C3L++: Implementing a Description Logics System on Top of an Object-Oriented Database System

T. Kessel and **M. Schlick** and
H.-M. Speiser
Equipe de Recherche en Ingenierie des Connaissances (ERIC), Ecole Nationale Superieure des Arts et Industries de Strasbourg (ENSAIS), 24 Bd de la Victoire, F-67084 Strasbourg CEDEX, France
{kessel—schlick—speiser}@eric.u-strasbg.fr

U. Brinkschulte and **H. Vogelsang**
Institute for Microcomputers and Automation, University of Karlsruhe, Haid-und-Neu-Str. 7, D-76131 Karlsruhe, Germany
{brinks—vogelsang}@ira.uka.de

Abstract. This paper describes the Description Logics system C3L++ that is implemented on top of the object-oriented POET data base. The benefits of such an integration are the management of large size knowledge bases, an increased performance and data persistency. Moreover, it is planned to integrate the relational database MERLIN via an object-oriented access layer in C3L++ and test afterwards its performance.

1 INTRODUCTION

There are different approaches to integrate knowledge representation systems (KRS) and data base management systems (DBMS). Hitherto, most research work was focused on the interfacing of KRS and relational DBMS, e.g. [21], or providing persistent storage systems for KRS, e.g. [15] and [14]. Although these projects rely on a very tight coupling between KRS and DBMS, there are still two different systems that have to be interfaced: the KRS and the DBMS. As a consequence the objective for developing the new KRS C3L++ from the scratch was to integrate a DBMS right from the start such that it is completely transparent for the user who accesses only the KRS front-end.

The knowledge representation system C3L++ is based on the paradigm of Description Logics (DL) [33],[23]. The principal guidelines for building C3L++ are to provide a performant DL system that can be employed in applications requiring large amount of data. This is the case in knowledge extraction from texts and the Plinius project [27] is one example for such applications that served as empirical test base for performance tests of DL systems [26].

The C3L++ KRS will be used in the following applications within the ERIC research lab:

1. A Knowledge extraction from texts supports the knowledge acquisition process by extracting all relevant terms and relations from large text corpora. Current applications focus on the study of medical texts written in french [10], [11].

2. Conceptual Modeling of relational DBMS is based on a terminological formalism to process and optimize queries written in natural language. This approach will be employed for a Data Warehouse application [25].

3. Configuration of a distributed electronic bus system aims at providing an interactive decision support for electronic engineers designing CAN bus application in vehicles [2],[18],[17].

These projects have in common that they require high performance, large size knowledge bases and data persistency. Although the implementation of C3L++ is still in progress, its expressive power will be rather small and therefore oriented at the one of CLASSIC [5],[6]. The most distinguishing feature of C3L++ in comparison to other KRS, i.e. CLASSIC, BACK++ is the full integration of the object-oriented POET DBMS [24]. Roughly speaking, C3L++ tries to combine the benefits of DL-based KRS with those of DBMS by providing on the one hand the front end of DL and on the other hand relying on the DBMS for handling large amount of data and ensuring data persistency. As a matter of fact we hope to obtain very good performance results, although there is still enough room for intelligent caching and data storage strategies improving performance. An alternative for employing the POET DBMS is to use MERLIN that is a relational DBMS designed for high performance in the context of automation. MERLIN provides access to its relational data base schemes via an object-oriented layer hence simplifying the mapping from object-oriented representation formalisms. Although the transformation from object-oriented to relational representation is nowadays standard in industry, the implemented solution in MERLIN promises some benefits.

The remainder of the paper is structured as follows. The C3L++ KRS is briefly described. One major application, the configuration of distributed electronic bus systems is presented motivating high performance and data persistency. Some preliminary results of run time performance tests are discussed and future research perspectives are elaborated. A short introduction to the object oriented layer of the relational DBMS MERLIN is given. Related work concerning the integration of KRS and DBMS is referred to.

2 PRESENTATION OF C3L++

C3L++ is the successor of the C3L research prototype that integrates DL and frames [16]. C3L is founded on a frame-based implementation and proposes in analogy to [22] a distributed concept and attribute specific subsumption [2]. Another particularity of C3L is the integration of procedural attachments, i.e. methods and demons, in form of tasks [18]. This explicit declarative modeling of tasks is oriented at SCARP [32] and LISA [13].

