# A Flexible Architecture for a Push-based P2P Database

Cristian Pérez de Laborda and Christopher Popfinger

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

**Abstract**

Originally developed for file sharing, P2P concepts are predestinated to realize information sharing in a more general way. Combined with the well known concept of loosely coupled multidatabases they provide a promising fundament for sharing data and metadata across autonomous and heterogenous data sources. The DÍGAME architecture allows each participating data source to decide autonomously, which kind of information to share. Data and schema updates on these shares are propagated actively to subscribing peers, without having to be managed by a central authority. Each peer is now able to maintain a replica of all the data demanded locally, regardless of where the data is stored originally. In this context we have identified several link patterns, indispensable for understanding and designing an information grid.

## 1 Motivation

Accessing data across various autonomous and heterogenous data sources is still a challenge to be taken. Especially global corporations possess a large amount of databases, often spread over different regions or countries. These local databases 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 [9]. An information system is required to integrate the information of these heterogeneous data sources to provide a global access.

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.

In section 2 we start describing the basic functionality of our architecture, followed by a discussion of each component. The most relevant patterns for linking the collaborating peers are given in section 3. Section 4 discusses related work and section 5 concludes and draws up future work.

## 2 DÍGAME Architecture

### 2.1 Concept

In this section we introduce the basic functionality of the DÍGAME 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. Of course data may also be actively requested by the subscribers, as done while subscribing to a new data share or after a transmission failure, e.g. a network breakdown. 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 university planning to manage data concerning their students more efficiently (Figure 1). To simplify our scenario we assume that there are only three faculties or departments involved: the central administration (A) of that university, the department of biology (B), responsible for the major subject and the department of chemistry (C), responsible for the minor subject. Further departments or divisions may join this collaboration at any time. Each department manages its own student database, containing all the relevant data for its everyday business.

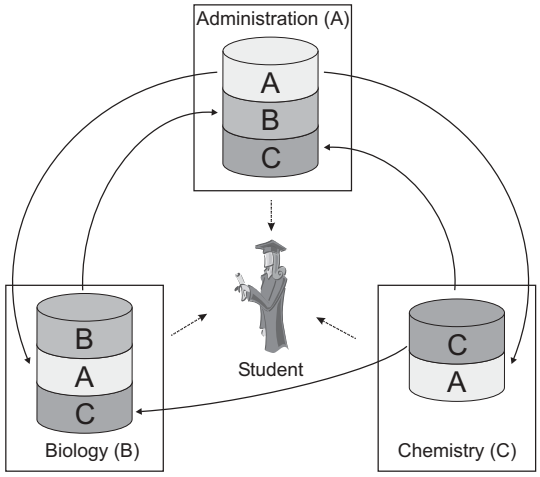


Figure 1: Data Management realized with DÍGAME

The central administration collects matriculation data of the students, including their names, addresses, and field of study. This basic information is substantial for both faculties, that of the major and minor studies, as well as for additional departments managing grants, internships, or sporting activities. The department of biology uses a predefined part of the matriculation data as basis for its own data stock. Local applications create additional data like exam results and absent times, stored separately. Accordingly, the department of chemistry enriches the matriculation data with information concerning the minor studies. This data is also required by the department of biology, since it is responsible to monitor the process of the studies and the compliance of obligations. Both, biology and chemistry departments use the administrative information for their specific assignment. In return, the central administration requires data for being able to charge the correct amount of tuition fees or remove the students from the registers.

