10

Enhancing the Data Openness of Frameworks by Database Federation Services

Ralf Böttger, Yaron Engel, Gerd Kachel, Silvia Kolmschlag, Dietmar Nolte, Elke Radeke
Cadlab – Joint Venture University of Paderborn and Siemens Nixdorf Informationssysteme
Bahnhofstr. 32, 33102 Paderborn, Germany
dbf-ma@cadlab.de

Abstract

Tools of existing frameworks can easily share data between each other by its integrated database system but data access to external database systems, file systems, or other frameworks is often not supported. We propose to incorporate database federation services into a framework to enhance the data openness of frameworks. These services allow a coupling of the frameworks database with external database systems while the autonomy of the individual database systems is preserved. Thereby, existing framework or database applications can continue running without any change. On the other hand, framework applications are enabled to access data of multiple databases uniformly and transparently via the old, possibly extended framework interface. This paper introduces the architecture of the proposed database federation services and presents two alternatives of their incorporation into a framework. The practical relevance is shown in an industrial environment where a framework is opened to the data of another framework and a PC database.

1 Introduction

Over the last several years, more and more CAx[1] systems use the benefits of a framework. All applications and tools of these systems can access the homogeneous user interfaces of the frameworks and exchange their data using the integrated database system (DBS) [4, 6, 9]. But the introduction of a framework is often handicapped by an empty or not sufficiently filled internal framework database system (FW DBS). Instead, the necessary data of the framework tools are already stored in file systems or other isolated, non-integrated database systems (see Fig. 1). The lack of data import facilities and database coupling services may impede the initialization of the internal framework database with the necessary data. Furthermore, different application areas in an enterprise often use different frameworks specific for their needs, e.g. CASE applications use a CASE framework. Each of these frameworks can have its own integrated DBS. Hence, there is a multitude of different, mostly isolated DBS in current enterprises. Although applications within a specific framework can share data, usually this is not possible in a controlled way to applications accessing other database or file systems.

[1]CAx is a place holder for CAD, CASE, CAM, CAE, EDA, etc.

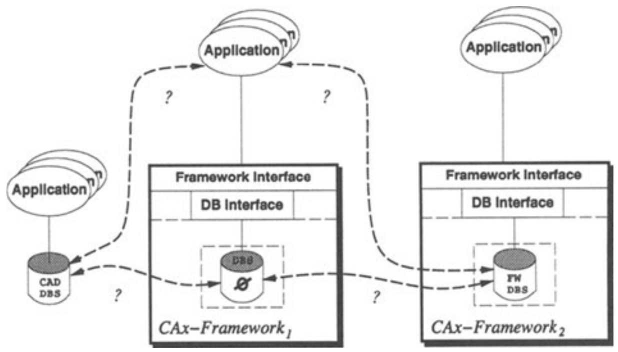

Figure 1: Frameworks lacking data openness

The introduction of database federation services (DBF) into a framework solves these problems and enhances the data openness of a framework. DBF offers a uniform, transparent access for framework applications to the FW DBS and to other DBS outside the framework. The chosen DBF architecture is very flexible and adaptable to the users needs. This is achieved by a modular, object-oriented design and generic software components.

In contrast to coupling techniques like import/export mechanisms or "gateways", DBF controls consistency across all coupled database systems. Furthermore, an extension of the coupled DBS is supported naturally by DBF but not by the other techniques. This good extensibility is achieved by the usage of a canonical data model within DBF and a generic adapter technique. Finally, we enhance the concepts of federated database systems [11] by different kinds of incorporation of DBF into a framework.

In Section 2, we present the DBF architecture and give an overview of its functionality. Its incorporation into a framework is explained in Section 3. The benefits of such an incorporation are stated in Section 4. To verify our concepts, we have started a pilot project with one of our industrial partners described in Section 5. Last, we summarize our paper and give an outlook to the future work in Section 6.

