10

Data Integration: a Federated Approach with Data Exchanges

G.N. Benadjaoud, B.T. David
Ecole Centrale de Lyon, Labora to ire GRACIMP, Dépt. MIS
B.P. 163, 69131 ECULLY CEDEX, FRANCE
Tél: (33) 72.18.64.42 Fax: (33)78.33.16.15
email: benadjao@ cc .ec-lyon.fr, david@ cc.ec-lyon.fr

Abstract

In this paper we present an integration approach based on data federation and data exchanges. The data exchange approach is confronted with two problems. The first one is the heterogeneity of DBMS that these applications use. The second one concerns the modelling technology and the perception of the real world which can be different for different applications. This approach is supported by a platform called DEE (Data Exchange Environment). The DEE allows the co-ordination and the control of data exchange between different applications while getting over the heterogeneity and conflict problems. It preserves existing applications which are valuable acquisitions of the organization and allows the integration of new applications to the data exchange environment It insures reliability and security of data during data exchanges and gives a global view of manipulated data.

Keywords

Integration, databases, federated databases, data translation, data exchanges.

1 INTRODUCTION

SEPTEN, one of the divisions of Electricité De France (EDF), studies the security in the power stations. Due to the importance and the complexity of its task, SEPTEN uses more than 200 applications that simulate and control the running of the stations. These applications assist engineers in the study of the behaviour of station components during a 'smooth' running and during a technical hitch. The studies focus on the different aspects of the stations: neutronic, hydraulic, mechanical and thermic.

Each application can focus on one or several aspects of the station, and has its own data and its own data manager without any interaction with other applications. An application may need the results of one or more other ones, but there is no means to transfer data. The data transfer is done manually by the users who analyze the outputs of some applications to prepare the inputs of others. Usually, this task takes up to nine months. Adding to this, the users have to deal with data inconsistency consequence of data redundancy.

The data integration of isolated applications is to let applications share and access the common data. The approaches mainly used are the one based on the application interfacing and the use of standard format and the one based on the federated databases.

L. M. Camarinha-Matos et al. (eds.), *Balanced Automation Systems II*

Interfacing applications is to create between each two applications that have to exchange data, a data translator. To ease this interfacing and to reduce the number of translators, standard formats have been defined, like IGES (Initial Graphics Exchange Specification) (Wix, 1986; Scholz, 1992), SET (Standard d'Echange et de Transfert) (Wix, 1986; Scholz, 1992) and STEP [Scholz, 1992; Brun 1992). They homogenise the data expression between applications. To exchange data using standard formats, the data are translated from the source application to the standard format and from the standard format to the target application. So, for each application two translators are needed : one for the data transfer from the application to the standard format, another one for the data transfer from the standard format to application.

The Federated Database Management System (Chung, 1990; Sheth, 1990; Ahmed, 1991; Clement, 1994) groups a set of DB systems and makes them cooperate under a federation. The advantage is that the DB systems continue to carry their local operations while participating to the federation.

Constrained to preserve the existing applications which are valuable acquisitions of the companies, our data integration approach is a federated one with data exchanges. We designed a Data Exchange Environment that allows the isolated applications that may use different DBMS and different file systems, exchange their data. In this paper we present a data exchange environment (DEE) that allows applications exchanging data despite their location and the data manager used by each of them. This environment preserves existing applications which are valuable acquisitions of the organisation and allows the integration of new applications to the data exchange environment. It also insures reliability and security of data during data exchanges.

2. A DATA EXCHANGE AGENT (DEA)

A Data Exchange Agent is in charge of data transfer between applications connected to it, solving perception and data representation heterogeneity that may exist. It manages application data import and export. It is directed by an Extensible Reference Model (ERM) which contains the schema of the data shared by the connected applications and the export schema of each connected application.

