DÍGAME: A Vision of an Active Multidatabase with Push–based Schema and Data Propagation

Cristian Pérez de Laborda, Christopher Popfinger, and Stefan Conrad

Institute of Computer Science
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Germany
{perezdel, popfinger, conrad}@cs.uni-duesseldorf.de

Abstract. Sharing information in loosely coupled enterprises or virtual corporations demands a flexible and dynamic architecture, suitable for their individual data policies. The aim of this paper is to present the DÍGAME architecture, which balances both, local autonomy and a reasonable degree of information sharing. Therefore we combine the well known concept of loosely coupled multidatabases with more recent research in Peer–to–Peer or grid computing, satisfying the needs of modern intra- and inter-enterprise collaboration. In our architecture data and schema updates are propagated actively to subscribing component databases, without being managed by any central authority. This replication gives us the possibility to realize individual integration on each single peer.

1 Motivation

Since the first centralized databases found their way into the enterprises in the late 60s, the needs and requirements have changed towards a more distributed management of data. Today there are many corporations which possess a large amount of databases, often spread over different regions or countries and generally connected to a network. These local databases (components or component systems) typically raised in an autonomous and independent manner, fitting the special needs of the users at the local site. The design of the databases and the functionalities provided, intend to fulfil the aims of the departments. This leads to logical and physical differences in the databases concerning data formats, concurrency control, the data manipulation language or the data model [1]. An information system is required to integrate the information of these heterogeneous data sources to provide a global access.

One of the main challenges in the integration of data in such environments, is the autonomy of the participating data nodes (peers). This autonomy implies the ability to choose its own database design and operational behaviour [2]. Local autonomy is tightly attached to the data ownership, i.e. who is responsible for the correctness, availability and consistency of the shared data. Centralizing data means, to limit local autonomy and revoke the responsibility from the local administrator, which is not reasonable in many cases. The federated architecture

for decentralizing data has to balance both, the highest possible local autonomy and a reasonable degree of information sharing [3].

In this paper we introduce the vision of the DÍGAME architecture, a **D**ynamic **I**nformation **G**rid in an **A**ctive **M**ultidatabase **E**nvironment, which actively propagates data and schema updates over import/export-components between dynamically connectable data peers. This architecture offers a flexible and fail–safe information platform based on the data policies in organisations achieving a feasible trade–off between local autonomy and a reasonable degree of information sharing.

2 DÍGAME Architecture

2.1 Vision

The aim of this work is to introduce an architecture which allows the dynamic connection of data sources without restricting their local autonomy in order to share selected information. This union is based on Peer-to-Peer (P2P) concepts and operates without any central administrative instance.

The administrator of each peer makes a subset of its data accessible. Other peers are now able to integrate this data into their local databases, subscribing to a specific part of the data provided. Thereupon updates are propagated automatically to the subscribers by the data source, including both, data and schema modifications. Each data source of this dynamic information grid is herewith able to maintain an up-to-date replica of the required data and schema items. As there is no general rule for the integration of the replicated data, it has to be integrated individually by the administrator of each subscriber database.

Our architecture is especially designed to support dynamic intra– and inter–enterprise collaboration, by enabling each department involved to supply all relevant partners with the required information.

2.2 Components

We will now discuss the required components of the architecture using a case study, to draw up the benefits of our dynamic information grid. This example describes our approach to solve one of the multiple challenges concerning collaborative work: distributed information management.

Consider a worldwide operating company, planning the launch of a new product. To simplify our scenario, we assume that there are solely three departments involved in this business process, the executive board (management), the sales office and product engineering. Further departments may join this collaboration at any time. Each department manages its own database, to store the information for which it is responsible. The management produces basic data of the product. This includes deadlines, descriptions, workflows and additional objectives. This management information is substantial for the further product development and the work in the participating departments. The product engineering

uses a predefined part of that management data as basic conditions for the concrete implementation and technical realization. Local applications like CAD or measurement programs create additional data which has to be stored separately. According to the product engineering the sales department enriches the authoritative management data with concrete concepts for the oncoming product launch. Furthermore concrete development plans of the product engineering are required to prepare sales strategies. Both, sales and product engineering departments, concretize the strategic guidelines of the management in their specific assignment. To keep track of the costs and the progress of the project, it is indispensable for the management to access the product engineering and sales departement's relevant information just mentioned.

Basically there are two different techniques for providing the peers (departments) with the required data. Contrary to the commonly used method querying the data sources actively, our approach uses replication of data and schema, initiated by the data source. Referring to our example, the executive board gets data updates whenever changes occur in the sales and/or product engineering databases rather than having to request for updated data items continuously.