2 Database Federation Services

In order to exchange and share data with other frameworks, database systems, or file systems, they have to be coupled somehow. There are two alternative architectures to achieve this (Fig. 2): (1) a coupling layer for each data system realizes the mapping to the others or (2) a single coupling layer with a canonical data model maps to all data(base) systems back and forth. In both alternatives, users can access the data system directly via its interface or use the coupling layer in order to access data of multiple/other data systems.

The first alternative requires $n * (n - 1)$ mappings between data systems with n the number of data systems while the second alternative needs n mappings. But current enterprises have a multitude of data management systems, so n is big (e.g. 20 for a department of an industrial project partner). Moreover their distribution, heterogeneity, and autonomy pose many tasks to the mapping, e.g. data model and schema transformation as well as redundancy control [7]. Therefore, we chose the second alternative for our architecture in order to decrease implemen-

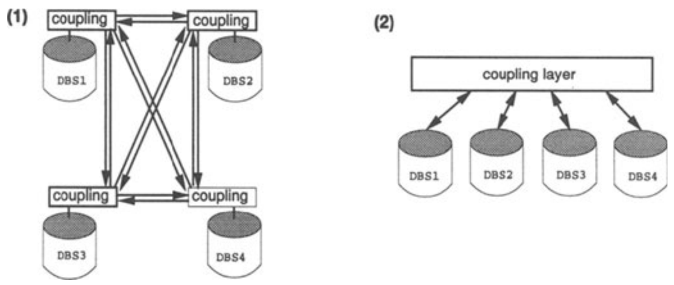

Figure 2: Alternative architectures:(1) point-to-point-connections, (2) canonical coupling layer

tation effort and ease extensibility with further database systems.

Fig. 3 presents our architecture in more detail. It represents a federated database system [11] preserving the autonomy of the distributed heterogeneous DBS. The DBS specifics are encapsulated in the DBS adapters while all other modules and interfaces of the database federation services are generic. When installing DBF in a specific environment, only DBS adapters for the enterprise's data systems have to be provided. The *DBS adapters* realize the communication with the various coupled database and file systems of the enterprise, e.g. with hierarchical, network, relational, and object-oriented DBS, or file systems. Moreover, they transform the heterogeneous data and operations into a canonical data model. For this, a coupling interface with 14 operations such as *createObj, deleteObj, loadObj, saveObj* is mapped to corresponding DBS data and operations. To give an impression of their complexity, a DBS adapter for relational DBS with an SQL interface needs about 3000 lines of C++ code (including comments) whereas that for an object-oriented DBS [6] requires only about 600 lines of C++ code. To ease the dynamic extension of the federation with new data(base) systems, there is a uniform layer (*database coupling*) on top of the DBS adapters.

As canonical data model, we use an object-oriented data model due to its expressive power [10]. It is based on the upcoming standard for object oriented database systems, the ODMG object model [2] which provides concepts like objects, classes, class inheritance, and object identity. We extend the ODMG data definition interface (ODL) by functionality for schema mapping/integration. Data are accessed either locally by the DBS interfaces of the autonomous data systems or globally via *global interface* which is based on the ODMG C++ OML. It allows a uniform and transparent access to data of all coupled DBS. Other interfaces are realized as *external adapters* on top of it, e.g. application specific interfaces or standard interfaces like SQL. In contrast to the DBS adapters, the realization of an external adapter is more expensive because the complete database interface has to be implemented and not only a subset. Nevertheless DBF will offer external adapters for standard database interfaces and only specific external adapters would produce extra development effort.

By the *administration interface*, the DBF is started, stopped, initialized, and DBS are coupled/decoupled to/from the DBF. The administration services are realized by three modules: *process management*, *coupling control*, and *user administration* (see Fig. 3). The user administration is responsible for the mapping from global DBF user to local DBS users. Beside these administration modules, the *federation kernel* contains several other modules which realize tasks such as query decomposition, global transaction management, and global redundancy control. A central role among these modules plays the *object manager* which is based on the context object

