javax.XXL:
A Prototype for a Library of Query Processing Algorithms

Jochen van den Bercken Jens-Peter Dittrich Bernhard Seeger

Department of Mathematics and Computer Science
University of Marburg
35032 Marburg, Germany

{bercken,dittrich,seeger}@mathematik.uni-marburg.de

In our demo we present XXL (eXtensible and fleXible Library), a toolkit for rapid prototyping of query processing algorithms. XXL is a flexible, easy-to-use, platform independent Java-library. It provides a powerful collection of generic index-structures, query operators and algorithms facilitating the performance evaluation of new query processing techniques. Query algorithms in XXL use the same set of basic classes like I/O routines and improved (de)serialization methods. Therefore, XXL is an ideal testbed for experimental comparisons of new approaches to established ones.
The most important packages in XXL deal with containers, cursors and index structures. The package *containers* consists of an interface and classes for implementing (persistent) sets of objects. A container is used to manage objects with respect to their identifiers generated by the container. This establishes an abstraction from the storage medium. Containers can be buffered by wrapping them with an instance of class *BufferedContainer* which provides an m:n-relationship between buffers and containers (Figure 1). Traditional storage manager like SHORE [2] support heterogeneous objects in a (distributed) repository. However, SHORE provides neither index structures nor query algorithms. The package *indexStructures* of XXL consists of both a framework of tree-based index structures and a set of implementations. Similar to GiST [5], this framework provides a skeleton implementation for a large class of index structures. In contrast to GiST, we also provide a wide range of generic query processing algorithms like the ones for joining [1] and bulk-loading. Index structures of XXL deliver the results of queries in a cursor-like fashion respecting the open-next-close interface (ONC) as it was proposed in Volcano [4]. Therefore, index structures can easily be used in queries. A typical example is a join cursor which consumes the outputs of two underlying cursors. Most of our work is however not dedicated to the area of relational databases, but mainly refers to spatial and temporal data. For spatial databases, for example, we provide several implementations of spatial join algorithms [3]. The cursor-based processing is however the major advantage of XXL in contrast to approaches like LEDA [6] and TPIE [7]. For more information on XXL see *http://www.mathematik.uni-marburg.de/DBS/xxl*.

MOD 2000, Dallas, TX USA

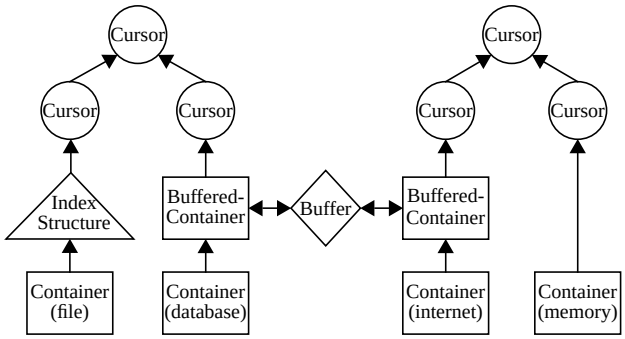

Fig. 1: Use-case of XXL

We will demonstrate the latest version of XXL using examples to show its core functionality. We will concentrate on three key aspects of XXL.

- *Usage:* We show how easily state-of-the-art spatial join-algorithms can be implemented in XXL using data from different sources.
- *Reuse:* We will demonstrate how to support different joins, e.g. spatial and temporal joins, using the same generic algorithm like Plug&Join [1].
- *Comparability:* We will demonstrate how XXL serves as an ideal testbed to compare query processing algorithms and index structures.

REFERENCES

[1] J. van den Bercken, M. Schneider, B. Seeger. Plug&Join: An easy-to-use Generic Algorithm for Efficiently Processing Equi and Non-Equi Joins. To appear in Proc. of EDBT 2000.
[2] M. J. Carey, D. J. DeWitt, M. J. Franklin et. al. Shoring Up Persistent Applications. SIGMOD Conference 1994: 383-394.
[3] J.-P. Dittrich, B. Seeger. Data Redundancy and Duplicate Detection in Spatial Join Processing. To appear in Proc. of ICDE 2000.
[4] G. Graefe. Volcano - An Extensible and Parallel Query Evaluation System. TKDE 6(1): 120-135 (1994).
[5] J. M. Hellerstein, J. F. Naughton, A. Pfeffer. Generalized Search Trees for Database Systems. VLDB 1995, 562-573.
[6] K. Mehlhorn, S. Näher. The LEDA Platform of Combinatorial and Geometric Computing. Cambridge University Press, 1999.
[7] D. E. Vengroff, J. S. Vitter. I/O-Efficient Scientific Computation using TPIE. Proc. Goddard Conference on Mass Storage Systems and Technologies, 1996, in NASA Conference Publication 3340, Volume II, 553-570.