Software Development Tools for Embedded Databases in Embedded Real-Time Systems*

Aleksandra Zagorac
Jörgen Hansson

Linköping University
Department of Computer Science
Linköping, Sweden
{aleza, jorha}@ida.liu.se

Dag Nyström
Christer Norström

Mälardalen University
Department of Computer Engineering
Västerås, Sweden
{dag.nystrom, christer.norstrom }@mdh.se

Project background

In the last years the deployment of embedded and mobile computing systems has increased dramatically. This is particularly true if one analyzes the growing market for cellular phones and the introduction of computers in various traditional mechanical areas, e.g., automotive systems. At the same time the amount of data that needs to be managed is growing. Current techniques adopted for storing and manipulating data objects in embedded and real-time systems are ad hoc, since they normally manipulate data objects as internal data structures in the application software, resulting in an unnecessarily costly development process with respect to design, implementation, and verification of the system. As complexity and the amount of data is growing in embedded systems, the need for a uniform, efficient and persistent way to store data becomes increasingly important, i.e., database functionality is needed to provide support for storage and manipulation of data. There are strong reasons why embedding databases into embedded systems have significant gains: reduction of development costs due to the reuse of database systems; improvement of quality in the design of embedded systems since the database provides support for consistent and safe manipulation of data, which makes the task of the programmer simpler. Consequently, this improves the overall reliability of the system. Furthermore, embedded databases provide mechanisms that support porting of data to other embedded systems or large central databases. Existing commercial embedded database systems, e.g., Polyhedra [10], RDM and Velocis [6], Pervasive.SQL [9], and Berkeley DB [11], have different characteristics and are designed with specific applications in mind. They support different data models, e.g., relational vs object-oriented model, and operating system platforms. Moreover, they have different memory requirements and provide different types of interfaces for users to access data in the database. Application developers must carefully choose the embedded database their application requires. However, finding an appropriate embedded database that matches the application requirements can in the worst case be a quite time consuming, costly, and difficult process. What would be a preferable approach is to have more tailorable and flexible system where the application designer can get an optimized database for a specific type of application, where the database provides only necessary functionality, gives a minimum memory footprint, and is highly integrated with embedded real-time systems.

Many embedded systems are also real-time systems that put special demands on the database. These demands have opened a new need for research on real-time databases. Significant amount of research has focussed on various schemes for concurrency control, transaction scheduling, and logging and recovery. Research projects that are building real-time database platforms include DeeDS [1], ARTS-RTDB [15], BeeHive [13], and RODAIN [5]. These emphasize real-time performance. While our project also addresses real-time requirements, the emphasis on how to tailor a database system for application-specific environments in the domain of embedded real-time systems.

Problem description

Traditional database systems are hard to modify or extend with new features mainly because of their monolithic structure and the fact that adding functionality results in increased system complexity. Having database systems that would allow adding or replacing functionality in its architecture in a modular (component-based) manner would be beneficial both for vendors and users. Oracle and Informix introduce limited customization of their database servers [8, 3], by allowing components for managing non-standard data types, data cartridges and DataBlade modules, to be plugged into fully functional database systems. A different approach to componentization is Microsoft's Universal Data Access Architecture [7] that unifies data access and manipulation. In this solution, an architecture is more flexible and customization ability is improved by al-

*This work is supported by ARTES (a network for real-time and graduate education in Sweden) and is done in collaboration with Upright Database Technology AB and Volvo Construction Equipment Components AB. More information about the project can be found at http://www.ida.liu.se/labs/rtslab/projects/ARTES_EmbeddedDatabases/

lowing reuse of existing components (data providers in this case) to build custom solutions on top of them, in addition to developing new components. To the best of our knowledge, the only research project focussing on construction of configurable databases is KIDS [2], which introduces configurable database composed out of components (database subsystems), e.g., object management and transaction management. Customization of this system is improved by having reusable architecture, as well as components stored in the library. Demands on development support in such configurable system are high, and KIDS has to some extent met these demands by providing a well-defined development process, available configuration support, and optional analysis tools, which are missing in other component-based database solutions.