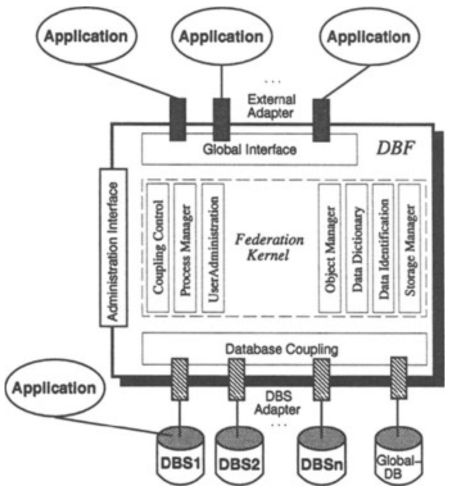

Figure 3: DBF architecture

model [5]. The object manager translates the global object accesses into local DBS objects (but still in the canonical DBF data model) and vice versa. These translations are done according to the 5 level schema architecture for schema integration in federated database systems [11]. The *object manager* uses the *data dictionary* module which stores the schema integration information and the *data identification* module which realizes the mapping of global resp. internal object identifiers to the heterogenous local DBS identifiers. The module *storage manager* is responsible for an efficient transfer of data between main memory and persistent storage including the caching of local DBS data.
Schema and configuration information as well as global new data like global relationships that do not map to any DBS are stored in an auxiliary database system. This *global DBS* is coupled via a DBS adapter.

3 Incorporation into a Framework

In this section, we present some new ideas for the incorporation of database federation technology into a framework. Figure 4 shows two possible alternatives how DBF can be incorporated into a framework. In both alternatives, DBF couples the internal framework DBS with other (external) DBS via the DBS adapters in a controlled way. Furthermore, a specific external adapter offers the complete framework database interface and manages the mapping between the framework data model and the canonical DBF data model. In the remaining section, we discuss both incorporation alternatives.

DBF as External Service

Alternative (1) couples the FW DBS in the classical way [11] whereby all coupled DBS are remaining autonomous. I.e. the existing applications and systems are not affected by the DBF (no relinking and no performance loss). But this architecture requires an already open FW

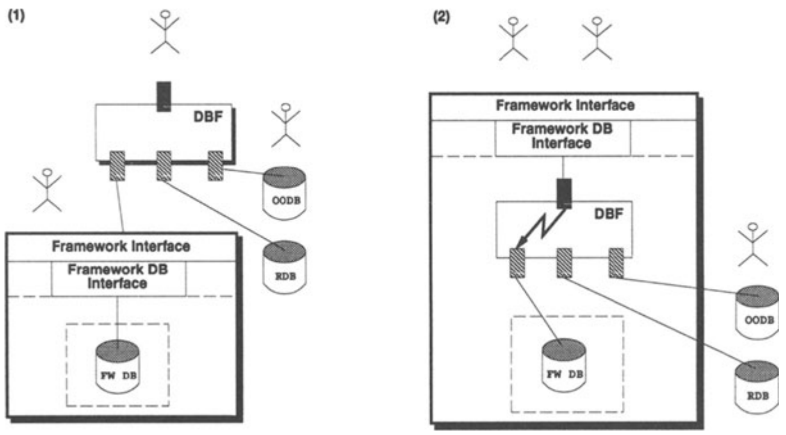

Figure 4: Two possible incorporations of DBF into a framework: (1) external and (2) internal

DB interface. I.e. the framework has to offer the database interface as a separate service which is not encapsulated into other framework services. This alternative allows a very easy introduction of DBF into an existing environment using framework technology. Neither the existing framework(s) nor their (local) applications have to be changed. On the other hand, new (global) applications can access the federated data of different systems coupled by DBF. But there are consistency problems between the local and the global applications [1], e.g. only *quasi-serializability* can be guaranteed [3].

DBF as Internal Framework Service

In contrast to alternative (1), alternative (2) allows full consistency control for all framework applications. Therefore, any data access using the framework interface is passed through the external DBF adapter which includes a consistency check for all federated data.[2] But the general consistency problem between the framework applications and applications using only the external DBS still remains.

