A unified architecture for data and message management*

by GEORGES GARDARIN*

Institut de Programmation
Paris, France

INTRODUCTION

The ANSI/X3/SPARC study group on data bases has proposed a general architecture for a data base management system. The keystone of this architecture is the conceptual schema which is an explicit description of the enterprise informations modeled in the data base. It portrays the entities, properties and relationships of interest in the enterprise and constitutes a stable platform in order to map the external schemas which describe the data, as seen by the programmers, onto the internal one which defines the data as seen by the system. Finally the ANSI/X3/SPARC DBMS study group envisions a three-level organization which induces three levels of administration functions, three levels of schema processors and three levels of data manipulation modules. This architecture is partially represented Figure 1. It should be noted that only functions are specified, not their implementation. The ANSI/X3/SPARC study group on distributed systems has proposed a reference model. This model allows application processes possibly located on different systems to exchange messages through sessions which logically join them. The major contribution is the structuration of the cooperating systems into layers. Each layer can be seen as a distributed sub-system. Cooperation between the distributed parts of a layer is governed by a set of protocols specific to the layer. Six, and now seven, layers have been identified. The first three, and now four, layers provide a universal transport service. The next layer supports interaction between cooperating application processes—it performs their binding and unbinding by sessions and controls the exchange of data through these sessions. The presentation layer enables the application processes to interpret the meaning of the data exchanged by transforming them into the desirable representation, format or model. The major concepts of this reference model are represented in Figure 2.

In this paper, the architecture of a distributed data base management system integrating the two proposals is presented. The basic assumption which guarantees the feasibility is that the geographical localization of data has a conceptual meaning. Consequently, the conceptual schema can be distributed, each part representing an enterprise department's description as modeled in a local data base. Obviously, an external schema is entirely situated on a site. The mapping of a user request which refers to an external object into one or several conceptual requests is performed by an external presentation module which can be seen either as an external to conceptual transformer or as a presentation control service. The conceptual requests which result are then taken in charge by a communication kernel which performs at first the session control functions of the ANSI/X3/SPARC distributed system reference model. In addition, it also performs the data base concurrency, recovery and security controls. After possible transmission through the transport management layers, the requests are submitted to conceptual data base managers which execute them. The answers follow symmetric paths in the distributed system. Finally, the external presentation module is responsible for constructing the concluding unique user answer.

This paper is organized as follows: In the first section, it is argued that, at least for a large class of applications, the distribution of objects is performed at the conceptual level. Consequently, an implementation of the conceptual schema in a distributed system is suggested. In the second section, the ANSI/X3/SPARC DBMS architecture is distributed in a computer network. That entails the development of a communication kernel allowing the interchange of the conceptual data manipulations through the transport network. Moreover, each local metabase which contains the local part of the conceptual schema must be accessible from remote computers; this feature permits the binding of an external schema with the distributed conceptual schema. In the third section, the communication kernel is presented using several concepts proposed in the ANSI/X3/SPARC distributed system contribution, mainly at the session control level. In the last section, a reference model for a distributed data base system is proposed. Data base and message management are integrated—(a) The external-to-conceptual transformer of the DBMS architecture is coalesced with the presentation controller of the distributed system architecture. (b) The read/write data base commands and the send/receive message commands appear as two different presentations of data interchange basic actions. (c) The integrity controls of

* Sponsored by IRIA, Le Chesnay, France.
CONCEPTUAL DATA DISTRIBUTION

Conceptual realm of data distribution

Several papers dealing with distributed database management system architecture assume that only the internal schema is geographically split and that localization of data concerns the conceptual to internal level mapping. Other papers are quite ambiguous on this topic. Our opinion is that at least for a large family of applications, the distribution of data is connected with the view of the enterprise. This opinion seems to be in agreement with the SDD architecture.

Such an opinion has been verified on real applications—first, the ordering, manufacturing and delivery of vehicles at the French motor car company Renault. For example, the fact that Renault model R30 is manufactured in Flins and that all the entities concerning a car of this model must be stored on the Flins site belongs to the enterprise description and should be explicated at the conceptual level. For this application, the localization criterion of an entity is the place of the modeled object in the real world. This is not directly connected with efficient use of computing facilities. On the other hand, the transfer of R30 entities from Flins to Paris in the distributed system would doubtless be of interest for the life of the enterprise—it would mean that the manufacturing of R30 model is moved from Flins to Paris.

The conceptual meaning of object localization has also been verified on other applications, like the management of spare parts. In the Renault real world, spare parts are distributed to two stores. Consequently, in the Renault computer world, they will be distributed in two local data bases managed by two interconnected computers. The localization criterion of objects is the modeling of the distribution of spare parts in the real world which is a conceptual property.

