CREATING A CONCEPTUAL MODEL OF A DATA DICTIONARY FOR DISTRIBUTED DATA BASES

By Fabio A. Schreiber & G. Martella

Distributed Data bases (DDB), which have been for many years the object of research and experiment, are now reaching a more precise role in the world of informatics. Among the aims of a DDB system the following seem to have a major relevance:

- To put data closer to their users.
- To avoid too much resources centralization which could cause a worsening of system efficiency and efficacy.
- To enhance system reliability.
- To ease data base maintenance and restructuring operations while maintaining a high availability rate.
- To tailor the EDP structure to the functional and organizational structure of the organization.
- To increase the utilization factor of EDP systems.

Reaching these objectives entails the solution of a number of technical problems. Among them we mention:

- An integrated DDB access capability.
- The transparency to the user of the physical data allocation.
- The logical, physical and "distributive" data independence.
- The software portability.
- An efficient cataloging system.

To most of these problems, solutions have been already given for centralized DB, such as the data independence problem[1]. However, in the DDB environment the solution to this problem, as well as to others, can be much more difficult and, even if many proposals have been made by different researchers[2], in many cases it has not yet been attained satisfactorily. Data independence, in fact, must be assured not only at a local level, but also with respect to a possible data reallocation among the nodes of the information network.

The authors are on the faculty of the Istituto de Elettrotecnica ed Elettronica in Milan, Italy.

In distributed systems, however, we also find some problems which, even if already present in the centralized environment, assume a particular relevance since they are critical to the development and use of the DDB. In particular:

- Prevention of nonplanned data redundancy and of possible data inconsistencies when developing new applications.
- Statement of standards for data definition and utilization.
- Efficient query management.
- Efficient use of the telecommunication resources (computer networks).

Solutions have been proposed for particular problems. As an example, a good documentation technique for data and programs can afford a development time reduction for new applications[3]; data redundancy can be avoided by suitable cataloging techniques[4]. However, no integrated tool has been proposed for the design and the management of a DDB. Such a tool, hereafter called Distributed System Dictionary (DSD), should consist of a set of automated procedures for performing a set of functions, and in data relevant to them.

These functions can be either part of a Distributed Data Base Management System (DDBMS), or part of a Network Operating System (NOS), or should be implemented ad hoc. We suppose that the DDB is the aggregation of several autonomous local DB each having its DBMS. This hypothesis has been made for giving a reference structure to the examples, but it does not limit in any respect the general validity of the exoposition.

As in centralized systems, a DSD is called upon to perform its functions at different moments of the life of an information system. These phases can be listed as follows: (a) design and implementation of the system, (b) updating and maintenance of the system, (c) design and implementation of applications, (d) updating and maintenance of applications, (e) run-time operation of the system and (f) run-time operation of the applications.

While phases a to d are peculiar functions which are often implemented ad hoc, phases e and f are related to functions which are usually embedded in other components of the system's software, such as the DBMS and the OS. However, the data necessary to these functions can mostly be stored in the DDBD and maintained by the DDMS.

Therefore many users are interested in the DSD: the information system designers, the information system administrators and the DDB users, meaning as "users" all the access requests coming from on-line transactions expressed in a conversational language as well as from application programs written in a self-contained or in a host programming language.

Figure 1 shows the different functions of the DSD with respect to each of the above-mentioned entities. This partitioning, however, must be considered only as a conceptual one, since practically all of the functions are heavily interdependent. In the following, the different functions of the DSD will be examined.

After a syntactic check at the high-level DMS, a complex transaction must be decomposed into a set of simpler, one-parameter, transactions. This operation has already been considered for centralized systems and does not bother with data distribution[5]. However, the logical decomposition can be made in different ways, and the selection of the best decomposition technique can be made considering that the subsequent operations must be performed into a distributed environment.

Therefore an evaluation must be made as to how to distribute the simple operations on the nodes of the computer network, and, if redundant copies of some file exist in the system, an evaluation must also be made of the transmission parameters (cost, delay and so on) to get the optimal solution[6,7]. The DSD therefore helps the designer by providing all of the information about the network relevant features and about the DDB parameters.

When a suitable set of execution nodes has been selected for the processing of the transaction, a call must possibly be issued to the translation function before starting the actual processing.

Usually a DDBMS offers the user two kinds of languages: a conversational language (for queries from online terminals), and a high-level procedural language (for application programs acceding to the DDB).