This alternative implies a change of the internal structure of the framework but the framework interface is stable. Thereby, existing applications could access (more) federated data via the same framework interface in a consistent way. This is also true if the set of coupled DBS changes e.g. by the coupling of a new DBS. A relinking of the existing framework applications becomes necessary if the incorporation of DBF into a framework affects the framework library which has to be linked by the applications. In a client-server architecture, where only the framework server changes through the incorporation of DBF, no relinking of the applications (clients) is necessary.

The passing of all data accesses via the external DBF adapter and the internal data model transformation may produce a performance loss for the complete framework. This is not acceptable for framework applications which are using data stored only in the FW DBS. To avoid this performance loss, unnecessary data model transformations between the framework database interface using the framework data model, DBF using the canonical DBF data model, and the FW DBS using again the framework data model are shortened. Therefore, the external adapter

[2] A detailed description of the consistency check will be presented in a future paper.

directly maps a data access to the internal FW DBS without using the DBF kernel if no other (external) DBS is affected by this data access. I.e. the DBF kernel becomes only active if a federated, external DBS is involved for the data access. These shortcuts of the external adapter allow an efficient processing of the original framework data. The DBF concept distinguishes between different kinds of shortcuts which are not discussed in this paper but we plan to prepare another paper with this topic.

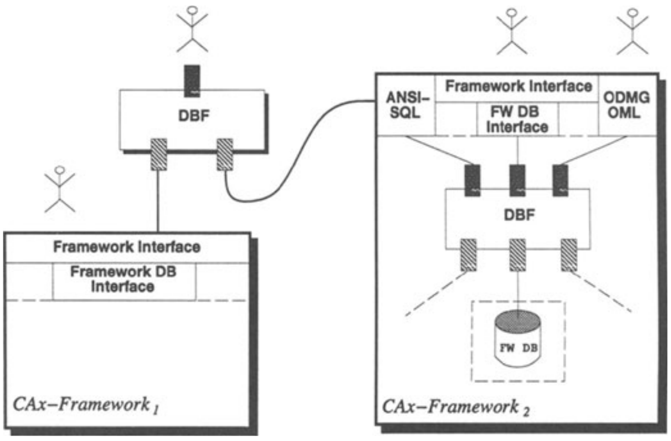

Figure 5: Coupling of different frameworks by using DBF

Beside enhancing the framework data access facilities by the federation of other (external) DBS, DBF can also enhance the openness of the framework to other frameworks or systems. This is realized by a further external DBF adapter, e.g. a standard database interface like ANSI-SQL or ODMG C++ OML. The "standard" external adapter allows a standardized data access to all federated DBS including the FW DBS. In alternative (2), this further external adapter implies an extension of the FW interface by a new "standardized" database interface (see Figure 5). The extended framework interface can be used by other applications which are not integrated into the framework or can be used for coupling the framework with another framework via a second external DBF.

This enhanced data openness of a framework is the first step in the direction of *framework interoperability*. DBF offers here a generic, powerful, and consistent solution.

4 Benefits

Openness is an important intention for any framework to support the exchangeability and integration of any tool or data. The integration, especially a deep integration, provides uniform access to shared data. For example, the integration of an expert system, which contains its own knowledge base system within a framework may provide access to shared data. The complete benefits of openness [9] can only be achieved through a standardized tool interface. The database federation services enable, via the global and external interface, the "standardized" access and interchange of heterogeneous data between tools within the framework and, nevertheless important, between tools of various frameworks as well.

The major benefits by using the database federation services are listed in the following:

- Existing framework applications or applications of the coupled data systems can continue running so that no data or application has to be migrated or recoded.
- Because existing database or file systems do not need to be replaced but can be supplemented by new ones, investments will be saved.
- The relaxation of the strong connection of applications to one or only a few special database systems allows an easy exchange and switch between different technologies.
- Common data do not have to be stored redundantly in different database systems or file systems.
- New framework applications can uniformly and transparently access data of the framework's integrated database system as well as other coupled database and file systems. Hence, applications crossing multiple frameworks can also easily developed.
- The database federation services control consistency across all coupled database and file systems. Thereby, consistency checks are not restricted to the framework's integrated database system. Moreover, consistency is performed also for operation sequences that access multiple databases in terms of global transactions.
- The framework user may define and access relationships among data of different database or file systems.
- The whole company can benefit because the various departments can easily share and exchange data which eases concurrent engineering as well as enterprise integration. Moreover, the possibility to realize one/few information models for the whole company resp. cooperation makes global plans and decisions much easier for the management.