3. THE DATA EXCHANGE AGENT ARCHITECTURE

In our concern to get an open architecture environment which allows connecting new applications, we have structured the DEA in several components with precise functionalities. This organisation makes easier the identification of connected applications dependent components and the independent ones. The application dependent components must be redesigned or at least adopted for each new application. To facilitate this specialisation we designed a set of tools which can be used in this integration process.

The DEA architecture, illustrated by figure 1, lies as main component Data Exchange Handler (DEH) which is responsible for the data transfer from an application to another. The DEH is the intermediate between the client application and the supplier application. When receiving request from an application ; the DEH localises the supplier application, extracts requested data and sends them to the client application. To perform these functions, the DEH uses the information in the ERM describing data model of this thematic group and export schema of each application.

The connection of applications which initially were not conceived to use the DEA is done via local interfaces. These ones handle the data representation and data access differences that might exist between applications and the DEA. The duty of a local interface is to facilitate the data access and make the DEH independent from the connected applications.

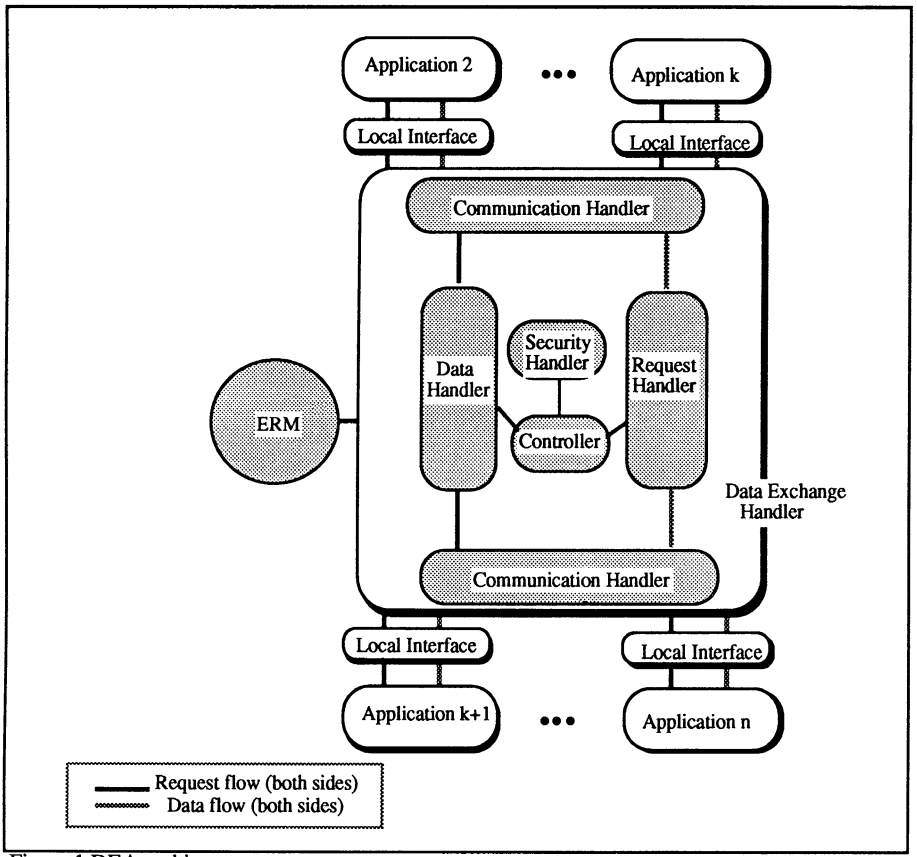

Figure 1 DEA architecture

3.1 The ERM

The ERM (Extensible Reference Model) is a schema of data shared by connected applications. It unifies the perception and the representation of data used by different applications.
To allow ERM supporting complex entities representation, view point representation and dynamic and static data coherence preservation, a canonical data model, called MO-MER (Benadjaoud, 1994a), has been chosen. This model is a ODMG (Atrood, 1993) inspired model, it responds to our requirements and allows the integration of future applications using object oriented DBMS.