Although the design of C3L++ underwent certain changes due to the necessary adaptation to C++, the C3L core architecture remains preserved. It provides so far only a limited functionality, because its implementation is still in progress. Now, the expressive power of

C3L++ is restricted to AND, ALL, ONE-OF, FILLS and CARD that is a combination of the ATLEAST and ATMOST operators. Further enhancements are planned, but not implemented yet. C3L++ is supposed to serve as one of the major knowledge representation tools within the ERIC lab for

1. knowledge extraction from texts
2. conceptual modeling of relational DBMS for Data Warehouse
3. configuration of distributed electronic bus systems in vehicles

It is widely accepted commons sense in the knowledge representation field that KRS and DBMS should be integrated to combine their advantages. Unfortunately, very few implemented KRS do rely on or are interfaced to DBMS, yet. As a matter of fact, the consequences are a lack of performance for large amount of data and the absence of data persistency. The major benefits of integrating an object-oriented DBMS within C3L++ are

1. the persistency of concepts and instances,
2. the possible large size of knowledge bases and
3. the acceptable performance.

Another advantage of using an object-oriented DBMS is the support for retrieving concepts and instances, hence resulting in an increased performance. Information retrieval is one the crucial tasks of such KRS, therefore any improvement, e.g. with respect to performance, can be significant for its success. Furthermore employing DBMS like POET or the object-oriented variant of MERLIN offers the facility of a very comfortable programming interface that they provide. These are the major reasons why POET and MERLIN are employed respectively studied for use in the C3L++ project.

Beyond measuring the performance of the POET DBMS for C3L++ we would like to compare the performance of both DBMS POET and MERLIN with respect to their use in C3L++. The results might guide other system designers whether to select a pure object-oriented DBMS, e.g. POET, or to prefer a relational DBMS, e.g. MERLIN, that has an object-oriented access layer.

C3L++ principal properties are as follows:

- it is built on top of the object-oriented POET DBMS
- it provides only a restricted expressive power
- it is designed for high performance
- it is implemented in C++ and runs under Windows 3.1 and UNIX

POET is an object-oriented DBMS from POET Software Corporation. C++ classes are marked as persistent and in consequence all their instances are handled as persistent objects. Note, that all classes that are inherited by persistent classes are automatically persistent as well. POET is in fact a pre-compiler that generates standard ANSI C++ code. Therefore its output source code can be compiled by any other C++ compiler. The advantage of using POET is that only small modifications have to be made at the source code, i.e. marking of classes as persistent. The shortcoming is that not all constructs and features of C++, e.g. templates, are supported by the POET pre-compiler. Hence it can be considered as a supplementary DBMS programming tool for a C++ development environment.

3 CONFIGURATION AS AN APPLICATION

In modern vehicles electronic sensors, actors and control units are connected via bus systems. The upcoming standardisation for these buses is the CAN (controller area network) [12]. The configuration of these buses has become rather complex. The engineer has to specify the functionality of the system, select the appropriate components and position them in the vehicle. He has to connect them via buses and specify the messages. Additionally he has to guarantee a maximum of safety of the system. Finally he has to find the cheapest solution for the configuration problem.

It is necessary to support the engineer with an interactive configurator. To build this configurator we use DL. That DL is useful to develop a configurator has already been shown by the AT&T research group in the framework of the PROSE project [31], but still it is not clear how to deal with large amounts of data.

An essential problem for the configuration of the vehicle buses is that large amounts of data have to be handled. E.g. about 300 objects are necessary to model only the different previewed airbag functionalities their components and their messages. It is obvious that this has to be stored in a database.

A sufficient access to the data is important for the configuration process. E.g. during the configuration the engineer has to find the appropriate compartment for a component. It is necessary to check whether the properties offered by a component match the demands of the compartments [2]. Typical properties are for example temperature, water resistance or electromagnetic compatibility conditions. Two different approaches could be used to solve this problem. The first is that the engineer proposes a compartment and the system checks whether the properties match or not. If not the system refuses the proposal. This would lead to a lot of refused configuration decisions, hence the engineer would not accept at all the configurator. The second possibility is to compute all possible matches between compartments and components and to ask the engineer to choose one of these. This is only possible if the computation is not too expensive otherwise the configurator will also be refused by the engineer. For the estimation of the computation costs it is necessary to know more about the performance of the system. Especially the combination between DL and an object-oriented DBMS is a promising approach, but also the combination with a relational DBMS is important as well because this would enable us to use already existing DB, i.e. product databases in companies.