The translation function consists in precompiling or interpreting the user's requests, in such a way as to obtain an instruction set in a language internal to the DDBMS, including calls to modules which define the access procedures. In this phase the DSD will

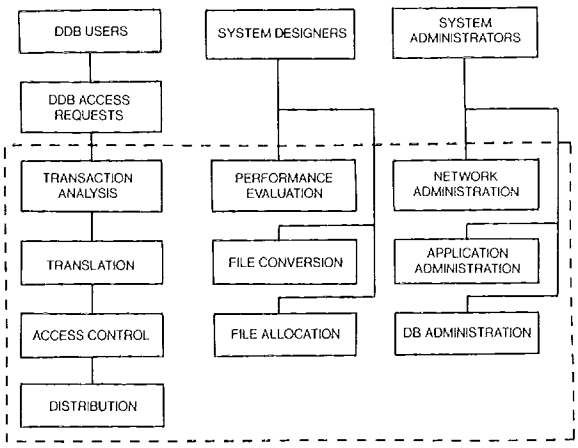

Figure 1—Functions of the DSD

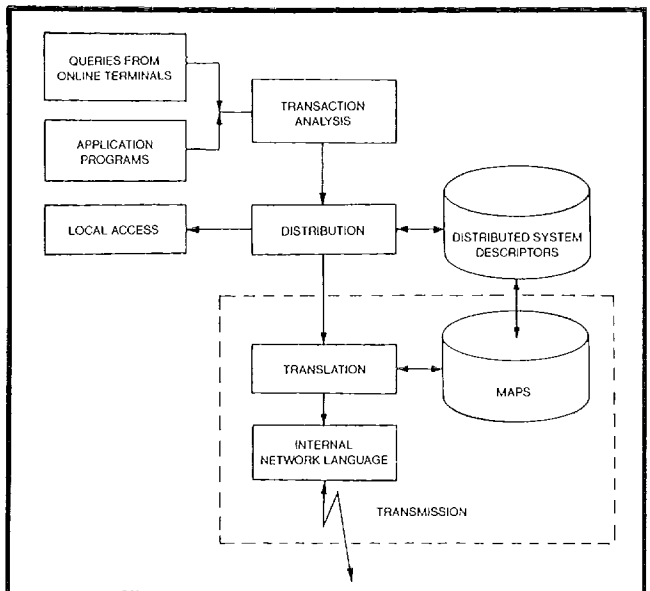

Figure 2—Translation function by DSD

make available maps allowing the translation of complex instructions in the high-level language into simple commands. These maps depend on the physical distribution of data, on the local files structure, on the local OS and DBMS and on the network software (Figure 2). We can see that the translation function entails the passage through different levels of the DDBMS architecture[8].

The main problem encountered in translating transactions into an internal language consists in choosing the target language itself. In fact this decision has influence on both the number of required translators and on that of the translation operations[9]. Moreover, the moment to perform translation influences the number of distribution modules and the number of translations which are required.

It will be the task of the DSD to suggest to the designer, possibly by means of built-in CAD programs, the solution which minimizes both the number of translator and distribution modules and that of translation operations on the basis of the "state" of the DDB and of the peculiar transaction.

The access control function has to assure the consistency of a DDB. This means to check for formal correctness of DDB data (integrity), to control concurrent access from processes to the DDB (locking), to grant access permission only to authorized users (privacy) and to protect the DDB from malfunctions or failures (recovery).

In Figure 3 the procedures and the information needed by these functions are shown. We see that for the integrity assurance the DSD must supply the tables of the constraints. The set of constraints, which are defined at the conceptual level in the DDB, can be either the union set of the constraints for the local DBs or a new defined set at the network level[10]. Moreover, each application programmer can add other constraints at the external schema level[11]. Constraints can be a static type (aimed at the control of the present state of data) or a dynamic type (aimed at the control of a semantically correct evolution from a state to the next one).

A typical problem in a DDB is often encountered when the access request is made at a node different from that where data are stored. In this case the DSD must choose where to do the control, possibly by means of an "optimizer" looking for the solution entailing the smallest network traffic.

