THE DISTRIBUTION OF DATA AND PROCESSES IN COMPUTER NETWORKS

E.J. Neuhold

Rapporteur : Dr. R.B. Gimson

Abstract

Distributed data base management systems have attracted considerable attention of researchers in the last one or two years. Commerical interest also has been developing rapidly but as of this writing (June 1978) no such system is commercially available.

Using POREL, a distributed data base system which is currently under development at the University of Stuttgart, as an illustrative example different aspects of distributed data base management systems are investigated and possible solutions to distribution related problems are discussed.

1. Introduction

Only a very few years ago data communication networks were simply seen as vehicles for remote batch processing or interactive access to centralised large hardware and software packets, as for example airline reservation systems.

In the last two years large research and development efforts have been oriented toward building distributed processing and distributed data manipulation networks. The advent of minicomputers and the multiple installations found of these systems in large companies have given a tremendous impetus to such developments. The limitations which are introduced by the relatively restricted speed and storage capacity of each minicomputer system can be relieved by interconnecting the processors and by using load balancing and data distribution strategies to utilise fully the capacity of each system in the network.

There has been much talk that distributing a computing facility between many different locations will reintroduce the inconsistencies, of data and data handling procedures inside an organisation, which were a large problem in the earlier days of computing and which have just been eliminated by centralising the computing facilities and integrating the data into data base management systems. Unfortunately this distribution already has happened and is still happening completely independent of computer networks. Minicomputers and microcomputers originally were only used for very limited and special purpose tasks but they have become so powerful in a very short time period that more and more general purpose work which should probably be organised in a coordinated manner throughout a company is being done locally on such machines. The reason usually is to avoid the red tape of the computer centre or to have "rough data" under local control so nobody else can look at them. The perennial distrust in privacy and data security mechanisms of course still persists and one does not feel comfortable if these "personal" data are floating around in some

large, distant computer centre where all "those people" are running around and physically handling disk or tapes containing ones data. A consequence of this behaviour of course is that programming efforts are duplicated and that everybody keeps even generally needed information on his personal file which then is either unavailable globally, or if the information also is maintained globally, most probably will be inconsistent with that information.

Connecting the computers into a network, as for example ARPA has been doing for a number of years, does not directly solve the problems since this network is not at all integrated when seen from the users point of view. He still has to know on which computer the required programs and data reside and which programming languages and commands to use to gain access to this information, in order to send input to a distant program, to execute it, and receive its output.

In a network containing many different computer systems the task of remembering even a few of these procedures becomes formidable and effectively restricts the users flexibility in the system.

Distributed processing and distributed data management try to eliminate these diversification problems for the user. Unfortunately I feel that already the name "distributed" has been chosen wrongly. It was selected in typical self-centred fashion by the system builders, because these systems look distributed when seen from their point of view, in other words, from the inside. Actually the whole purpose of these systems is to present a unified or integrated processing and data management system to the user. The user can work on the system without concern as to where the programs he is to execute are stored, where his data are kept and where ultimately the processing resulting from his work will take place. For him, the system looks as if everything would be located in his interactive terminal or at most the local computing facility.

Since we can view programs just as a special kind of data, and all executions as transactions on these data, an integrated system in a computer network will be provided if we construct a data base management system which accomplishes data manipulation and program execution in a consistent manner throughout the system. Such systems are called Distributed Data Base Management Systems (DDBMS) and at the University of Stuttgart we are building one of them, named POREL.

Summing up the most important characteristics of a DDBMS we arrive at

1. It works on a computer network which is formed by computers of many different types using many differing operating systems, that is, the network is inhomogeneous.

2. It presents a unified view on data and programs to each of the users sharing the system. The users never have to be aware where their programs and data reside and where the processes they start are actually executing. Using data manipulation requests which are identical to those of centralised data base systems the users can work with the

total distributed system.

As of this time no general purpose distributed data base management system is available from vendors, although users have already developed ad-hoc systems. Some vendors are even selling distributed file systems which they call distributed data base systems, but a deeper analysis usually reveals that they do not satisfy the above characteristics of a DDBMS. They are usually lacking either in data base functions, such actually representing distributed file systems, or do not present a unified view of data and programs to the user.

2. The Architecture of Distributed Data Base Management Systems

Distributed data base management systems of course contain many features and mechanisms that also appear in centralised DBMS's. In our investigations we shall concentrate however on problems which are of direct concern to the distribution and homogeneity aspects of a distributed system.

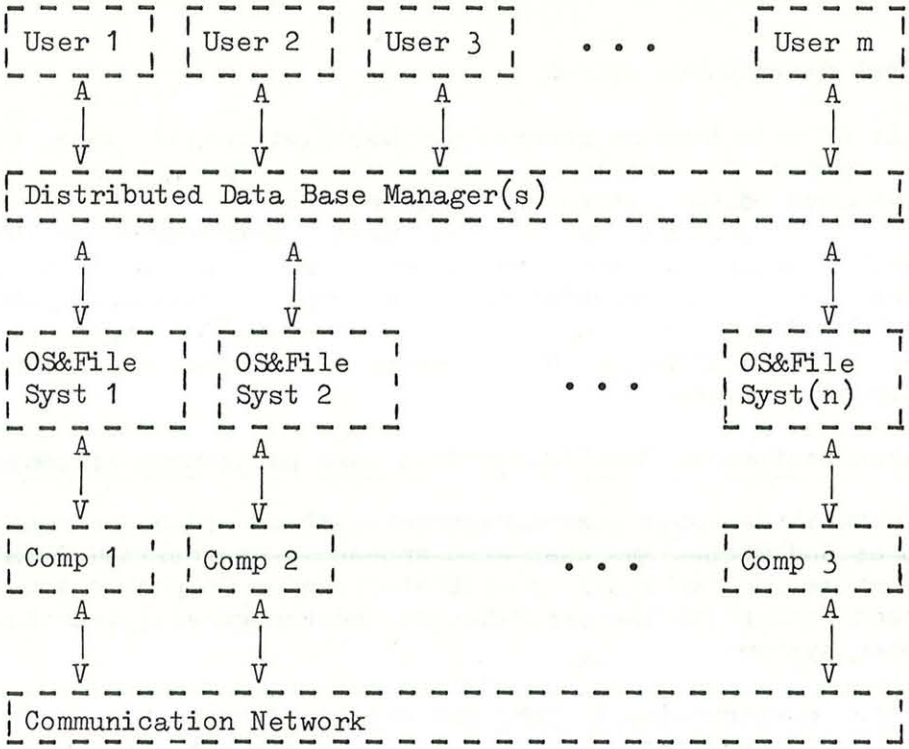
When constructing a DDBMS two principal approaches have to be distinguished depending on the planned environment of its use (1). They are illustrated in Figure 1, parts 1 and 2:

