Report 78-04
Stanford -- KSL

Distributed Database Coupling.
Hector Garcia-Molina,
Mar 1978

Scientific DataLink

HPP-78-4

March 1978

DISTRIBUTED DATABASE COUPLING

By

Hector Garcia-Molina

Computer Science Department
Stanford University

DISTRIBUTED DATABASE COUPLING

By Hector Garcia-Molina.

ABSTRACT

The concept of distributed database coupling is presented as a criterion for the classification of distributed databases. Coupling provides a handle for studying how data may be distributed. A new type of distributed system, the distributed knowledge base, is suggested for large applications where the data is loosely coupled.

INTRODUCTION.

The term "distributed database" encompasses systems of great diversity. The fact that there is a great variety of distributed databases makes their study and comparison difficult. It is hence important to clearly define what is meant by a given distributed database; it is also important to understand the spectrum of available alternatives. This understanding should elucidate the design of distributed databases and clarify the comparison of different systems.

We will define a distributed database in a general way, so as to include most types of distributed databases:

> A distributed database is a system that allows integrated access to a collection of logically independent databases [Garcia77].

Associated with each database there is a computing facility we will call a node. Notice that two or more of the databases (and their nodes) may be physically located in a single computer. To differentiate a single non-distributed database in a single computer from a set of databases on a single machine, it is important to determine the logical independance of the individual databases. That means that from a logical point of view, the databases could as well be located on separate machines.

By integrated access we mean that a query entered at any computer in the system can access data in any one of the databases. This is a minimum requirement for integrated access. In particular note that being able to update or add data at a remote computer is not a minimal requirement.

CLASSIFICATION OF DISTRIBUTED DATABASES.

Distributed databases have been classified by several different criteria. Each criterion can be thought of as an "orthogonal axis" in the distributed database "space". Some of the more common criteria that have been used for classifying distributed databases are:

1) Number and type of computing facilities. Distributed databases can be categorized by the number and type of nodes. Nodes vary according to their computing power and their storage capacity. For example, some nodes can be back-end data machines [Canaday74] which are special purpose computers dedicated to data management. Other nodes can be multi-functional nodes which perform a variety of

tasks.

2) Logical network organization. In some systems the nodes are organized in a "horizontal" fashion, that is, all the nodes are functionally equivalent. On the other hand, some networks can be "vertically" organized. In this case, some nodes are functionally subordinated to others (e.g. a hierarchy). (Distributed databases could also be classified to the physical network used, but this classification is not as important to us. From the distributed database point of view, the physical connection of the nodes affects only the performance of the system.)

3) Spatial distribution. Distributed databases can also be grouped according to the degree of spatial distribution. It is possible to have a geographically distributed system, but it is also possible to have a complete system in a single room.

4) Data model. Distributed database systems, as well as centralized systems, can be classified by the data model used (e.g. relational, network, hierarchical, etc.) [Wiederhold77]. However, in general distributed systems, it is possible to use more than one data model in a single system. (See next item below.)

5) Homogeneity. Another way to categorize distributed databases is by the uniformity of the data model and data languages used throughout the system. In a homogeneous system, all the nodes will use the same data model and data language. In these systems it will not be necessary to have either data or command translation for inter-node communications. In a heterogeneous system, translation between nodes will be necessary.

6) Complexity of the interactions. Since not every system allows the most general types of interactions, distributed databases can be grouped by the complexity of the interactions permitted. Some common restrictions are to limit the number of nodes a query can span, to limit the number of intermediate steps for query processing, to permit only updates that have been generated at the same node (local updates), etc.

7) Degree of centralized control. The extent of centralized control allowed can drastically change the design of a system. Centralized control mechanisms can include a central system directory, a deadlock prevention mechanism, a synchronization center, or a central locking mechanism. In a centralized control system, one or more special nodes are in charge of the control functions, while in a distributed control system, the control of the system operation is shared by all the nodes. As a general rule, a distributed control system is more complex but more reliable.

8) Amount and type of data redundancy. Data can be duplicated for two reasons: performance and reliability. By having several copies of the data, access time to it can be reduced (performance) and if one of the copies is lost, the data is still accessible (reliability). In some systems, only certain parts of the database will be stored redundantly at some nodes, while in other systems the complete database may be replicated at every node.

9) Amount of dynamic reconfiguration. Distributed database systems can be categorized by the amount of dynamic reconfiguration that is allowed. In a static system, the location of the data and directories, the names and numbers of files (or relations) and the system structure are all fixed at creation time. In a dynamic system, data can migrate and be duplicated, files (or relations) can

be created and destroyed, and nodes can be added to the system, all without interrupting the operation of the system. Clearly, dynamic systems are more complex but more flexible.