A possible solution, for example, performs control before trans-

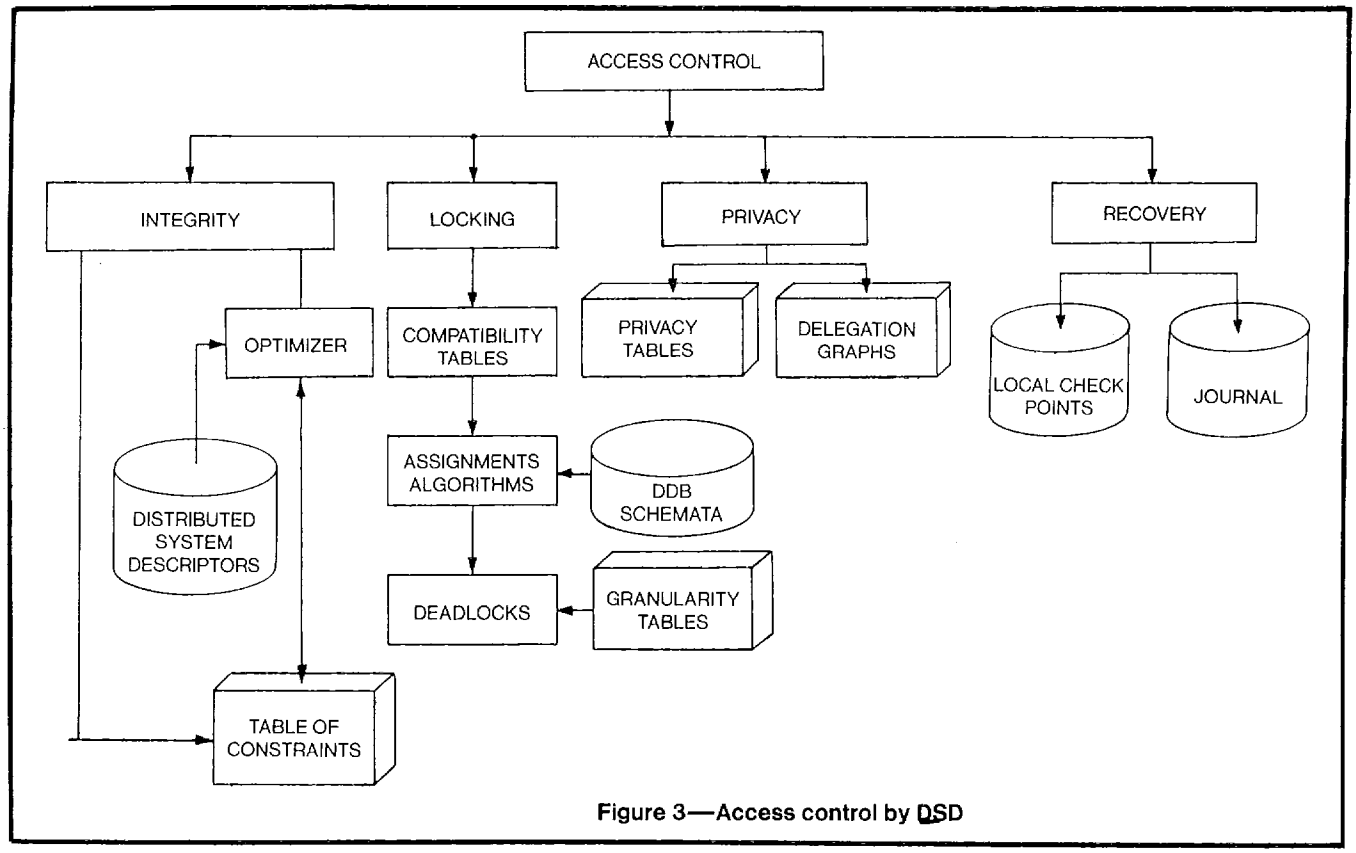

Figure 3—Access control by DSD

mission, that is at the access node if the request is for an insertion operation and at the storage node in the other cases. If data are stored at several different nodes, the set of constraints is to be examined at all of them and the existence of the DSD can greatly ease the operation.

As to locking (Figure 3), the DSD has to manage the compatibility tables for the resources assignment algorithms. Compatibility tables specify if a process P_1 can be assigned to a resource R_1 even if R_1 has already been assigned to another process P_2 in a given mode. An example of such a table can be found in[12]. It must be noticed that, in a distributed environment, different compatibility tables can coexist reflecting different architectures of local DBMS in the network.

The DSD must manage the tables on the basis of the origin and target nodes of the transactions. Moreover, the DSD has to find a proper solution to the problem of the locking granularity in accord with the peculiarities of each local DBMS. This solution will define the assignment of the system's resource to the different requests.

The assignment algorithm utilizes information coming from the DDB schema and information automatically attached to the request by the DSD, at the area opening time, concerning the opened area name, the opening mode, the requests order number and its origin. These data are to be used by the remote assignment routine to reply to the requesting computer. The answers (area correctly opened, or area not opened, and why) will be analyzed by local assignment routines which will optimize the request queues management[13].

For example, the knowledge of DDB schemata (for example, the relations hierarchy in a relational system) allows the DSD to prevent deadlocks[14]. It must use two tables, the first, T_1, containing the network global resources in a predefined order, the other, T_2, recording the resources needed to satisfy each request. The latter is ordered following the defined hierarchy, and the DSD has the job of locking all the hierarchycally lower resources which have not yet been locked when considering each resource request.

As to the control of data access, the DSD has to manage the privacy tables and relations which must have been defined when building the DDB. Privacy tables and relations can represent different control philosophies, since they depend on the local DBMSs. Usually tables identify several classes (subjects, objects and so on), which can be differently defined[15]. Relations, for instance, could be tuples (s, o, t, p, e) where "s" is the subject, "o" is the object, "t" the access mode which is granted if predicate "p" is true, and "e" is the action to be taken if "p" is false[16]. This relational approach makes DSD's functions easier since it allows considering the access laws as system data, avoiding their extraction at table reading time.