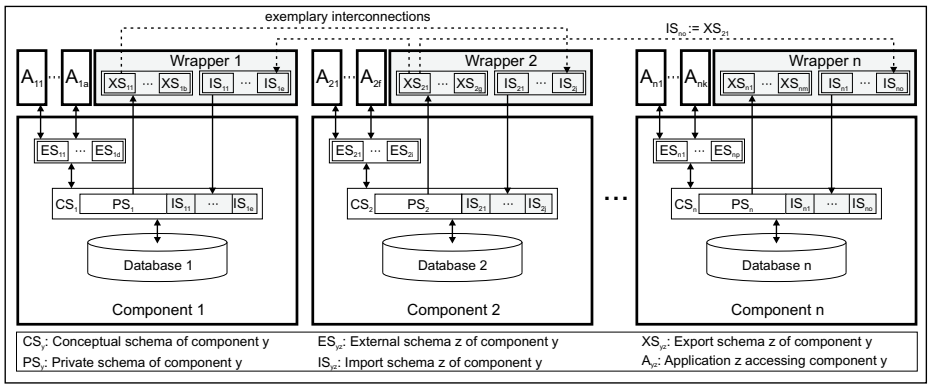

Fig. 1. DÍGAME Architecture

Our DÍGAME architecture (Figure 1) consists of the following components:

Autonomous Component Databases: As already mentioned our architecture is designed for integrating data from across autonomous data sources. The peers involved are linked to an information grid, retaining their local autonomy completely. According to the 3-level-architecture [4] and the architecture for loosely coupled multidatabases [3], each component database consists, besides the internal schema, of a conceptual and several external schemas. The conceptual schema (CS) comprises the locally maintained private schema (PS) and a couple of schemas imported from other peers (IS), managed by a wrapper component. Local applications (A) access the data

via external schemas (ES), which are derived from the whole conceptual schema, providing solely read-only access to all imported schemas. The grid infrastructure does not include a global view over the integrated data, but every peer maintains its own integrated schema. Due to the absence of a global view, we have ideal conditions for individual integrations on each peer.

Wrapper: The core of our dynamic information grid is the wrapper component. As part of the middleware, it is responsible for negotiating and establishing communication and exchanging data between the component databases. Therefore it maintains a *repository* containing all the meta data accumulated, particularly a copy of the import schemas mentioned above and export schemas (XS) based exclusively on the private schema. Of course corresponding schema information has to be stored only if data is imported or exported. Thus the participation level corresponds directly to the amount of schema information the wrapper has to manage. In fact each import schema matches an export schema, offered by one of the remaining peers.
In addition to the repository, two more modules have to be implemented inside the wrapper. A *publishing unit* is used for transmitting information. Earlier research proposes several mechanisms helping a wrapper to identify data modifications [5, 6]. If there are triggers of underlaying database systems available, they should be used. The counterpart of the publisher is a *subscribing unit*, which receives incoming information. Both units contain a *negotiator*, which sets up a communication channel, used by a *data handler*, to exchange data in a standardized format (e.g. RDF).

2.3 Characteristics

Our architecture combines the advantages of established concepts known not only in the database field, but also in related research areas, like grid or P2P computing. The combination of the established *Three Schema Architecture* [4] and that of loosely coupled multidatabases [3] with the achievements of the more recent field of P2P data management [7] provides a promising framework for an enterprise information platform. We are thus able to apply the flexible interconnectivity of P2P systems to multidatabases, including not only relational databases, but a loosely coupled federation of virtually any kind of data source. Although we focus in this paper on relational database systems, our architecture can be adapted to X.500 directory services or even file systems by adjusting the wrapper component.

In fact, the data replication on each subscriber database provides a couple of advantages, according to distributed databases systems [8]. As the peers serve all their local applications with the data required, there is no need for these to query remote data sources, leading to an increase of performance. Furthermore, a temporary network blackout can be bridged without being noticed by the applications. Comparable to distributed databases, in scenarios with more data queried than modified, even a significant reduction of network traffic can be

obtained. Of course a replication entails the redundant storage of data items, but this disadvantage is neglectable due to the rapid decrease of storage costs.

To ensure a high level of data quality, the data can only be modified by the data owner. Information sharing between autonomous data sources is realized without loss of data ownership and autonomy on each peer, leading to a higher quality of data [9].