From a real-time point of view, none of the approaches discussed enforce real-time behavior. Issues related to embedded systems such as low-resource consumption are not addressed at all. In contrast, existing component-based embedded real-time systems preserve real-time properties [12, 4, 14]. A component in existing component-based real-time systems is usually mapped to a task, e.g. passive components [12], binary components [4], and port-based object components [14] are all mapped to a task. Therefore, analysis of real-time components in these solutions addresses the problem of managing temporal attributes at a component level by considering them as task attributes [4, 12, 14]: worst case execution time, release time, deadline, precedence constraints, and mutual exclusion. Also, it is pointed out in [12] that components used in embedded systems must have explicit memory needs and power consumption requirements. Most component-based systems, both database and embedded real-time, do not provide adequate configuration and analysis support, which is vital for real-time systems. Thus there is a gap between embedded systems, real-time systems, database systems, and component-based software engineering that needs to be bridged to provide a database that can be tailored for different classes of embedded real-time systems.

Project status and goals

The focus of our research is to provide an embedded database platform that can be tailored for different real-time and embedded applications. We anticipate the following results: (i) a component-based library that holds a set of techniques and protocols for storing and manipulating data, i.e. database functionality, and (ii) a set of tools that can be used when designing and tailoring a database component for embedded real-time systems. Our research should also give a better understanding of the specification of components to be used in real-time setting. This includes functionality provided by the component, the resource demand required by the component when executed on different platforms, and rules for specifying how components can be combined and how the overall system can be correctly verified given that each component has been verified.

References

[1] S. F. Andler, J. Hansson, J. Eriksson, J.Mellin, M. Berndtsson, and B. Eftring. Deeds towards a distributed and active real-time database system. *ACM SIGMOD Record*, Volume 25, 1996.

[2] A. Geppert, S. Scherrer, and K. R. Dittrich. KIDS: Construction of database management systems based on reuse. Technical report, Department of Computer Science, University of Zurich, 1997.

[3] Informix DataBlade technology. Informix Corporation. http://www.informix.com/datablades/, March 2001.

[4] D. Isovic, M. Lindgren, and I. Crnkovic. System development with real-time components. In *Proceedings of ECOOP Workshop - Pervasive Component-Based Systems*, France, June 2000.

[5] J. Lindström, T. Niklander, P. Porkka, and K. Raatikainen. A distributed real-time main-memory database for telecommunication. In *Proceedings of the International Workshop on Databases in Telcommunications*, number 1819 in Lecture Notes in Computer Science. Springer, 1999.

[6] MBrane Inc. RDM Database Manager. http://www.mbrane.com.

[7] Universal data access through OLE DB. OLE DB White Papers. Microsoft Corporation. http://www.microsoft.com/data/oledb/, March 2001.

[8] All your data: The Oracle extensibility architecture. Oracle Technical White Paper. Oracle Corporation, 1999.

[9] Pervasive Inc. Pervasive.SQL Database Manager. http://www.pervasive.com.

[10] Polyhedra Inc. Polyhedra database manager. http://www.polyhedra.com.

[11] S. Software. Berkely db. http://www.sleepycat.com.

[12] J. Stankovic. VEST: A toolset for constructing and analyzing component based operating systems for embedded and real-time systems. Technical report, University of Virginia, 2000.

[13] J. Stankovic, S. H. Son, and J. Liebeherr. BeeHive: Global multimedia database support for dependable, real-time applications. In *Proceedings of the Second Workshop on Active Real-Time Databases (ARTDB-97)*, number 1553 in Lecture Notes in Computer Science. Springer, 1997.

[14] D. S. Stewart. Designing software components for real-time applications. In *Proceedings of Embedded System Conference*, San Jose, CA, September 2000.

[15] Y. K. Kim et al. A database server for distributed real-time systems: Issues and experiences. Technical report, Department of Computer Science, University of Virginia, 1994.