Conceptual schema design and implementation

The conceptual schema contains the definition of conceptual objects and the expression of their properties. In distributed data bases, the first property of conceptual objects is to be distributed. Therefore, this property can direct the
design of the conceptual schema in agreement with the geographical distribution of sites. Thence, it is proposed to divide the conceptual schema in parts, each part corresponding to a local site. Obviously, it may be argued that some properties involve objects localized on different sites, such as distributed integrity constraints. They must be known in each concerned part; that means that when the conceptual schema is divided, the distributed properties must be integrated to each interested part.

It is emphasized that only one conceptual schema must be designed—to provide a coherent conceptual view of the distributed enterprise couched on a unique booklet using the conceptual model should enhance significantly the interest of the system. The implementation of each part of the unique conceptual schema is performed on each site which must present conceptual objects to other sites in agreement with the conceptual schema. Thus, it should provide a distributed stable platform to which local internal schema and global external ones may be bound (see Figure 3). The design process of the conceptual schema may be distributed (each site designing its part) or centralized (designed by some enterprise administrators), but the existence of such a platform is very important in distributed enterprises because it defines the objects which may be interchanged between the departments.

ANSI/X3/SPARC DBMS ARCHITECTURE DISTRIBUTION

Assuming that the conceptual schema can be divided in local parts, the ANSI/X3/SPARC DBMS architecture can be easily distributed in a computer network. In order to simplify the presentation, host computers of the network are classified in two types:

- **Data computers**, which contain a set of data stored in a local data base.
- **Processing computers**, which execute external data base application programs.

In the last section, an integration will be presented where

![Figure 2—Distributed system levels.](image)

![Figure 3—Schema binding and division. (1) External schema (several external models). (2) Conceptual schema divided in local parts (one model). (3) Internal schema (one internal model for each computer type).](image)
each host is simultaneously a data computer and a processing computer.

Data dictionary directory

A key point of the ANSI/X3/SPARC DBMS architecture is the data dictionary directory. It can be seen as a metadata base where every DBMS processor fetches the parameters required by its execution. In order to implement the ANSI/X3/SPARC DBMS architecture with data computers and processing computers, it is necessary to distribute the elements of this data dictionary directory. The main elements of interest are:

- The user program descriptions
- The external data base schema object type descriptions
- The mapping structures relating external and conceptual objects
- The conceptual data base schema object type descriptions
- The mapping structures relating internal data base and conceptual one
- The internal data base schema object type description

A large part of these elements is now easy to distribute. By hypothesis, a user program description and an external schema are located on a processing computer. In the same way, each data computer needs at least an internal schema to describe the internal structure of its local data base. Then, as the conceptual schema is divided in local parts, it is desirable to implement each part on the corresponding data computer, particularly in order to avoid duplication of the whole conceptual schema on each processing computer. Consequently, mapping structures relating internal and conceptual level are situated on data computers.

The only point to discuss is the place of the mapping structures relating external and conceptual objects. As they are strongly dependent on the model seen by external application programs, localization on each computer which uses them improves the independence between computers. Let us point out that an external schema can be mapped on the whole conceptual schema; thence, an external object can be mapped into many conceptual objects of different localization. Consequently, the mapping structures relating external and conceptual objects must include the distribution rules of the external objects.

Functional processors

Once the data dictionary directory is distributed, each DBMS processor can be located using the simplest criterion of setting it on the computer which manages the parameters most frequently required by its executions. Consequently a processing computer is equipped with an external data base schema processor (one for each external model) and a conceptual/external data base transformer (one for each external model). A data computer is endowed with an internal data base schema processor, a conceptual data base schema processor, an internal data base/conceptual transformer and an internal storage/internal data base transformer. Let us point out that the transformation of external objects into conceptual ones includes the distribution of external objects on data, that is to say, the decomposition of global queries and updates into local ones.

Interfaces

The distribution of ANSI/X3/SPARC DBMS interfaces is straightforward. However, two interfaces must be transformed into a protocol.

a. The conceptual data manipulation language (system format) must be exchanged on the network between a pair processing computer—data computer. This includes conceptual objects requests and receipts with associated control. For this purpose, it is proposed to develop a standardized data manipulation protocol. Such a protocol must be derived from a data manipulation language with a high degree of functionality in order to take care of the network slow rate of communication. A good example of such a language is QUEL as used in SDD and in progress in the SIRIUS project.

b. The external data bases schema processor needs have access to the distributed conceptual schema in order to bind the external objects to objects declared in the conceptual schema. It can be performed by consulting the meta-data base which contains on each data computer the conceptual schema. For this purpose, the previous data manipulation protocol can be utilized. It allows documentation on the conceptual schema from a processing computer if a facility is provided to manipulate the meta-data bases which contain the local parts of the conceptual schema.