The ERM is a reference for shared data between connected applications. It offers a view of manipulated objets and their relationships for a group of applications. The ERM is also used by the DEH, it helps in switching and dispatching data. In addition to the schema describing the shared data (the shared schema), the ERM contains, for each connected application, its export schema. Thus it contains the application data offers. It identifies the set of data which the application would like to share with others.

A set of correspondences link between export schemas and the shared schema. It matches each entity in the shared schema to its similar entities in the export schemas. The representation of

these links is a tree representation where the root is the shared schema and the leaves are the export schemas. Thus, it simplifies and facilitates the request and the data dispatching managed by the DEH.

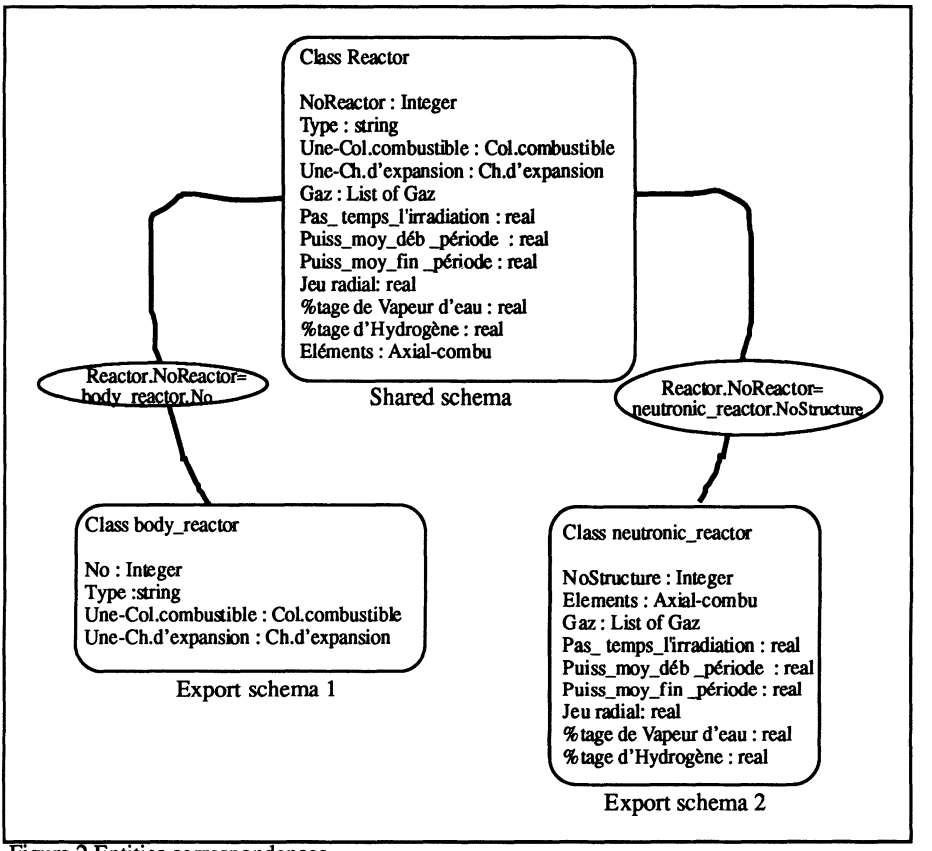

Figure 2 Entities correspondences

A correspondence example is illustrated by figure 2. An elementary reactor modelling is represented. The neutronic aspect of a reactor, which can be used, for example, by nuclear reaction simulation softwares, is represented in export schema 2. The body part of a reactor, which can be used, for example, for accident simulation and precisely for testing the resistance of the reactor partitions to any link, is represented in export schema 1. The correspondence between Reactor Class in the shared schema and Neutronic_Reactor Class is illustrated by an arc and a condition which put in correspondence the Reference Number of the Reactor Class and the Reference Number of the Neutronic_Reactor Class.