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 biology department gets data updates whenever changes occur in the administration and chemistry databases rather than having to request for updated data items continuously and vice versa. Our DÍGAME architecture (Figure 2) consists of the following components:

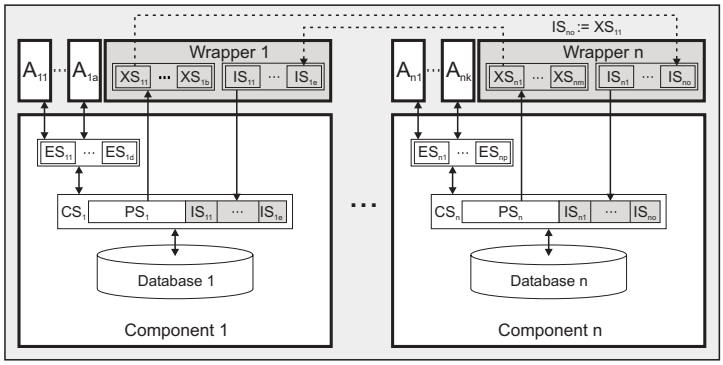


Figure 2: DÍGAME Architecture

**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 [3] and the architecture for loosely coupled multidatabases [8], 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 architecture is the wrapper component. As part of the middleware, it is responsible for negotiating and establishing communication and exchanging data between the component databases. For this it has to detect modifications on the local data and schema. If there are triggers of underlaying database systems available, they should be used, particularly their extended functionality given by recent developments in database systems [11]. All the meta data accumulated is stored in a *repository*, particularly a copy of the import schemas mentioned above and export schemas (XS) based exclusively on the private schema. In fact each import schema matches an export schema, offered by one of the remaining peers.

## 2.3 Characteristics

The combination of loosely coupled multidatabases with the achievements of the more recent field of P2P data management 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. 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. 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. 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 we integrate a pull-based fallback mechanism into our architecture.

# 3 DÍGAME Link Patterns

Any information grid based on our DÍGAME architecture, can either be planned or evolve dynamically. In both strategies the following link patterns between the participating peers can be identified (Fig. 3):

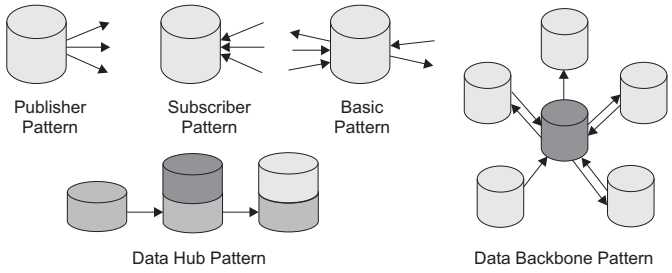


Figure 3: DÍGAME link patterns

**Publisher Pattern:** A peer, which exclusively provides data without receiving any data, implements the Publisher Pattern.

**Subscriber Pattern:** A peer, which exclusively receives data without publishing any data, implements the Subscriber Pattern. Of course a peer can only subscribe to data, published by a remote one.

**Basic Pattern:** Usually a peer will implement this combination of both elementary link patterns. It receives data from remote data sources, while it publishes data to subscriber databases. Certainly the amount of its incoming data flows is independent from its outgoing data flows.

**Data Hub Pattern:** This pattern is a specialisation of the basic pattern. As soon as a peer forwards an exact copy of the data received from another peer, it is called a data hub. If there is no possibility or no intention to establish a direct link between two peers, the data hub provides a way to receive data from an indirectly connected source. Is the data propagated by more than one data hub, we have to prevent a peer to get identical information items from different sources, without noticing it. Therefore we have introduced a global identification mechanism for information items [10]. Please note, that the data hub pattern can only be applied, if our DÍGAME architecture (Fig. 2) is extended to allow the data exported to be based on the complete conceptual schema, i.e. particularly the data imported from other peers. Thus the data owner may lose control over its own data.

**Data Backbone Pattern:** A special form of a data hub is the data backbone pattern. A peer, which corresponds to this pattern, distributes data starlike from multiple sources. Thus it is a crucial node inside the information grid and represents always a weakness of a peer-to-peer environment. This single point of failure should be avoided as far as possible.

Of course, this list of link patterns is not exhaustive, since there is a multitude of peer constellations imaginable. We have only presented the most important building blocks for our DÍGAME information grid.