The DSD also has the task of giving data to the authorization mechanism for user access. These data can be either in the form of many-to-many relations or of tables relating the different classes into which data are subdivided to the local data responsibilities. Since one of the main actions to be taken is the delegation or the revocation of the authorization to new responsibilities for the control of the classes or of their subsets, the task of the DSD is to manage the time sequence of delegations and revocations, for example by means of "delegation graphs[17]."

As to the recovery restart function, the DSD has to manage information about DDB checkpoints, forward and backward images and several levels of journals to follow, as well as operations performed by each process and those performed on each local DB, to be able to recover the DDB whenever a failure has occurred.

The distribution function has to optimize the distribution of the elementary requests, following predefined criteria. This means, for instance, to separate local requests from those requiring remote data, to choose the execution order of the elementary requests, to choose the optimum routing, to optimize some merit factors such

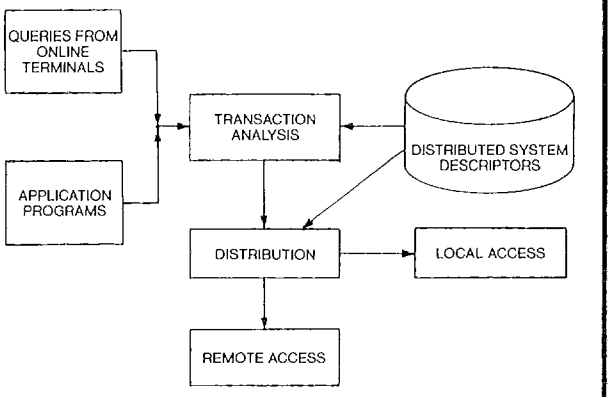

Figure 4—Distribution function by DSD

as access times, processing times, network traffic and so on. It is clear that this function is tightly bound to the transaction decomposition function. The latter needs information about data (such as locality, accessibility, availability, files cardinality, statistical properties and so on) and system resources; the former needs information about the network "response" to the different decomposition hypothesis. The task of the DSD is to coordinate these functions, (Figure 4) and to give them all the required data.

The DSD is also a tool for easing the design of the system. In fact by means of DSD, the designer will be able to evaluate the system performance corresponding to the different possible choices of hardware/software components and of different architectures. For instance, the DSD can be used as a support tool in choosing a local DBMS, to simulate different access methods for selecting those with the highest performance and in choosing report updating and generating procedures. Moreover, the DSD can ease the conversion of conventional files into a data base system. In this case a reallocation of files can be convenient and the DSD can provide all the data needed by the assignment algorithms. If multiple copies of files exist in the system and their updating is made by means of broadcasted messages, then the best routing toward each copy has to be chosen on the basis of data stored in the DSD[18].

In a DDB, different administration functions can be identified. In the case of aggregation of several autonomous DB we find the following functions (Figure 5):

1. Network administration. It is the function which has the task of creating and maintaining the network conceptual schema, the aim of which is to give an integrated view of data in the network's different computers.
2. Network global applications administration. It is the function which has the task of creating and maintaining the external schemata for those applications which actually use the facilities of the DDB, requesting data from different locations.
3. Local application administration. It is coincident with standard application administration of a centralized DB.
4. Local DB administration. It is coincident with DBA function of a centralized DB.

Relevant to functions 1 and 2 are the new applications identification, security and consistency control of the DDB, authorization levels definition for data access, data allocation among local DB following efficiency criteria, identification of the network technical features and of network software availability.

Relevant to functions 3 and 4, besides all the functions concerning the local operation of the DB, is the creation and maintenance