3.2 Data Exchange Handler (DEH)

The data exchange handler is a data dispatcher. Each information exchange between connected applications passes through it. The DEH is in charge of the services offered by the DEA : Wide information access, Information distribution, Data consistency control, Information flow control and Data Deduction. These services are executed by five components that compose the

DEH. All components are in relation with the ERM, concerning information on objects, their characteristics and their locations.

Requests handler

The local interfaces communicate with the DEH through a data manipulation language ; close to SQL language, called DEL (Data Exchange language). This language allows the expression of data consultation, data modification and data deletion.

The data requests handler is responsible for interpreting requests and dispatching them over concerned applications. It is constituted from two components :

- an interpreter that analyses requests and builds the correspondent syntactic trees,
- a request decomposer that decomposes a request into a set of sub-requests destined to the concerned applications. This decomposition is done by using a syntactic tree and information from ERM concerning the data location.

The request decomposition is supervised by two controllers. The first one is the user controller ; it verifies that the user has got the data access rights. The second one is the information flow controller ; it verifies that the information flow is respected.

Data Handler

The data handler is responsible for the forwarding of the data to their destination. Three principal functions appear in the data handler :

- Data grouping : In a case of a data extraction there is one client application (asking for data) and eventually several supplier applications. Once the suppliers respond to the client, the data are grouped and then presented to local interface of the client application.

- Data decomposing : In a case of a data insertion or data modification, there is one supplier application and one or several client applications. The data sent by the supplier are divided into packets and each packet is sent to the concerned application.

- Data deducting : The deductible attributes and their calculating functions are known by the ERM. The deductible data are calculated at the data grouping, the data handler verifies the arrived data and then fires the calculating functions for deductible data before their dispatching over client applications.

Controllers

The controllers guarantee a reliable running of the DEA, a transferred data reliability and security. We distinguish three controllers :

- User Controller : verifies that the data use respect the rules fixed by the security handler.
- Flow controller : is activated before request and data distribution over the concerned applications. Its duty is to verify that the data do not move in a opposite direction of the defined flow.
- Data Controller : serves avoiding any conflicts or any incoherence between applications data. It verifies that the data respect the local constrains laid down in the applications and the global constrains laid down in the ERM.

Communication handler

The communication handler is the component that uses the network potentialities to forward the data to the applications. It is in charge of transmitting and receiving data to/from local interfaces. It is also in charge of the communication synchronisation necessary when the applications DBMS have not a same time response. Thus, before grouping the responded data, the communication handler verifies that all asked data have been received.

Security handler
Each connected application is responsible for the security of its used information. Hence, it is not enough as the application data can be shared by others, thus accessible by other users who may have no right to use the application data. So a global security handler system is needed. It manages the users access right to the shared data. It allows the authentication of users defined by their identities and passwords. The users are catalogued by the access right they have to manipulate the objects catalogued by their protection levels.

3.3 Local Interfaces

To avoid expensive modifications of existent applications to connect the DEA and to conceive independent DEAs are the aim of the local interfaces. These transfer data from applications to the DEA and vice versa. They resolve incompatibility problems that exist between the data model used by the DEA and the data model used by the application. Thus, they are dependent on the applications they connect and particularly on the DBMS used by the applications. This means that they are dependent at once on the DBMS data model, on the DBMS pre-defined types and the DBMS data manipulation language.