4 PERFORMANCE TESTS

Although the implementation of C3L++ is not finished yet, however some preliminary performance tests allowed to validate the current state of its TBox. The tests consist essentially of generating thousands of concepts that are independent, that means they are not linked to each other. Note that the objects constituting concepts are nested. The very first performance tests measured only the efforts to create and to store concepts in the main memory respectively in the database. Afterwards, the dependencies between data loading ("caching") strategies and inferences, e.g. subsumption, classification, are studied. Until now, these tests were only done by means of POET, but we hope to integrate as soon as possible the relational MERLIN DBMS via its object-oriented layer. Detailed results of these tests can be found in [19]. By the way, the measurements were run on a Pentium PC with 100 MHz and 32 MB main memory. In the following, some results of the above mentioned tests are listed.

No. of concepts	Main memory	POET DBMS
10	0,01	1,04
100	0,16	11,48
1000	1,76	112,48
10000	27,47	1289,89

Employing an object-oriented DBMS engenders a considerable overhead — in comparison to the main memory — for the creation of concepts. On the one hand, the concept generation is essentially slowed down, for instance creating 10.000 concepts takes only 27 seconds in the main memory in comparison to 1289 seconds in the DBMS. Surprisingly it does not make a difference whether concepts are stored separately or in clusters. This fact seems to indicate that performance gains are possible thanks to better caching strategies.

Obviously, the hard disk access time limits severely the performance that means storing an object in the main memory is approximately hundred times faster than storing it as a persistent object. On the other hand, it was impossible to create more than 15.000 concepts in the main memory due to a systematic breakdown of the memory system. Although it remains still feasible for the object-oriented DBMS version. Furthermore, the performance for creating many thousands concepts directly in the main memory increased in a non-linear way, whereas the DBMS performance evolved linearly. We are curious to compare these results with the ones made by MERLIN in a similar test.

However the gain of security thanks to data persistence cannot be expressed in quantitative terms, but hit as to be taken into account as well. Note that the effects observed so far are only valid for large knowledge bases consisting of many thousands of concepts.

A more sophisticated knowledge base test generator is currently in study. It promises to engender nested concepts with varying degrees of complexity. Moreover, knowledge bases will be reasoned about by means of different inferences, e.g. subsumption, classification, in analogy to the tests done by [1] and [26]. But this time the topic is focus is performance behaviour for large knowledge bases thanks to the integrated data base. The work will be oriented at the approach of Karp in [15] and [14]. However the work is still at its early stages. Nevertheless we are very interested in the real performance of usual DL inferences on average knowledge bases in order to estimate the practical value of implemented DL systems, i.e. in the context of knowledge acquisition and of configuration systems [18].

5 OBJECT-ORIENTED DATABASE INTERFACE

This interface for the existing relational database ”MERLIN” [7],[8],[9] brings together the needs for object persistence and high speed database access. It was created as a component in the construction of system using the ECBS-process [28],[30]. Later it has been found out that this is an ideal solution for C3L++ too. The interface offers persistence as a property of an object through which its existence transcends time [3]. This is realised using one or more database servers simultaneously, even on distributed hardware platforms.

Persistent objects are realised by declaring the corresponding classes as persistent. This is done by deriving these classes from the internal class ”Persistent” on the applications side. On the systems internal side every persistent object is connected with one (of several possible) database objects. These acts as stubs for the real local or remote database server in a heterogeneous net of computers. The connection to a stub is dynamically changeable so that an object’s content can be loaded from or stored to different databases. The advantage of this design is that a persistent object is placeable anywhere on a net. No object has to know its own storage place, only the reference to the local stub is kept.