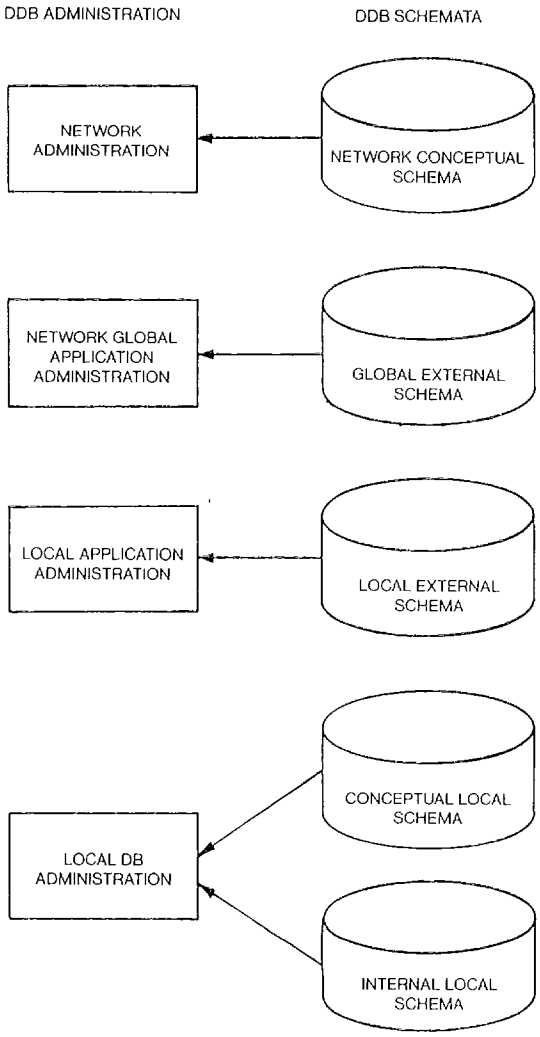

Figure 5—Administration functions in DDB

of the interface between the local and the network DB schemata. The existence of a DSD can greatly improve and facilitate these activities. In fact, as we saw earlier, the DSD manages all the information related to existing applications, security constraints, data allocation, distributed system descriptors and so on.

All the examined functions require a relevant amount of data. These data have to be used at different times and in different ways. Moreover, they must be structured in such a way as to allow their efficient sharing among the different users. Therefore they constitute a real data base supporting all the activities. A possible structure for this data base, henceforth called data dictionary (DD), is presented later.

As we saw, the DDBD must supply system procedures and system users (data base administrators, network administrator, application administrators) with information about stored data. The information can be classified as information about data (a) location, (b) structure, (c) availability and (d) usage.

All of the information is managed by procedures which are part of the DDBD. It constitutes the information base of the DDBD as

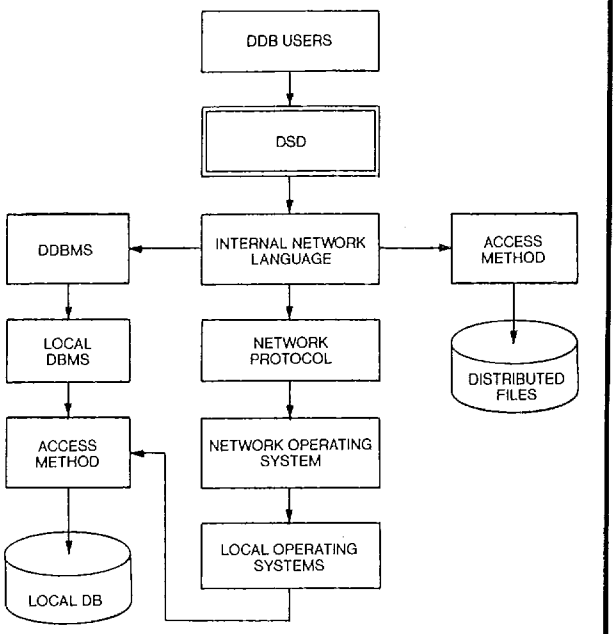

Figure 6—DSD as network interface

well as the functions it must perform for an efficient "external" management of the DDB.

Very important are the interfacing functions the DSD performs among the network, users, designers and system programmers. These functions are automatic generation of macros to interface different operating systems, automatic call to DML translators, automatic generation of macros for the different access methods by means of a catalog of nodes, names and of addressing methods of local data and interfacing by means of correspondence maps possible different transmission protocols.

Figure 6 shows a possible use of the DSD as a network interface. User requests are translated into simple queries by means of the DSD in connection with the DDBMS. Then DSD coordinates distribution and control on execution of requests in the network in connection with the network operating system.

The complexity of management functions in a system with a DDB and the characteristics of data relevant to them have been introduced in the preceding sections. The aim now is to describe a logical structure of the DD, which is shown in Figure 7.

The DD has been partitioned into different views, each characterized either by information contained in it, by its aggregation state, by its differentiated use or by users it is relevant to. In the proposed structure the following levels can be identified: network, external, conceptual and internal. Each of these levels is in correspondence with one or more dictionaries, in particular:

- One network dictionary.
- One global external dictionary and as many local external dictionaries as the network nodes.
- One global conceptual dictionary and as many local conceptual dictionaries as the network nodes.
- As many internal dictionaries as the network nodes.

The physical storage structure and storage devices involved with the DD differ following the type of data (dynamic or static), its use (in real time or offline) and access frequency and so on.