# 4 Related Work

With the raise of filesharing systems like Napster or Gnutella [4] the database community started to seriously adopt the idea of P2P systems to the formerly known loosely coupled database systems. Particularly the *Piazza* [7] project is worth mentioning, where a P2P system is built up with the techniques of the *Semantic Web* [2] with local point–to–point data translations, rather than mapping to common mediated schemas.

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 is able to heal itself. In contrast to the broadcast

disks [1], 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 [1] with the major difference that our approach is not based on broadcast disks, but on a push–based replication strategy.

Most of the research on active multidatabases has been done concerning global integrity. Chawathe et al. [5] propose a toolkit for constraint management in loosely coupled systems. Worth mentioning is also the idea of Gupta and Widom to optimize the testing of global constraints by local verification [6]. Conrad and Türker [12] 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.

# 5 Conclusion and Future Work

We have presented an architecture, which connects heterogeneous and autonomous data sources creating a dynamic P2P database and acts without any central authority. Preserving local autonomy we have achieved, that each local administrator may decide on his level of participation. Data provided by other peers can be subscribed and integrated into the local database as needed, whereupon changes on the subscribed data and schema items are actively propagated to the relevant peers. The composition of the information grid can either be planned or evolve dynamically, similar to classical P2P systems, where we have identified link patterns.

In the next steps we have to concretize the concepts proposed, especially event detection and communication failures. Besides the implementation of a relational prototype with a concrete wrapper component, we have to specify a communication protocol including a sophisticated data exchange format (e.g. XML or OWL).

Due to its characteristics DÍGAME provides a promising infrastructure for a diversified application field, including e-business, e-science, e-government, or e-health.

# References


[1] Swarup Acharya, Michael Franklin, and Stanley Zdonik. Balancing push and pull for data broadcast. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 183–194, Tucson, Arizona, 1997. ACM Press.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, Mai 2001.

[3] Thomas Burns, Elizabeth N. Fong, David Jefferson, Richard Knox, Leo Mark, Christopher Reedy, Louis Reich, Nick Roussopoulos, and Walter Truszkowski. Reference model for dbms standardization, database architecture framework task group (daftg) of the ansi/x3/sparc database system study group. *SIGMOD Record*, 15(1):19–58, 1986.

[4] Bengt Carlsson and Rune Gustavsson. 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*, pages 347–354, Hong Kong, China, 2001. Springer.

[5] S. Chawathe, H. Garcia-Molina, and J. Widom. A Toolkit For Constraint Management In Heterogeneous Information Systems. In *Proceedings of the International Conference on Data Engineering*, pages 56–65, New Orleans, Louisiana, February 1996.

[6] Ashish Gupta and Jennifer Widom. Local Verification of Global Integrity Constraints in Distributed Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data SIGMOD'93*, pages 49–58, Washington, DC, 1993.

[7] Alon Y. Halevy, Zachary G. Ives, Peter Mork, and Igor Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *Proceedings of the twelfth international conference on World Wide Web*, pages 556–567, Budapest, Hungary, 2003.

[8] Dennis Heimbigner and Dennis McLeod. A Federated Architecture for Information Management. *ACM Transactions on Information Systems (TOIS)*, 3(3):253–278, 1985.

[9] Witold Litwin and Abdelaziz Abdellatif. Multidatabase Interoperability. *Computer*, 19(12):10–18, 1986.

[10] Cristian Pérez de Laborda and Stefan Conrad. A Semantic Web based Identification Mechanism for Databases. In *Proceedings of the 10th International Workshop on Knowledge Representation meets Databases (KRDB 2003), Hamburg, Germany, September 15-16, 2003*, volume 79 of *CEUR Workshop Proceedings*, pages 123–130. Technical University of Aachen (RWTH), 2003.

[11] Christopher Popfinger. Realisation of Active Multidatabases by Extending Standard Database Interfaces. In *Workshop on Foundations of Databases (Grundlagen von Datenbanken)*, pages 40–44. Faculty of Computer Science, University of Magdeburg, 2003.

[12] Can Türker and Stefan Conrad. 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*, pages 93–100, Los Alamitos, CA, 1997. IEEE Computer Society Press.