Database Cooperation: Classification and Middleware Tools

PAOLO ATZENI, Università di Roma Tre
LUCA CABIBBO, Università di Roma Tre
GIANSALVATORE MECCA, Università della Basilicata

INTRODUCTION

The recent growth of computer networks, in terms of technology, methodology and actual deployment, has created enormous expectations in many users. The achievement of a smooth interaction of the various components within the organization and the development of a simplified, uniform interface offered to the outside world can be considered major long-term goals of an enterprise-wide network (Kim, 1995). In this perspective, data should circulate easily, without need for replication nor for reentry.

In most cases, networks have appeared late in a picture that already included many application systems, developed independently from one another. As a natural consequence, it is expected that these applications cooperate, possibly within a larger effort to reengineer not only the information system but also the business processes of the enterprise. The need for establishing cooperation among these systems arises also for other motivations. First, it is common now to buy specific applications, for example to manage accounting or personnel data, from specialized vendors; these independent applications have then to be integrated in "the" enterprise information system. Second, companies now often merge or split, and existing systems have to evolve accordingly. Finally, there are systems that have existed for years, such as airline reservation systems, that by their own nature need to interact with information systems from different companies. Specifically, the discussion in this paper is the synthesis of a contribution to the development of applications over a national network being deployed in Italy, to connect all the offices in the civil administration.

It is important to clarify that the interaction among different systems over a network may occur at various levels. The simplest is connectivity, which is realized when systems and networks are linked together in some way and so allowed to exchange information; this is for example the case of any system connected to the Internet using the TCP/IP protocol. A more complex form of interaction is interoperability, in which systems and networks interact by means of standard services. In Internet, we have standard protocols above TCP/IP, such as file transfer (with the file transfer protocol, ftp), virtual terminal (telnet), electronic mail (X.400 or ESMTP/MIME), directory service (X.500), the World Wide Web (http). In this case, the interaction between different systems is limited to standard services offered on top of the connectiv-
ity environment. Finally, the highest form of interaction is cooperation, the only one that gives actual benefit to the final user in a transparent way; in this case, applications over different systems interact with one another, and, at the extreme level, integrated, distributed applications coordinate existing local applications.

The cooperation of information systems requires that the participant systems offer services, and occurs when systems make use of services offered by other systems. A major feature of most cooperative information systems is that their component systems have to serve two different sets of goals: those related to the specific task they have originally been designed for and those shared with the other participants in the cooperation. Component systems are subject to evolution for technical or organizational reasons, and the same could happen to the requirements of the overall cooperative systems. Therefore, it has been observed that a fundamental issue in the study of cooperative information systems is the “management of change” (DeMichelis et al., 1998). It turns out that most of the issues related to this framework also arise when one considers migration of information systems: if a conservative, gradual approach is taken (and this is usually the only reasonable choice (Brodie and Stonebraker, 1995), except for a few extreme cases), then intermediate phases in the migration activities would require cooperation among components. Moreover, since migration is almost a never ending activity, as the steady state is seldom reached, it could often be the case that complex systems need to be cooperative at all times.

Cooperation can be studied in various ways. From a rather standard information-system point of view, we believe that it could be important to distinguish two main forms of cooperation:• data-oriented cooperation, in which data in a system is visible and/or accessible to other systems;• process-oriented cooperation, in which systems offer services, exchange messages (or data, documents) and trigger activities.

Practical systems are usually based on both kinds of cooperation. However, it is very often the case that one of the two aspects — data sharing or process sharing — represents a prevalent requirement for the final system. Moreover, for methodological reasons, we tend to study data-oriented features and process-oriented features in a rather independent way. As a matter of fact, in our study for the Italian public administration, we found the major instances of potential cooperative systems to be placed in one of the two areas. For example, all the systems related to interoffice payments fall in the process-centered cooperation, whereas integrated civil service register or census systems fall in the data-centered cooperation.

In this paper we explicitly consider the issue of data-oriented cooperation, studying features and requirements of applications that need to share data. As an ideal goal, one could think that database cooperation should aim at offering to the user a fully-fledged distributed database: the whole patrimony of relevant data is shown as if it were stored in a unique, possibly distributed database, with a complete transparency with respect to location, ownership, and with continuous access to non-replicated operational data allowed to all authorized users. However, current technology falls short from supporting such a situation or, at best, can support it only at very high costs, especially if performance, availability, and reliability are important. As a consequence, the design of a system for database cooperation requires an in-depth evaluation of costs and benefits.

The main contribution of the paper consists in a set of criteria for classifying data-oriented cooperative applications, which help to identify how close we should go to the ideal “distributed-database goal”. The paper is organized as follows. First, we discuss the different needs from which cooperation may arise, and propose a set of classification criteria that go beyond the usual degrees of heterogeneity, autonomy, and distribution. Based on these criteria, we develop a new classification of data-oriented cooperative systems, which incorporates traditional architectures, such as federated databases or data warehouses. We then briefly discuss how existing middleware tools can be used as a basis for the implementation of the main categories of cooperative systems. Finally, we conclude with a discussion that relates cooperation with reengineering and migration.

**CLASSIFICATION CRITERIA**

Cooperation means that we have multiple systems; data cooperation means that we have multiple databases that handle data. Traditional criteria for classifying cooperative systems, in the context of distributed and federated databases (Sheth and Larson, 1990), refer to the level of: (i) distribution; (ii) heterogeneity; and (iii) autonomy of the component databases. The level of distribution is a measure of how the various resources, and especially data, are spread over the system. Distribution may range from different databases on a same machine to databases spread over a geographic network. Note that, as opposed to usual distributed databases, here distribution is not a design process, but a fact, due to the preexistence of the cooperating databases.

Heterogeneity arises in many different aspects. There can be differences in the computing environments (hardware, operating system, network software). The database management systems involved may differ in the data model (e.g., relational, hierarchical, object-oriented, file-based), in details in the same data model (e.g., incompatible versions of the relational model, with different data types and available constraints), or in the language (e.g., different dialects of SQL). There can be semantic heterogeneity, due to differences in the meaning of data. Autonomy is the absence of a common (and coordinated) control over the various systems. The level of autonomy measures how much the component
Related Content

Managing Data Security in E-Markets through Relationship Driven Access Control
[www.igi-global.com/article/managing-data-security-markets-through/65539?camid=4v1a](www.igi-global.com/article/managing-data-security-markets-through/65539?camid=4v1a)

Cache Management for Web-Powered Databases
Dimitrios Katsaros and Yannis Manolopoulos (2003). *Web-Powered Databases* (pp. 203-244).
[www.igi-global.com/chapter/cache-management-web-powered-databases/31429?camid=4v1a](www.igi-global.com/chapter/cache-management-web-powered-databases/31429?camid=4v1a)

Normalizing Multimedia Databases
[www.igi-global.com/chapter/normalizing-multimedia-databases/11181?camid=4v1a](www.igi-global.com/chapter/normalizing-multimedia-databases/11181?camid=4v1a)

View Materialization in a Data Cube: Optimization Models and Heuristics
[www.igi-global.com/chapter/view-materialization-data-cube/4300?camid=4v1a](www.igi-global.com/chapter/view-materialization-data-cube/4300?camid=4v1a)