The network dictionary is the nucleus around which all the management functions of the DDB are centered. It contains information to start every management process of the DDB, in particular:

a. Information for the DDB design:
   —Files access programs.
   —Total volumes of queries for each file.
   —Total volumes of updates for each file.
   From this information, for instance, one could evaluate the optimal number of redundant copies.
b. Information for the distribution function:
   —Number and types of transmission links, their unit costs, their mean utilization factor.
   —Routing tables.
   —CPU workloads.
   —Disk occupations.
   This information, together with that listed at a, may, for instance, help determine the optimal allocation of redundant file copies and of possible operation parallelism.
c. General information about data sharing (whether it belongs to a single application or is shared among different users), how many copies of it exist in the network and their location, whether an application requires data involving either one or more DB.
d. Information about data availability such as the existence of privacy constraints and possible constraints due to system components failure.

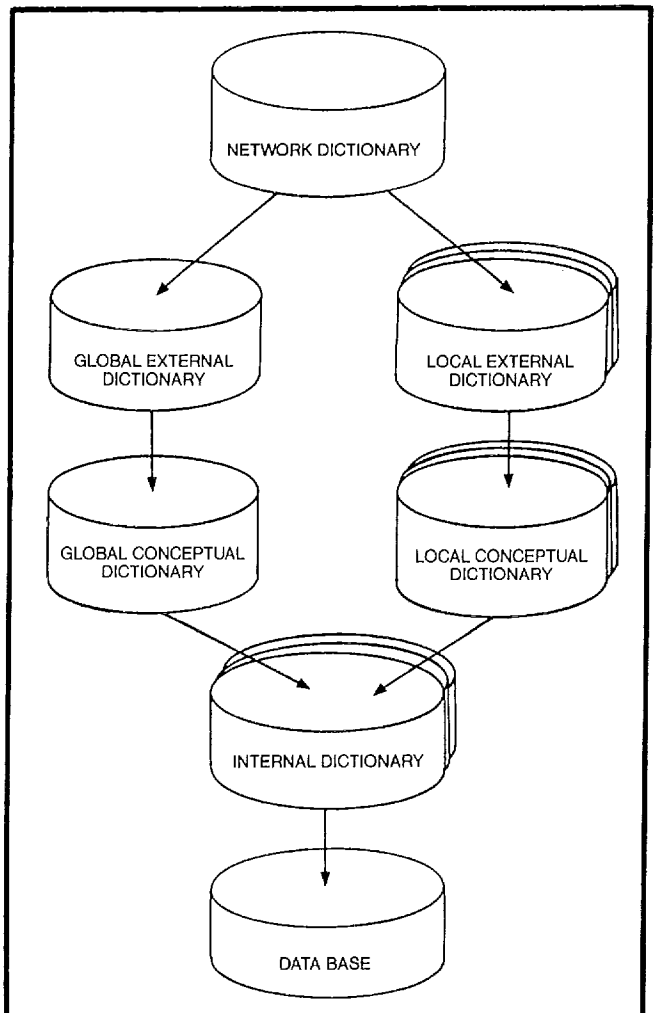

Figure 7—Logical architecture of DD

This information is therefore of aggregated or statistical type. It has the task either of preparing all the relevant specific information for the subsequent processing steps or of giving all the necessary elements for the design and administration of the DDB. For instance, it may be useful in finding the allocation for redundant data, with respect to the query/update rate and to the consistency properties.

e. Information about data transportability.

The network dictionary, besides being used by the distribution algorithm and module and by the translator, can be used by the network administration function, particularly for those actions related to data distribution inside the DDB.

From the network dictionary one goes to the global external dictionary or to a local external dictionary respectively if the required data belong to several different DB or to a single DB.

The global external dictionary consists of information related to data which are used by global applications, such as those applications which work on data belonging to different local DB, and to data which are redundantly stored. As an example, data about component parts of a machine can be stored at different warehouses or production plants. However, assembly information has to be obtained at the assembly site and administrative information that has to be processed by the application programs in the administration computer. Therefore, in order to design and run the application or retrieval programs, the following information has to be stored in the global external dictionary:

a. Data structures.
b. Data location.
c. Data availability (owing to integrity, consistency and so on).
d. Data accessibility (owing to security, compatibility and so on).
e. Data translation maps, access paths.

Information in a is constituted, in the case of a relational model, by the names and formats of the relation's domains, the name of the key domains, statistical quantitative semantical data about domain's values and so on.

Information in b is constituted by location tables containing relations names, their storage site, partition criteria, cardinalities and so on.

Information in c is related to both the constraints tables and dynamic authorization tables (locking prevention). In the case that a very tight consistency of different global copies of data is not required, information about last update time has to be provided, too.

Information in d concerns DBMSs, system software and hardware characteristics of the local computing systems involved in the data distribution.

The global external dictionary, therefore, has to supply information which, together with that found in the network dictionary, can determine which local DB and DBMS are interested in the transaction and how the last has to be formulated to obtain the most efficient result.