An important classification criterion is missing from the above list: "distributed database coupling". Most of the literature has not considered this factor explicitly. The amount of coupling is a powerful abstraction which can be used to summarize database alternatives. We feel that the amount of distributed database coupling is a crucial factor in the design of distributed databases. Loosely coupled systems are drastically different from closely coupled ones, and the degree of coupling will determine the efficiency of the system.

COUPLING.

The term "coupling" has been used previously in the parallel processing literature [Lesser75]. The coupling factor between parallel processes can be intuitively defined as the ratio of the time the processes spend communicating to the time they spend computing. A closely coupled system has more communication than a loosely coupled one; the looser the coupling of the processes the less communications overhead there is.

The concept of coupling in distributed databases is similar to the above concept of coupling. In the following sections we will define coupling explicitly for distributed databases. This definition, although not formal, will be more detailed and will be embedded in a distributed database setting.

Initially, we will only consider coupling between a pair of databases in a distributed database system. The degree of coupling will be a measure of how closely related or tightly knit the two databases are. Databases with loose or weak coupling will have fewer ties and interactions than two databases with close or strong coupling.

There are two kinds of coupling that can be identified: usage coupling and structural coupling. They will be described in the following sections.

USAGE COUPLING.

The degree of usage coupling between two databases depends on the queries or transactions that are given to the system. If a single transaction involves or spans the two databases, then the two databases become usage coupled. As the fraction of the transactions that span both databases increases, the databases become more closely coupled.

In the analysis of usage coupling, it is also important to consider the types of transactions since some transactions produce more coupling than others. For example, consider the following two relations:

```
At database A (at main office):
            EMPLOYEE( NAME, SALARY, MANAGER, ...)
At database B (at branch office):
            EMPLOYEE( NAME, SALARY, MANAGER, ...)
```

The query "Find Smith's salary" produces some coupling because both A and B are involved in the search for Smith's salary. (If the

databases are searched sequentially and "Smith" is found in the first one, then the second database might not even be searched. However, even in this case there is some coupling since the search logically spans the two databases.) On the other hand, the query "Find the names of the employees that make more than their managers" should produce much more coupling since solving this query requires more interactions and data transfers between the databases [Stonebraker75].

As another example, suppose that an item is duplicated in two databases. A transaction that updates the item will produce more inter-database traffic than one that simply reads the item since the update transaction also has to update the duplicate item in the other database.

Therefore, we will say that usage coupling is not only proportional to the number of transactions that span the two databases, but it is also proportional to the number of interactions between the databases that are produced by such transactions. This interaction could be measured by the number of values from one database that are used to manipulate data in or from the second database. Notice that we have not specified that the interaction should be through a direct communication link between the databases. For example, a query at node A may produce coupling between databases B and C if the query solution at A requires the interaction of data obtained from B and C.

The term "locality" is similar but not equivalent to usage coupling. A database exhibits locality of reference if most of the transactions that originate at its node only span that single database. Thus locality is a partial indication of weak usage coupling. A database with a given locality may have different usage couplings with different databases depending on the complexity of the "non-local" transactions.

STRUCTURAL COUPLING.

It is not possible to have usage coupling between two databases without some underlying framework that ties the two databases together. Thus we say that two databases are structurally coupled if there are links or ties between them. These structural links can be classified into three possibly overlapping categories:

1) Consistency links. Consistency links are used to keep two databases in a consistent state without data disagreements or conflicts [Eswaran76]. That is, consistency links exist between two databases if modifying one of the databases requires the modification of the other database. Some consistency links are required for the maintenance of duplicate data: If parts of database A are duplicated in database B, then when duplicate data in A is updated, the corresponding data in B must also be modified to maintain consistency. Consistency links also occur when data values are a summary of other data values in the database. Other consistency links may be due to user consistency constraints. For example, consider a database with "parts" inventory and another database with "orders from suppliers". Assume that there is a requirement that when the number of parts in the inventory reaches a certain low value, then an order for more parts be entered in the second database. In this case, the databases have to be kept "consistent", so ther is structural coupling between them.

2) User links. Two databases are also structurally coupled if one of the databases has user pointers or references to the other

one. In the relational model of data [Wiederhold77], these pointers can be any datum (component) whose value is a relation name, attribute name or the value of another datum (component) in the other database. For example, consider the following relations:

At database A: PART(PART#, SUPPLIER#, COLOR, ...)
At database B: SUPPLIER(SUPPLIER#, ADDRESS, ...).

These two databases are structurally coupled since the values of the SUPPLIER# component in relation PART reference entities in the SUPPLIER relation.