An other requirement of C3L++ is the persistence of relations between objects, which are expressed in object oriented programming languages by pointers. To keep the relations between persistent objects after a program termination alive the smart pointer class ”PersistentPtr” is introduced. In addition to the memory mapped reference to the destination object it contains the objects unique identification. This class either allows the automatic reloading of the destination object during its own creation or a program controlled recreation. This two mechanisms enable either a total rebuild of the entire data structure only by reloading the base object or a partial loading of huge structures at a given point of time.

6 RELATED WORK AND REMARKS

On the contrary to many previous attempts to interface KRS with DBMS by means of a loose coupling, e.g. [4], [20] we do not suppose a loose coupling, but a transparent and tight integration. Contrasting with the work of interfacing KRS and relational DBMS, e.g. [21], or providing persistent storage systems for KRS, e.g. [15], [14], the DBMS is integrated by means of a pre-compiler, hence resulting in a full, closed integration. Furthermore we are interested to compare the performances of the object-oriented DBMS POET and the relational DBMS MERLIN with respect to the same KRS: C3L++.

Therefore the DBMS is completely hidden within the KRS. Furthermore, the object-oriented KRS implementation structures are directly mapped to adequate data structures provided by the object-oriented DBMS. Therefore, complicated transformation operations between object-oriented and relational representations are avoided.

The following remarks are some kind of disclaimers that should help to better understand the research work. First, combining KRS, i.e. DL systems, and DBMS rises always the question how both semantics, namely the open world semantics of DL and the closed world semantics of DBMS are conciliated. The answer is rather simple, because the DL frond end provides the open world semantics for the user, whereas the underlying DBMS is only employed for storing and loading persistent objects. Retrieval results of the DBMS are especially processed to respect the open world semantics of DL.

Second, the current implementation state of C3L++ does no optimise the mapping from he DL front end to the underlying object-oriented POET DBMS, although this is foreseen in the future. The transition from the formalism of DL to DBMS is completely transparent for the user, hence the user does only see the DL front end.

Third, the overall objective of the C3L++ project is to build a performant DL system by fully integrating an object-oriented DBMS. Therefore the focus is on performance and reliability for large knowledge bases, although these goals risk to be perceived as engineering instead of research issues, they seem to be indispensable to us in order to satisfy the requirements of our applications.

ACKNOWLEDGMENTS. The described joint research work was done within the framework of the french-german Institute for Automation and Robotics (I.A.R.). We would like to thank in particular Prof. Schweitzer and Prof. Keith for establishing this fruitful cooperation between both institutes.

References

[1] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, et al. *An Empirical Analysis of Optimisation Techniques for Terminological Representation Systems.* Principles of Knowledge Representation and Reasoning (KR-92) (San Mateo, CA, 1992) pp.270-281.

[2] T. Berger, T. Kessel, F. Rousselot, M. Schlick. *C3L: A System Integrating Description Logics and Frames.* (ERIC, 1996), Technical Report.

[3] G. Booch. *Object oriented design with applications.* The Benjamin-Cummings Publishing Company, 1991.

[4] A. Borgida, R.J. Brachman. *Loading Data in Description Reasoners.* ACM SIGMOD Intern. Conference on Management of Data (Washington, D.C., USA, 1993) pp.217-226.

[5] R.J. Brachman. *'Reducing' CLASSIC to Practice: Knowledge Representation Theory Meets Reality.* Principles of Knowledge Representation and Reasoning (KR-92) (Cambridge, Mass., 1992) pp.247-258.

[6] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, et al. *Living with Classic : When and how to use a KL-ONE-Like Language.* In: Principles of Semantic Networks (Morgan Kaufmann, San Mateo, California, 1991) pp.401-456.

[7] U. Brinkschulte. *MERLIN — Ein Prozeßdatenhaltungssystem für Echtzeitanwendungen.* Echtzeit 93, Karlsruhe, Germany, 1993.

[8] U. Brinkschulte. *Architektur eines Datenhaltungsdienstes.* 39. Wissenschaftliches Kolloqium TU Illmenau, Illmenau, Germany, September 1994.

[9] U. Brinkschulte. *Database Services.* Knowledge Engineering and Object-Oriented Automation Workshop, Strasbourg, France, May 1995.

[10] P. Frath, R.Oueslati, F. Rousselot, T.Barthelemy. *Sémantique et traitement automatique des langues: Réflexion et application pratique.* TALN'96, Troisiéme conférence annuelle sur Le Traitement Automatique du Langage Naturel, Marseille, 22 - 24 Mai 1996.