The global conceptual dictionary contains mainly information about data entities, common procedures, events and their interrelations. For instance, entities will be described in terms of their attributes and relationships, of the access and identification keys and so on in a terminology which avoids as much as possible any reference to the data model actually used in the conceptual schema of the DDBMS.

Moreover, since this information is dynamically evolving, the global conceptual dictionary must contain details on the different versions which are recognized as valid in different environments. The dictionary eventually has to contain information about data which possibly are not part of a structured DB, but which exist as conventional files structures.

Local external dictionaries consist of the information related to data which are used by the local applications, such as data stored at a single network node. This information concerns data structures, availability and accessibility. The first is the same as that referred to in the global external dictionary, but concerns only the data of a single node. Information about the second and third are related to the local privacy and security constraints and to the integrity requirements imposed by the data shareability. Local external dictionaries are also used by local application administrators in designing and maintaining local applications.

The local conceptual dictionaries contain mainly information about local data entities, local procedures and their interrelations. Information about the first describes entities in terms of their attributes and relationship as in the conceptual schema of the DB it refers to, but prescinding from the particular data model used in the DB schema. Information about the second describes the elementary procedures related to local data which can be used to build different local applications.

Internal dictionaries contain mainly information about physical storage structures of local data, access methods and possibly access paths, physical storage devices and redundancy of elementary data items.

The many different data in the DD, the different functions they are used for and the different user requirements make the use of an automatic management tool for the DD necessary. This tool is constituted by a set of languages and of programs which are integrated in a Data Dictionary Management System (DDMS), and it is useful for creating, updating and maintaining the DD itself. The DDMS has to cooperate with the network operating system and the Distributed Data Base Management System, as shown in Figure 8.

This cooperation, besides what has been discussed earlier, consists of a continuous exchange of dynamic data concerning

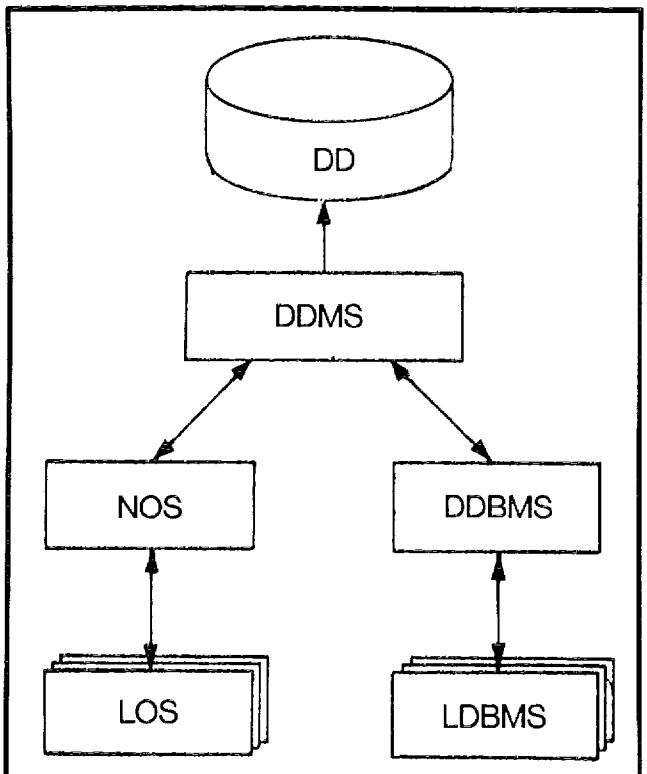

Figure 8. DDMS Toward NOS and DDBMS

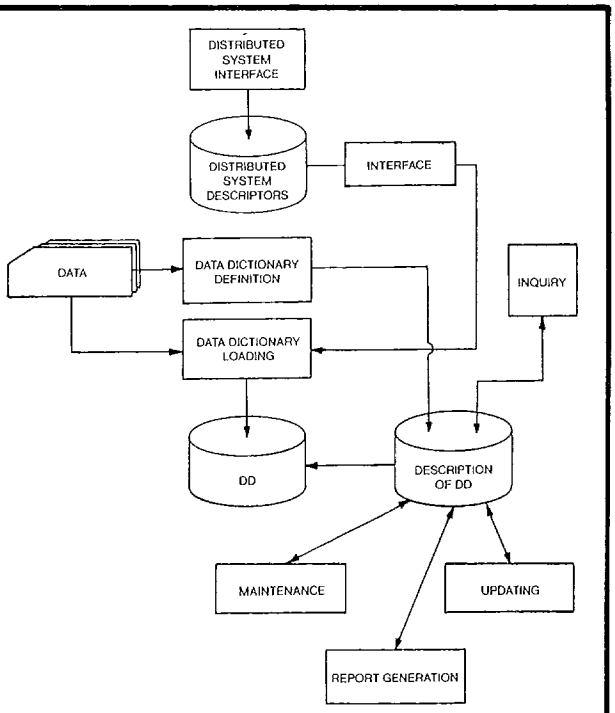

Figure 9. DDMS functions

the instantaneous state of the network and of the data base. In particular the network operating system has to supply data concerning the network traffic, the occupancy level of each node, the features and the availability of the local hardware and software and so on. The DDBMS has to supply data about the network schema, about local external and conceptual schemata, about local data dictionaries, about access frequencies and so on.