3) Semantic links. There is a third type of link that can structurally relate two databases: a semantic link. The simplest way to describe these links is through an example. Consider database A with data on certain parts p1, p2, p3 and database B with data on part p4. If parts p1, p2 and p3 can be assembled to make part p4, then the databases are coupled by means of the semantic link "part p4 consists of parts p1, p2 and p3". This link may be part of a semantic graph or net [Mylopoulos75] stored in the distributed database system or it may simply be known to the users. In either case the link exists and produces coupling. There are many other types of semantic relationships between data items, all of which produce structural coupling. (E.g., x is a member of y, x is equivalent to y, x supplies part y, etc.)

The amount of structural coupling between two databases is then proportional to the number of consistency, user and semantic links between them.

SYSTEM COUPLING.

Structural and usage coupling are not independent, for structural coupling is required to support usage coupling. Thus, the amount of structural coupling indicates the potential usage coupling and its cost. For example, if there is no structural coupling at all, a user at one node will not be able to produce any interaction whatsoever between the databases. As the structural coupling increases, it will become simpler to enter transactions that produce inter-database interactions.

The total distributed database coupling is proportional to the usage and structural coupling between every pair of databases in the system.

COUPLING AND DISTRIBUTABILITY.

Distributed database coupling is an important factor to be considered during system design time [Wiederhold73]. At this time, distributed database coupling should be reduced as much as possible by adequately distributing the data among the databases. Communication between databases will always be more expensive and time consuming than communication within a single database (often orders of magnitude more expensive). In particular, updating in a distributed database is a very expensive operation if the data is duplicated and database consistency must be maintained. The more closely coupled the system is, the more time consuming the updating will be.

The lower the extent of usage coupling is, the less need there will be for inter-database communications, so a weak usage coupling system is clearly desirable. And since usage coupling is supported by structural coupling, by reducing the underlying structural

coupling, the potential for usage coupling will be reduced. To reduce the coupling, the system data should be partitioned among the databases in such a way as to minimize the number of inter-database links and the inter-database interactions. If the coupling between two databases is strong, it is probably better to fuse the two into a single one, thus eliminating the communications problem.

Distributed database coupling gives us a handle for studying the "distributability" of the data. Data that are closely coupled (i.e., that if placed in two separate databases would produce strong coupling) should not be distributed, while loosely coupled data can be distributed with very low overhead.

Of course, it will not always be possible to reduce coupling as much as one would like. There are other factors that might dictate distribution even though the distributed database coupling is strong. (E.g. reliability, managerial or political factors.) In these cases, the system operation will not be as efficient as possible (from a technical point of view).

The problem of reducing database coupling is difficult and has only been addressed in a limited way. Most of the work has only addressed the problem of reducing usage coupling in distributed file systems [Casey72, Chu69].

DISTRIBUTED KNOWLEDGE BASES.

As was mentioned earlier, distributed database coupling provides an "axis" in the distributed database space where systems can be classified. At one end of the spectrum is the extremely strongly coupled distributed database model which would be best implemented as a centralized database. At the opposite end is the very loosely coupled distributed database which is well suited for distribution. Distributed databases are currently being designed at several points along the axis. However, there seems to be a gap in the very loose coupling end, precisely where distribution is most appropriate. This is why we will propose a very loosely coupled system which has significant potential in large database applications. A conceptual model of such a distributed database follows.

Each database in the system will be an expert in a certain area of knowledge. The databases will not only contain actual values, but will also include inference rules and procedures associated with the expert's knowledge. We will call such a system a distributed knowledge base to emphasize this fact. The database experts interact much like human experts would in solving a problem. If a transaction deals only with knowledge from one area, it should be solvable by one expert only. If the solution requires knowledge from several experts, then the original transaction is decomposed into transactions for other experts. These transactions may in turn be decomposed until they are answerable by one expert only [Lenat75].

To have a very loosely coupled system, it is important that each expert database have relatively few structural links to the other experts. For example, if the system contains data on employee salaries, they should all be in one database, and that database will be the expert on salaries. Of course, if the salary data is such that it can be split into two databases (e.g. salaries at site A and salaries at site B) with very weak coupling, then the system may have two experts on salaries.

Only local updates will be allowed in a distributed knowledge base. By a local update we mean an update that originates at the node

associated with the database being updated. This is a reasonable restriction since if a database is to be an expert, only it should be allowed to modify its knowledge. Of course, we will permit another expert to "suggest" a change in the database, but the local expert will be free to accept or ignore this change.

Crucial data will not be duplicated in the system because this would increase coupling. By crucial data we mean data whose global correctness and consistency is essential to one or more users. For example, if the data are employee salaries, duplicate expert databases will not be allowed. Just like in real life, it is hard to make experts agree on something and if that something is of great importance, it is preferable to have only one expert. If reliability is also important, then what looks like a single expert database to the distributed knowledge base may actually be a tightly coupled distributed database in itself. If the data or knowledge is not crucial, then duplicate experts will be allowed. However, these experts may have conflicting knowledge and it will be up to the users who to consult or believe. Users that discover such conflicts may inform the involved databases and then the databases, if possible, could settle the discrepancy among themselves.