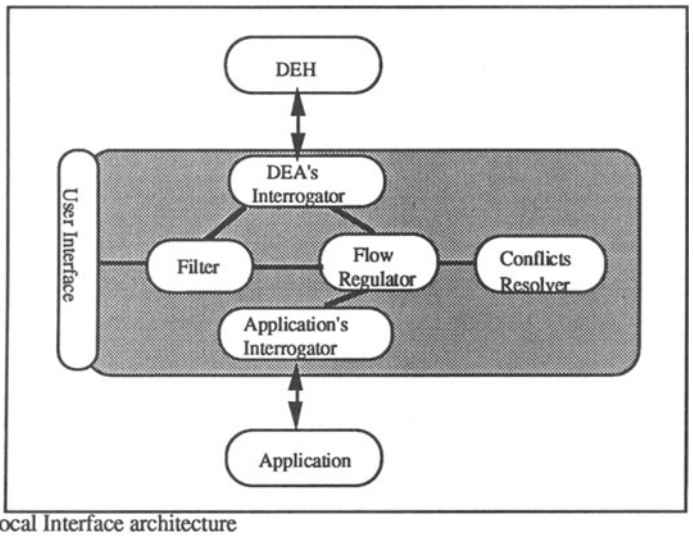

Figure 3 Local Interface architecture

The general architecture of a local interface is conceived in a way to include all forms of interfaces connecting different kinds of applications. The figure 3 illustrates the components that constitute the interface. These components are :

• *The DEA Interrogator* : Contains a set of primitives to communicate with the DEA. These primitives allow importing and exporting data, sending and receiving requests.

• *The application interrogator* : Contains a set of primitives for accessing and updating application data. This component is tightly dependent on the application DBMS.

• *The Filter* : In a case of a DEA-application, it locates in the DEA, the data pointed out by the user to be transferred in the DEA. In case of an application-DEA, it locates the data pointed out by the DEA requests to be transferred, in the application storage spaces.

• *The conflicts resolver* : resolves conflicts, that may occur during the transfer, between the data definition in DEA and the data definition in the application. These conflicts can be type conflicts or interpretation conflicts. The type conflicts occur when the type definition of the data in the DEA differs from the type definition of the same data in the application. In this case the conflict is resolved by applying a type transformation function. The interpretation conflicts occur when the data value is relative to a unit of measure or to any other unites. So the values of the same data can be different whether they are in the DEA or in the application. For example, a temperature can be expressed in degrees centigrade in the application and expressed in degrees Fahrenheit in the DEA. This kind of conflict is resolved by a value transformation functions during the data transfer.

• *The flow regulator* : co-ordinates the data transfer and synchronises the DEA interrogator activities with the application interrogator activities.

• *The user interface* : It is the communication mean between the local interface and the user. It allows the user to select the data to be transferred.

4. DATA EXCHANGE ENVIRONMENT

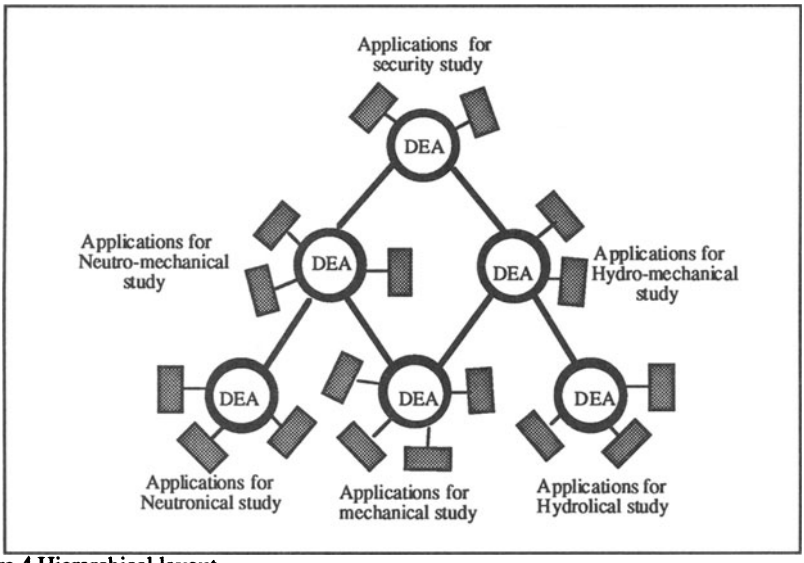

Figure 4 Hierarchical layout

The DEE is a network of distributed Data Exchange Agents (DEA). To each DEA agent is connected a certain number of applications that need to pass on data to one another. The global schema of all shared data is distributed over the DEAs ERMs. In this approach global ERM equally does not exist, we have only partial ERMs in relation with each DEA. The connection of a DEA agent to another DEA agent is handled in the same way as the connection of an application to a DEA agent and the same communication protocol is used for the communication between DEA agents and between applications and DEA agents.