These data, which have been called "system descriptors," are supplied by means of an interface (Figure 9) which has the task of extracting, analyzing and filtering them and of putting them in a format directly loadable into the DD. The other data constituting the DD are loaded and managed by means of other programs shown in Figure 9.

The definition procedures allow specification of each data item, the level of the involved dictionary, the data type, its features and so on by means of a simple definition language.

The loading procedures accept the data input and store data in DD files.

The inquiry procedures interpret user queries expressed in a user query language. The user asks values of one or more data items, often selecting them by logical operations and specifying arithmetical and statistical processing.

The maintenance procedures allow data merging, sorting, restructuring, linking and so on.

The updating procedures allow modifications of already loaded data, taking into account the side effects that these changes have on the remaining part of the DD.

The composition and production procedures are essentially a report generator; they allow the specification of the data to be printed and their formats.

Summary

The many problems encountered in designing, maintaining and operating distributed computing systems need, for their solution, a large number of support data which are common to many of them. These data must be organized and maintained like a true data base. Therefore, we have tried to identify data needed by the different functions of the distributed system and to outline a logical architecture for the Distributed Data Dictionary itself and for the system to manage it.

References

1. ANSI/X3/SPARC, Study Group on Data Base Management Systems, Interim Report, 1977.
2. M. Adiba et al., "Issues in Distributed Data Base Management Systems: a Technical Overview," in H. Weber, A.I. Wasserman (Ed.)—Issues in "Data Base Management," Proceedings of Very Large Data Base IV Conference, North Holland, 1979.
3. D. Teichroew, E.A. Hershey, "PSL/PSA: A Computer Analysis Technique for Structured Documentation and Analysis of Information Processing System," IEEE Transactions on Software Engineering, January 1977.
4. E.J. Neuhold, H. Biller "Distributed Data Bases on a Network of Minicomputer," Journées "Bases de Données Réparties," Paris, March 1977.
5. E. Wong, K. Youssefi "Decomposition—A Strategy for Query Processing," ACM-TODS, Vol. 1, No. 3, September 76.
6. A.R. Hevner, S. Bing Yao, "Query Processing on a Distributed Data Base," 3rd Berkeley Workshop on Distributed Data Management, August 1979.
7. G. Pelagatti, F.A. Schreiber, "A Model of an Access Strategy in a Distributed Data Base," in G. Bracchi; G.M. Nijssen (Ed.) Data Base Architecture—IFIP-TC2 Working Conference, North-Holland 1979.
8. C. Baldissera, S. Ceri, F.A. Schreiber, "Basi di dati distribuite"—to be published on Rivista di Informatica, Vol. IX, No. 3, September 1979.
9. G. Bracchi, C. Baldissera, S. Ceri, "Query Processing Strategies for Distributed Data Base Systems," EEC-CREST Advanced Course on Distributed Data Bases, Sheffield City Polytechnic, July 1979.
10. G. Gardarin, S. Spaccapietra, "Concurrency in Distributed Data Base," IFIP W.G. 2.6 Meeting, Brussels, 1976.
11. K.P. Eswaran, D.D. Chamberlin, "Functional Specification of a Subsystem for Data Base Integrity," Proceedings of International Conference on Very Large Data Bases, ACM, New York, 1975.
12. C. Parent "Integrity in Distributed Data Bases," Proceedings of AICA 77, 1977.
13. J.N. Gray, R.A. Lorie, G.R. Putzolu, "Granularity of Locks in a Large Shared Data Base," Proceedings of Very Large Data Base, Boston, 1975.
14. G.L. Everest, "Concurrent Update Control and Data Base Integrity," Data Base Management, J.W. Klimbie and K.L. Koffeman (Eds.), IFIP 1974.
15. G. Martella, G. Nannini, "Security Techniques in Information Systems: An Overview," Proceedings of Convention Informatique, SICOP, Paris, 1975.
16. M.W. Blasgen et al., "System R: A Relational Data Base Management System," IBM Research Laboratory, San Jose, California, 1976.
17. P.P. Griffiths, B.W. Wade, "An Authorization Mechanism for a Relational Data Base System," IBM Research Laboratory, San Jose, California, 1976.
18. F.A. Schreiber, "Problemi posti dal progetto di Data Base distribuiti: i problemi di allocazione," Rivista di Informatica, Vol. V, No. 2, 1974.