Finally, only one data manipulation protocol must be added. This protocol is implemented as the first level of the communication kernel which is described in the next section. Figure 4 gives the schematic of the system on a processing computer and Figure 5 gives the schematic on a data computer. Let us point out that Interface 3 must stay accessible through the network for the application system administrator; this interface can be implemented with the data manipulation protocol.

THE COMMUNICATION KERNEL

An overview

The communication kernel performs and controls the communications of data manipulations and resulting entities. It includes the transport management which carries out the transfer of data between endpoints and which corresponds to Levels 1, 2 and 3 of the ANSI/X3/SPARC distributed...
reference model (1, 2, 3 and 4 in ISO 78), see Figure 2) when the endpoints are located on different computers. When they are located on the same computer, the transport management corresponds only to a buffer movement. In addition, the communication kernel includes two layers:

- The session control which controls the correct interaction of processes (transaction and possibly batch processes) with the conceptual data base.

- The data manipulation control which controls requests and receipts of conceptual object occurrences.

The functions of these two levels are specified in the following. Each of them requires a specific protocol between pairs of controllers. The different layers implemented on every computer are summarized in Figure 6. Figure 7 illustrates the different levels of protocols between two sites.

Data manipulation controller functions

The different functions of the data manipulation controller are the following:

- Communication of conceptual data manipulations, i.e. coding/decoding into/from messages and sending/receipt of these messages.
- Communication of status, i.e. coding/decoding into/from messages and sending/receipt of these messages.
- Communication of conceptual object occurrences, i.e. packing/unpacking into/from messages with possibly ciphering/deciphering and sending/receipt of these messages.
- Fatal error control and occasional abortion of conceptual data manipulations.
• Global flow control of the number of objects generated by each conceptual data manipulation.

The data manipulation protocol specifications define the format of messages which contain data manipulations, status and conceptual objects.

Session controller functions

The session controllers coordinate the distributed processing. The main functions of this layer are the following:

• Initiation and termination of processes.
• Start/stop of process steps (transaction commitment unit or job step).
• Journalizing of updates.
• Commitment of updates and step back-up and recoveries.
• Resolution of concurrency conflicts.

The session control protocol specifications define the format of messages requesting initiation and termination of processes, start/stop of steps, commitment of updates, locking and unlocking of objects, collection of locking status for deadlock detection, step back-up and recoveries.

THE UNIFIED ARCHITECTURE

The unified architecture is now straightforward. Two general-purpose computers are represented Figure 8.

The external presentation box summarizes the processors and interfaces represented in Figure 4, except the external application programs and the communication kernel. It includes the external schema processors whose roles are the validation of the external schema declarations, their binding to the conceptual schema and the insertion of the resulting structures in the local part of the data dictionary directory.

It also includes the conceptual/external transformers which receive the user primitives and translate them into standardized conceptual data manipulations. For this purpose, the structure describing the external schemas and their binding to the conceptual one are utilized. Of course, several external models and external data manipulation languages should be offered to the users.

The data base management box summarizes the processors and interfaces represented in Figure 5, except the communication kernel. It includes the conceptual schema processor which interacts with the enterprise administrator for the declaration of the local part of the conceptual schema. After compilation, the structures resulting from this local part of the conceptual schema are stored in the local part of the data dictionary directory. This box also includes the internal/conceptual transformer which receives the conceptual data manipulation from the communication kernel and transform them into internal data manipulations using the structures resulting from the conceptual schema and the mapping structures relating conceptual schema and internal one. Of course, the data base management box also includes all the internal and storage facilities.

Let us point out that there is no direct path to interchange a message between two application programs located on the same computer or on different ones. This can be performed through data bases. However, in order to simplify such a path, objects can be defined in the internal to conceptual mapping description as stored in main memory—only buffering will be performed.

CONCLUSION

The proposed architecture requires the definition of every interchangeable object in the conceptual schema. That is, in the mind of the author, a key-point to ensure the success of distributed systems. Indeed, it is alarming to see in some distributed enterprises the development of distributed applications without control over the application-level communications—each application programmer is allowed to specify his own application protocol.
At the present time, the question of great importance is to control and standardize the high-level communications between computerized workstations. The conceptual description of objects which are modeled and consequently can be interchanged is a necessary tool towards the ultimate goals: "To make every process in the world addressable to one another such that they can exchange information when such exchange appears useful. . . ." But also, to make every set of data accessible to every process when such accesses are useful and authorized.

ACKNOWLEDGMENTS

The proposed unified architecture has been improved by helpful discussions with R. Gomez, M. Jouve, with the SIRIUS project members and with C. W. Bachman.

REFERENCES