The layout of DEAs in the DEE used in EDF/Septen is an hierarchical one : the DEE is an hierarchy of DEA agents. The communication between two DEA agents is done via another DEA agent from a higher level. This layout fits the information system pyramid. This layout

can be applied in any multidisciplinary area where applications are involved in one or more subjects. The applications involved in one subject can be grouped, by subjects, around agents at the bottom of the hierarchy. The applications involved in two subjects will be connected to an agent at the upper level. Finally, the applications involved in all subjects can be connected to an agent root of the hierarchy.

5. CONCLUSION

The flexibility and the openness are the major advantages of the DEE approach. However, the implementation of a local interface, for each application to be connected to the DEE, is necessary in the integration process.

To facilitate this step of the integration process a Local Interface Specification Tool (LIST) is conceived (Benadjaoud, 1994b). The LIST allows the design and the implementation of local interfaces. It is an open set of generic local interfaces. For each DBMS and file system a generic local interfaces is dedicated where data transfer and conflict resolution functions are specified. The local interfaces are instanciations of the generic ones. The LIST generates local interfaces from users specification which are : the application export and import schemas and the type of the DBMS or the file formats used by the application.

6. REFERENCES

Ahmed, R.; Smedt, P.D.; Du, W. ; Kent, W. ; Ketbachi, M.; Litwin, W. A.; Rafii, A. and Shan, M.C. (1991) The pegasus heterogeneous multidatbase system, IEEE Computer, Vol. 24, No 12, December 1991, p.17-p.27.

Andersson, M.; Dupont, Y.; Spaccapietra, S.; Yetongnon, K.; Tresh, M. and Ye, H. (1993), The FEMUS Approach in Building a Federated Multilingual Database System, 3rd International Workshop on Research Issues on Data Engigneering : Interoperability in Multidatabase Systems (RIDE-IMS'93), Vienna, Austria. April 19-20, 1993.

Benadjaoud, G.N. (1994), Le modèle MO-MER, RR-MIS-94-12, MIS ECL Lyon Août 1994.

Benadjaoud, G.N. (1994b), LIST : un outils de spécifications des interfaces locales, RR-MIS-94-16 , MIS EC-Lyon Decembre 1994.

Brun, J.M (1992) IPDES final report : Common Data Model, Esprit Project 2590, june 1992.

Chung, C. W.(1990), DATAPLEX : An Access to Heterogeneous Distributed Database, Communication of the ACM, Vol. 33, No. 1, January 1990.

Clement, D.; Ganesh, M.; Hwang, S. Y.; Lim, E. P.; Mediratta, K.; Srivastava, J.; Stenoien, J. and Yang, H.R. (1994), Myriad : Design and Implementation of Federated Database Prototype, Proc of 10th IEEE Data Eng. Conf., 1994.

Atrood, T.; Duhl, J.; Ferran, G.; Loomis, M. and Wade, D. (1993), The Object Database Satandard: ODMG-93, Ed. R. G. G. Catell, 1993.

Scholz-Reiter, B. (1992), CIM Interfaces, Concepts, standards and problems of interfaces in Computer-Integrated Manufacturing, ed. Chapman and Hall 1992.

Sheth, A.P. and Larson, J.A. (1990), Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol. 3, No. 22, p. September 1990.

Wix, J. and McLelland, C. (1986), Data Exchange between Computer Systems in the Construction Industry, BSRIA 1986.

G.N. Benadjaoud is an assistant lecturer in computer science department at Ecole Centrale de Lyon. He works on data integration problems in manufacturing and on object oriented modelling.

B.T. David is a computer science professor at Ecole Centrale de Lyon. His research interests are in the areas of Human-Computer Interfaces Development, Computer Integrated Manufacturing and Computer Supported Cooperative Work (CSCW).