To guarantee both, the correctness and up-to-dateness of the data, each single modification can be propagated by pushing it to the subscriber databases. Hence each peer is able to provide, a running network environment supposed, up-to-date data to its applications at any time. Due to the push-based characteristics of DÍGAME updates may be lost if a communication failure caused by a network or computer breakdown occurs. In this case there are basically two possibilities to re-synchronize the data. Either the publishing peer repeats the lost update propagations or the subscribing peer itself demands for these data and schema modifications. Since the first option enforces the data source to keep a complete track on the success of every propagation to each subscribers, we prefer to integrate a pull-based fallback mechanism into our architecture. This means that the subscribing peer has to search for lost data and schema updates by itself. The concrete definition of this functionality is part of future work.

3 Related Work

Simultaneously to the first generation of grid computing in the mid 1990s [10] some efforts arised to use distributed resources for information retrieval. Although the *Information Grid* of Rao et al. [11] is focused on giving an integrative user interface for distributed information, this approach can be seen as an early forerunner of the so called *Data Grid* [12], a specialization and extension of grid computing. Its intention is to create an architecture of integrated heterogeneous technologies in a coordinated fashion. Though we admit that a global metadata repository as proposed by Chervenak et al. would simplify many of the challenges, we abstain from that that effort of re-centralization, as it brings many difficulties about: every schema change has to be replicated to the global schema directory. The effect is a single point of failure, exactly the opposite of what we wanted to construct. We thus prefer to keep the databases as they are: autonomous, loosely coupled, and without a single point of failure.

With the raise of filesharing systems like Napster or Gnutella [13] the database community started to seriously adopt the idea of P2P systems to the formerly known loosely coupled database systems. Contrary to the data grid, P2P database systems do not have a global control in form of a global registry, global services, or a global resource management, but multiple databases with overlapping and inconsistent data. These P2P databases resemble heterogeneous and distributed databases, also known as *multidatabases* [14]. Currently the database community makes a great effort in investigating P2P databases. Particularly the *Piazza* [7] project is worth mentioning, where a P2P system is built up with the techniques of the *Semantic Web* [15] with local point–to–point

data translations, rather than mapping to common mediated schemas or ontologies. Contrary to Halevy et al., we deal mainly with relational data and do not have a global schema, as every peer may have its own import–/export–schema combination. For a more general glimpse on data mappings in P2P systems see [16].

Our strategy allows data to be exchanged among distributed databases connected through a lazy network. This means, that although a running network may not be guaranteed and thus some data broadcasts may be lost, the system heals itself. This challenge resembles the problems known from environments with mobile databases. Current research covers synchronous mobile client synchronization, i.e. data changes are propagated periodically and not just in time of the data change. In contrast to the broadcast disks, we ensure in our model, that data is only broadcasted to the clients when changes occur, unless the communication between both peers crashes. Hence our approach resembles a *push–based* system with a *pull-based* fallback, similar to [17] with the major difference that our approach is not based on broadcast disks, but on a push–based replication strategy also found in mobile clients like [18], resembling the software engineering's *Observer–Pattern* [19]. This pattern gives us a prototype of how to notify all interested databases about data updates [18]. As a result, communication is only started, if a data update has occurred and a database is interested. In consequence, data broadcasts are minimized.

Following the argumentation in [20] and [12] our model provides Single–Master Replication, the only guarantor for data stability and clear defined data flows.

Most of the research on active multidatabases has been done concerning global integrity. Chawathe et al. [21] propose a a toolkit for constraint management in loosely coupled systems. To mention is also the idea of Gupta and Widom to optimize the testing of global constraints by local verification [22]. Conrad and Türker [6] sketch a more general architecture for an active federated database system. They extend a multidatabase system by ECA-Rules to preserve consistency. A main challenge hereby is to detect local events, especially schema and data modifications, which is commonly done by a software module for each data source, i.e. a monitor or wrapper component. Basically two approaches are therefore proposed: Conrad and Türker use the event detection ability of the underlying subsystem, while Blanco et al. [5] use the operating system to signal schema modifications by directly observing changes to the data(base) files.

4 Conclusion and Future Work

We have presented in this paper an architecture for a dynamic information grid in an active multidatabase environment, suitable for sharing information across autonomous and heterogeneous data sources. Our loosely coupled federation enriched with P2P and grid computing concepts, enables collaborative work preserving local autonomy. Data and schema modifications are actively propagated

to the clients after they have subscribed to the information offered by the data source.

For a complete implementation of the DÍGAME architecture, there are still some challenges to be taken. In the next step of the project we will focus on the detailed specification of the wrapper component, particularly on the negotiator and the data handler, for being able to establish communication between isolated peers. Therefore we have to specify a communication protocol and a data and schema exchange format.