5 Application in a Pilot Project

An important part during the realization of the database federation services is the verification of our concepts by a pilot project within a company where problems solvable by DBF can be found. Therefore, an application on base of a DBF pre-release will be realized within a department of Siemens Nixdorf Informationssysteme (SNI).

Within the SNI department, there are several heterogeneous systems for managing hardware component data. Therefore a CAD framework consists of several systems for handling CAx hardware component data and stores technical information like the component's technology the relational database ADABAS[3] offering an SQL interface. An electronic design framework realizes a development environment for compatibility checks of electro magnetical circuits and is supported by various tools and a data handling system based on file system.

Within this environment, many applications have to access data of different database systems or file systems. This had been realized by import/export functions supported by a cross reference list for identification of data. To ease this large-scale process, the department requested a system which offers coupling of both frameworks and allows access to all database systems and

[3]ADABAS is a trademark of SOFTWARE AG.

file systems. Therefore, the actual problem is the lack of efficient support for data exchange between these data handling systems of the frameworks.

The DBF pilot concerns the controlled data coupling of the CAD framework on a DEC workstation, the electronic design framework on a SUN workstation, and a relational database for personal computers, namely ACCESS[4], as a hardware component library where data of the two frameworks shall be stored for the usage of PC tools. For realization, we chose the first alternative discussed in Section 3 where DBF is an external service of the furtheron autonomous frameworks. The following data handling systems will be federated by DBF:

- relational database ADABAS storing CAx data of the CAD framework
- Unix file system library storing the hardware component data of the electronic design framework
- relational database ACCESS for PC storing hardware component data for the administration of component data of the electronic design framework.

The necessary data exchange and data conversions will be managed by DBF on a SUN workstation. The schemas of the file system and the relational database ADABAS will be unified and the resulting federated schema will be stored in the PC database. The administration of the electronic design framework uses the PC database to initialize the data exchange. Global relationships between electronic design components and CAD components will be stored in the internal DBF database. The federated data will be accessed via an external adapter of the *global DBF interface* the electronic design administration via remote shell.

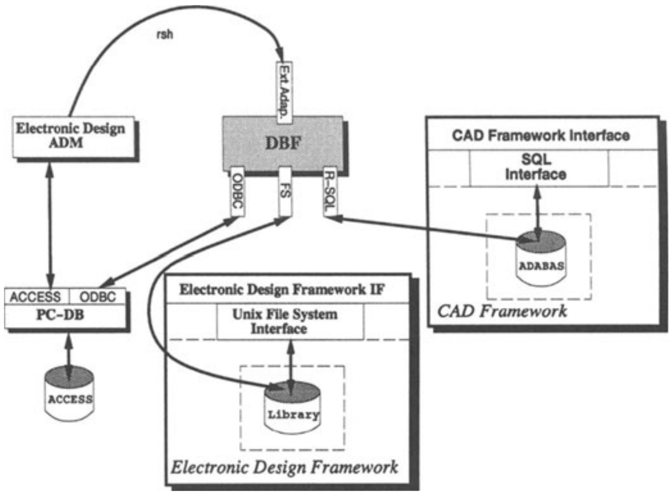

Figure 6: Data handling systems coupled by DBF pilot

For connections from DBF to the different data handling systems, different adapters have to be provided offering different interfaces to the data handling systems (Fig. 6):

[4]ACCESS is a trademark of Microsoft.