As an example of a distributed knowledge base consider a university with several computers. One computer and its database is the expert on accounting (student payments, faculty salaries, alumni gift records, etc.). Another machine contains the registrar expert database (student courses and grades, classroom assignment to courses, etc.), while a third computer contains the library database (card catalogue, book checkout records, etc.). Another computer for use by the students has data on the usage of the machine and the progress of the students in computer programing classes. Additionally, there is a computer at each academic department which has data on the student and faculty's status, the requirements for a degree in that department, etc.

Since there are few structural links between the databases, the usage coupling will also be weak. Therefore, we expect that most transactions in this system will involve one area of knowledge only and will thus require the intervention of only one expert. The transactions that involve several experts produce little interaction between databases since the inter-database communications are at a "question and answer" level. For example, the transaction "Can student Smith receive his degree?" will be given to the registrar expert. The transaction will be analyzed and the following queries will be made to other experts:

To library expert:
"Does Smith have any books checked out?",
To accounting expert:
"Has Smith payed all his bills?",
To registrar expert (itself):
"Has Smith taken all required courses?".

Answering the last question might involve further consultation with Smith's department database expert to find out what the required courses are.

In this example, it is important that there are no conflicts regarding data on student payments, so there is only one copy of it (in the accounting database). On the other hand, student grades are not crucial (as defined above), so a temporary duplicate may be kept in each department's database, while the permanent record is in the registrar database. Updating is only done on local data. Thus, to

change a students grade from the department's computer, first the local copy is updated and then a message (with the proper authorization) is sent to the registrar database requesting the change. This mechanism might lead to inconsistencies in student grades. Fortunately, these inconsistencies will arise only in very rare circumstances. For example, at the same time two different departments may request different changes to a same grade. In this case, one of the databases will end up with an incorrect copy of the grade. This is certainly an odd case since one of the two departments has no business changing the grade. Since there is only one expert on student's payments and all updates are performed locally, there is no possibility for inconsistencies in this area of knowledge.

The elimination of duplicate data, the requirement for local updates and the weak usage coupling make the operation of a distributed knowledge base simple and efficient. We believe that these types of systems have potential for very large databases that can be naturally distributed. Databases in universities, large hospitals, military applications, govermental planning, or large enterprises are good candidates for distributed knowledge bases.

CONCLUSION.

Distributed database coupling has been introduced as a criteria for the study of distributed database systems. Coupling may aid the study of data "distributability" and leads to the concept of a very loosely coupled distributed knowledge base.

ACKNOWLEDGMENT.

Many of the ideas presented here originated from discussions with Ramez ElMasri, Kjell Knutsen, Reid Smith, John Shoch and Gio Wiederhold.

REFERENCES.

[Canaday74] R. H. Canaday, R. D. Harrison, E. L. Ivie, J. L. Ryder and L. A. Wehr, "A Back-end Computer for Database Management", CACM, Vol. 17, Num. 10, October 1974, pp.575-582.

[Casey72] R. G. Casey, "Allocation of Copies of a File in an Information Network", Proc. Spring Joint Computer Conference, 1972, pp.617-625.

[Chu69] W. W. Chu, "Optimal File Allocation in a Multiple Computer System", IEEE Trans. Computers, Vol. 18, Num. 10, October 1969, pp. 885-889.

[Eswaran76] K. P. Eswaran, J. N. Gray, R.A. Lorie and I. L. Traiger, "On the Notions of Consistency and Predicate Locks in a Database System", CACM, Vol. 19, Num. 11, November 1976, pp.624-633.

[Garcia77] H. Garcia-Molina, "Overview and Bibliography of Distributed Databases", Report HPP-77-27, Computer Science Dept., Stanford University, August 1977.

[Lenat75] D. B. Lenat, "Beings: Knowledge as Interacting Experts", IJCAI Proceedings, Tbilisi, USSR, September 1975, pp.126-133.

[Lesser75] V. R. Lesser, "Parallel Processing in Speech Understanding Systems: A Survey of Design Problems", Speech

Recognition, Edited by D. Raj Reddy, Academic Press 1975, pp. 481-499.

[Mylopoulos75] J. Mylopoulos and N. Roussopoulos, "Using Semantic Networks for Data Base Management", Proc. International Conf. on Very Large Data Bases, Farmingham, Ma., September 1975, pp. 144-172.

[Stonebraker76] M. Stonebraker and E. Neuhold, "A Distributed Data Base Version of Ingres", Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, May 1977, pp. 19-36.

[Wiederhold72] G. Wiederhold, "An Approach to Determining the Suitability of Computing Alternatives", 2nd Communications Conf. at California State University, San Jose, January 1973.

[Wiederhold77] G. Wiederhold, "Database Design", McGraw-Hill Book Co., 1977.