Due to its characteristics DÍGAME provides a sophisticated infrastructure for a diversified application field, including e-business, e-science or e-health, initiating the next generation of collaborative work.

References

1. Litwin, W., Abdellatif, A.: Multidatabase Interoperability. Computer **19** (1986) 10–18
2. Mullen, J.G., Elmagarmid, A.K., Kim, W., Sharif-Askary, J.: On the Impossibility of Atomic Commitment in Multidatabase Systems. In: Proceedings of the 2nd International Conference on System Integration, Morristown, New Jersey, IEEE Computer Society Press (1992) 625–634
3. Heimbigner, D., McLeod, D.: A Federated Architecture for Information Management. ACM Transactions on Information Systems (TOIS) **3** (1985) 253–278
4. Burns, T., Fong, E.N., Jefferson, D., Knox, R., Mark, L., Reedy, C., Reich, L., Roussopoulos, N., Truszkowski, W.: Reference model for dbms standardization, database architecture framework task group (daftg) of the ansi/x3/sparc database system study group. SIGMOD Record **15** (1986) 19–58
5. Blanco, J.M., Illarramendi, A., Pérez, J.M., Goñi, A.: Making a Federated Database System Active. In: Database and Expert Systems Applications, A.M. Tjoa and I. Ramos (eds.), Springer Verlag, ISBN 3-211-82400-6. (1992) 345–351
6. Türker, C., Conrad, S.: Towards Maintaining Integrity of Federated Databases. In: Data Management Systems, Proc. of the 3rd Int. Workshop on Information Technology, BIWIT'97, July 2–4, 1997, Biarritz, France, Los Alamitos, CA, IEEE Computer Society Press (1997) 93–100
7. Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: Data Management Infrastructure for Semantic Web Applications. In: Proceedings of the twelfth international conference on World Wide Web, Budapest, Hungary (2003) 556–567
8. Ceri, S., Pelagatti, G.: Distributed databases principles and systems. McGraw-Hill, Inc. (1984)
9. van Alstyne, M., Brynjolfsson, E., Madnick, S.: Why not one big database?: principles for data ownership. Decision Support Systems **15** (1995) 267–284
10. Roure, D.D., Baker, M.A., Jennings, N.R., Shadbolt, N.R.: The Evolution of the Grid. In Berman, F., Fox, G., Hey, T., eds.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc., New York (2003) 65–100
11. Rao, R., Card, S.K., Jellinek, H.D., Mackinlay, J.D., Robertson, G.G.: The Information Grid: A Framework for Information Retrieval and Retrieval-Centered Applications. In: Proc. of the 5th Annual Symposium on User Interface Software and Technology (UIST'92), Monterey, CA (1992) 23–32

12. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S.: The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications **23** (2000) 187–200
13. Carlsson, B., Gustavsson, R.: The Rise and Fall of Napster - An Evolutionary Approach. In: AMT 2001, Proceedings of the 6th International Computer Science Conference - Active Media Technology. Volume 2252 of Lecture Notes in Computer Science., Hong Kong, China, Springer (2001) 347–354
14. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu:, I.: Data management for peer-to-peer computing: A vision. In: Proc. of the Fifth International Workshop on the Web and Databases, WebDB 2002, Madison, WI (2002)
15. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
16. Kementsietsidis, A., Arenas, M., Miller, R.J.: Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, San Diego, CA, ACM Press (2003) 325–336
17. Acharya, S., Franklin, M., Zdonik, S.: Balancing push and pull for data broadcast. In: Proceedings of the 1997 ACM SIGMOD international conference on Management of data, Tucson, Arizona, ACM Press (1997) 183–194
18. Hara, T.: Cooperative caching by mobile clients in push-based information systems. In: Proceedings of the eleventh international conference on Information and knowledge management, McLean, Virginia, USA (2002) 186–193
19. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements od Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, New York, NY (1995)
20. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The Dangers of Replication and a Solution. In: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Montreal, Canada, ACM Press (1996) 173–182
21. Chawathe, S., Garcia-Molina, H., Widom, J.: A Toolkit For Constraint Management In Heterogeneous Information Systems. In: Proceedings of the International Conference on Data Engineering, New Orleans, Louisiana (1996) 56–65
22. Gupta, A., Widom, J.: Local Verification of Global Integrity Constraints in Distributed Databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data SIGMOD'93, Washington, DC (1993) 49–58