- A *Remote SQL adapter* realizes data exchange between ADABAS on a DEC workstation and DBF on a SUN workstation across hardware borders.
- A *file system adapter* offering a hierarchical oriented file format connects the Unix file system library storing electronic design data to DBF. The electronic design framework offers such a hierarchical oriented file format interface providing direct access to the file system library.
- An *ODBC adapter* ("Open Database Connectivity")[5] couples the PC database ACCESS to DBF. The usage of ODBC allows a direct coupling of the database without creation of temporary import/export files.

The concepts of federation and migration have been realized for the DBF pilot. Within the SNI department, the federation of the two frameworks provides a uniform and transparent interface to all federated systems. The migration of hardware component data supports the reduction of database and file systems and the realization of a single hardware component library. For administration of hardware component data, efficient data access will be realized and data consistency will be controlled automatically.

6 Conclusion

We showed the useful enhancement of frameworks by database federation services (DBF). These new services enable framework applications to access beside the internal framework database system also existing external database systems resp. other frameworks in a transparent and uniform way. This enhanced data openness of a framework is the first step in the direction of *framework interoperability.*

The presented DBF architecture is very flexible and adaptable to the users needs. This is achieved by a generic adapter technique between applications and DBF as well as between DBF and (autonomously) coupled databases. The DBF kernel is realized as a toolbox of configurable, object-oriented services. This can be used in different kinds of incorporation between DBF and frameworks. Our approach enables multiple kinds of framework incorporation and additionally, uses object-oriented implementation technology which allows itself federation of any data from flat relations to highly structured objects. This combination leads to a variety of benefits for framework users. Some of these benefits are reflected by our pilot application where concepts will be proven and our implementation will be improved.

Beside the pilot application, we are providing our concepts for the JESSI[6] framework JCF [6]. Since our first DBF release will provide basic functionality, it is obvious to enhance functionality in detail in future, e.g. more sophisticated transaction mechanisms and global recovery mechanisms. One of the more general extensions is the migration of data from one database to another as well as migration of applications (tools) from one database to another [8].

[5]ODBC is a trademark of Microsoft.

[6]This work is granted by the ESPRIT project JESSI COMMON FRAME No. 7364.

References

[1] BREITBART, Y., GRACIA-MOLINA, H., SILBERSCHATZ, A. Overview of multi-database transaction management, 1992.

[2] CATTELL, R.G.G. *The object database standard: ODMG'93.* Morgan Kaufmann Publisher, 1994.

[3] DU, W., ELMAGARMID, A.K. Quasi-serializability: a correctness criterion on global concurrency control in InterBase. In *5th Int'l Conf. on Very Large Data Bases (Amsterdam, Netherland)*, pages 347–355, 1989.

[4] European Computer Manufactures Association (ECMA). Reference model for frameworks of software engineering environments, 3rd edition, 1993.

[5] KACHEL, G. A multilayered database system architecture for supporting tool integration. In *Proc. of the 2. International IFIP WG 10.2 (Charlottesville)*, 1990.

[6] KRANNICH, G., DIPPER, H., FOX, W., KATHOEFER, T., SCHLEGEL, W., SCHMID, J. Jessi-Common-Framework V4.0, Global View, Detailed Functional Specification SP2. JCF Report JCF-SNI/055-01/29-Apr-94, 1994.

[7] NOLTE, D., KOLMSCHLAG, S. , KACHEL, G., RADEKE, E. Coupling Services for Heterogeneous Database Systems, Jessi-Common-Framework V4.0, Detailed Functional Specification. JCF/CADLAB/068-01/11-April-94.

[8] RADEKE, E., SCHOLL, M.H. Federation and stepwise reduction of database systems. In *Proc. 1st Int'l Conf. on Applications of Databases (Vadstena, Sweden)*, 1994.

[9] RAMMIG, F.J., STEINMÜLLER, B. Frameworks and design environments (in german). *Informatik Spektrum 15*, pages 33–43, 1992.

[10] SALTOR F., CASTELLANOS M., GARCIA-SOLACO M. Suitability of data models as canonical models for federated databases. *SIGMOD RECORD 20(4)*, pages 44–48, 1991.

[11] SHETH, A., LARSON, J. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys 22(3)*, pages 183–236, 1990.