up multiple-keyword queries. When a multiple-keyword query (having m keywords) is processed, corresponding posting lists are m-way merge-joined. Using subindexes, the exact parts of posting lists that need to be merged can be identified. Since needless parts of the posting lists are skipped, performance is enhanced. We call this optimization technique *posting skipping*. Recent works by Guo et al. [4] and by Halverson et al. [5] use similar techniques to speed up the processing of multiple keyword queries in XML databases.

Attribute embedding is a query processing technique used to speed up queries involving both keyword and attribute conditions. It evaluates keyword and attribute conditions together by reading a single posting list, in which the values of the attributes of documents are embedded. This technique obviates the need for accessing data records stored in the main database since attribute values can be obtained by accessing only text IR indexes. Accessing the data records is very expensive since it incurs random disk accesses over a large amount of storage space. Thus, attribute embedding significantly enhances the performance. The user can specify the attributes to be embedded in a posting list when s/he defines the database schema, and the query processor can automatically find the values of the embedded attributes when executing a query.

4 Demonstration

We demonstrate a web search engine (ODYS) implemented using Odysseus to show excellence of tightly-coupled IR features. ODYS stores approximately 20 million web pages in a SUN T3+ disk array with 450 GB and runs on a SUN Enterprize 3500 server with 400 MHz CPU. We provide a web interface for querying Odysseus.

The user can limit the scope of the query to a single site. We call this feature *site-limited search*. Odysseus offers two methods for processing site-limited search. The first method handles site-limited search as a query involving both keyword and attribute conditions by embedding site identifiers into the corresponding posting list. That is, we use the attribute embedding technique. The second method handles site-limited search as a multiple-keyword query by declaring the site identifier attribute as type text and creating a text IR index on it. That is, we use the IR index join technique. Since the size of the posting list of a site identifier is typically small compared to that of a keyword, many skips occur, due to posting skipping, while doing the m-way merge-join among the posting lists. Thus, the performance of the second method is enhanced significantly. The choice between the two methods can be made by the system designer or can be automatically made by the optimizer. In this demonstration, we use the second method to process site-limited search.

References

[1] Banerjee, S., Krishnamurthy, V., and Murthy, R., All Your Data: The Oracle Extensibility Architecture, Oracle White Paper, Oracle Corp., Oracle Parkway, California, 1999.

[2] Biliris, A., "The Performance Three Database Storage Structures for Managing Large Objects," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 276–285, 1992.

[3] Faloutsos, C., "Access Methods for Text," *ACM Computing Surveys*, Vol. 17, No. 1, pp. 49–74, Mar. 1985.

[4] Guo, L., Shao, F., Botev, C., and Shanmugasundaram, J., "XRANK: Ranked Keyword Search over XML Documents," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 16–27, 2003.

[5] Halverson, A., Burger, J., Galanis, L., Kini, A., Krishnamurthy, R., Rao, A. N., Tian, F., Viglas S., Wang, Y., Naughton, J. F., and DeWitt, D. J., "Mixed Mode XML Query Processing," In *Proc. the 29th Int'l Conf. on Very Large Data Bases*, pp. 225–236, 2003.

[6] IBM, DB2 UDB Text Extender Administration and Programming, 2003.

[7] Oracle, Oracle9i Data Cartridge Developer's Guide, 2002.

[8] Whang, K., Park, B., Han, W., and Lee, Y., "An Inverted Index Storage Structure Using Subindexes and Large Objects for Tight Coupling of Information Retrieval with Database Management Systems," U.S. Patent No. 6,349,308, Feb. 19, 2002, Appl. No. 09/250,487, Feb. 15, 1999.

[9] Whang, K., "Tight-Coupling: A Way of Building High-Performance Application Specific Engines," Presented at the panel session of *Int'l Conf. on Database Systems for Advanced Applications (DASFAA)*, Japan, Mar. 2003, available on-line from http://db-www.aist-nara.ac.jp/dasfaa2003/file/Prof_Kyu-Young_Whang_5.pdf.