[11] P. Frath, R. Oueslati, F. Rousselot. *Identification de relations sémantiques par repérage et analyse de cooccurences de signes linguistiques.* JAVA 95, Grenoble, avril 95, 1995.

[12] ISO 11898 and ISO 11519-2

[13] I. Jacob-Delouis, J.-P. Krivine. *LISA: un langage réflexif pour opérationnaliser les modules d'expertise.* Revue d'intelligence artificielle, Vol.9, ,1995 pp.53-88.

[14] P. Karp, S.M. Paley, I. Greenberg. *A storage system for scalable knowledge representation.* Conference on Information and Knowledge Management (CIKM-94) (1994).

[15] P.D. Karp, S.M. Paley. *Knowledge Representation in the Large.* International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, 1995, pp.751-758.

[16] T. Kessel, O. Stern, F. Rousselot. *From Frames to Concepts : Building a Concept Language within a Frame-Based System.* Int. Workshop on Description Logics (DL-95), Rome, Italy, 1995, pp.140-142.

[17] T. Kessel, M. Schlick, O. Stern. *Accessing Configuration-Databases by means of Description Logics.* KRDB-95, Bielefeld, Germany, 1995.

[18] T. Kessel, M. Schlick, O. Stern. *Modelling for Configuration employing hybrid Knowledge Representation.* Engineering of Computer-based Systems (ECBS-96), Friedrichshafen, 1996, pp.118-124.

[19] T. Kessel, H.-M. Speiser. *Run Time Performance Measures of C3L++.* ERIC, 1996, Technical Report.

[20] C. Kindermann, P. Randi. *Object Recognition and Retrieval in the BACK system.* International Working Conference on Cooperating Knowledge Based Systems (CKBS'90), 1991, pp.311-325.

[21] E. Mays, S. Lanka, B. Dionne, R. Weida. *A persistent store for large shared knowledge bases.* IEEE Transactions on Knowledge and Data Engineering Vol.3, no. 1, 1991, pp.33-41.

[22] A. Napoli, C. Laurenco, R. Ducournau. *An Object-Based Representation System for Organic Synthesis Planning.* International Journal of Human-Computer Studies Vol.41, no. 1/2, 1994, pp.5-32.

[23] B. Nebel. *Reasoning and Revisioning in Hybrid Representation Systems.* Springer, 1990.

[24] POET. *POET 3.0 Reference Guide.* POET Software Corporation, 1995, Technical Report.

[25] D. Rudloff. *Terminological Reasoning and Conceptual Modeling for DataWarheouse.* KRDB-96, Budapest.

[26] P.-H. Speel. *Selecting Knowledge Representation Systems.* University of Twente, Enschede, the Netherlands, 1995. Thesis.

[27] P.E.v.d. Vet, H.d. Jong, N.J.J. Mars, P.-H. Speel, et al. *Plinius Intermediate Report.* Department of Computer Science, University of Twente, Enschede, the Netherlands, 1994, Technical Report UT-KBS-94-10.

[28] H. Vogelsang, U. Brinkschulte, M. Siormanolakis. *Archiving System States by Persistent Objects.* Engineering of Computer-based Systems (ECBS-96), Friedrichshafen, 1996

[29] Holger Vogelsang, Uwe Brinkschulte. *Persistent Objects in a relational Database.* In: Proceedings of WOON'96, St. Petersburg, Russia, 1996

[30] M. Voss. *System Theories for an Engineering Discipline of Computer-Based Systems.* EMCSR96/Session C

[31] J.Wright, E. Weixelbaum, et al. *A knowledge-based configurator that supports sales, engineering and manufacturing at AT&T Network Systems".* Proceedings of the Innovative Applications of Artificial Intelligence Conference, 1993

[32] J. Williamowski, F. Chevenet, F. Jean-Marie. *A development shell for cooperative problem-solving environments.* Mathematics and computers in simulation Vol.36, no. 4-6, 1994, pp.361-379.

[33] W. Woods, J. Schmolze. *The KL-ONE family.* In: Semantic Networks in Artificial Intelligence, Pergamon Press, 1992, pp.133-177.