1. The homogeneous data base system

   Still under the assumption of an inhomogeneous computer network and differing operating systems, a single integrated data base management system is constructed, parts of which are executing on the different computers in the network.

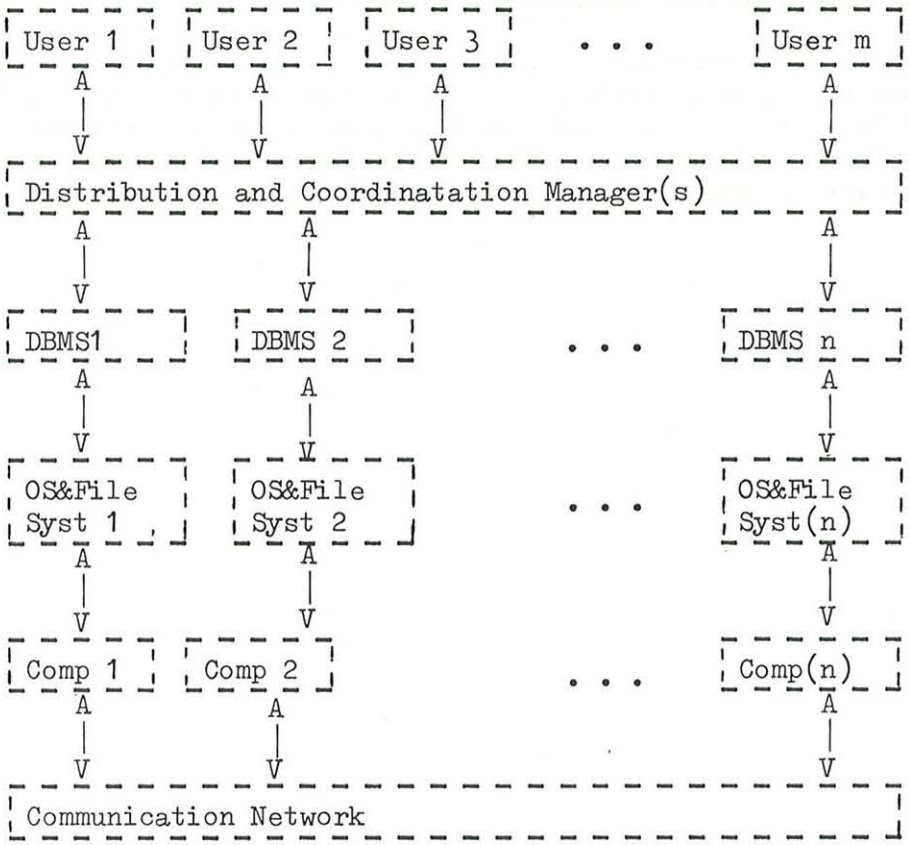
2. The inhomogeneous data base system

   Under the assumption that on each of the computers in the network a local (centralised) data base management system already exists a distribution and coordination system is built which makes these DBMS's available in a homogeneous fashion to the users of the network.



Homogeneous Distributed Data Base System

Figure 1: The Architecture of a Distributed Data Base Management System (part 1)



Inhomogeneous Distributed Data Base System

Figure 1: The Architecture of a Distributed Data Base Management System (part 2)

In both systems the user is presented with a unified and homogeneous view of the data and programs available in the network. In the first case no specific partitioning strategy is imposed a priori. Data and program may reside and may be processed where it is economically most feasible. One is also fairly free in choosing the data model and data representation mechanism most appropriate for the selected distribution strategies. Work in this area has been described in (2,3,4,5,6,7). Data which are already included in existing (centralised) data bases will have to be transformed and transferred into the new distributed system.

In the second approach already existing data bases will remain untouched. A translation and transformation mechanism has to be constructed to produce the unified user view. However the problem of translating user requests and data representations correctly is very difficult and at present not very well understood. As a consequence the translation mechanisms will be very complex and the transferability of data and programs in the network much restricted (8,9,10). The problem is somewhat simplified if one can assume that all the local data base systems are built according to a single architecture, for example CODASYL-DBTG (11). Actually such a system can be considered to stand in between approaches one and two depending on the level of general control data base administrators associated with a local system can execute.

Although contrasting in basic approach, the two strategies have a great deal of problems and concepts in common. They both have to ensure that the user is presented with a unified view of the whole system. They both imply the use of some communication medium and of some localised file handling mechanism. They also require the existence of some distributed executive supervising the total system operation. Both must pay attention to performance, reliability, and cost.

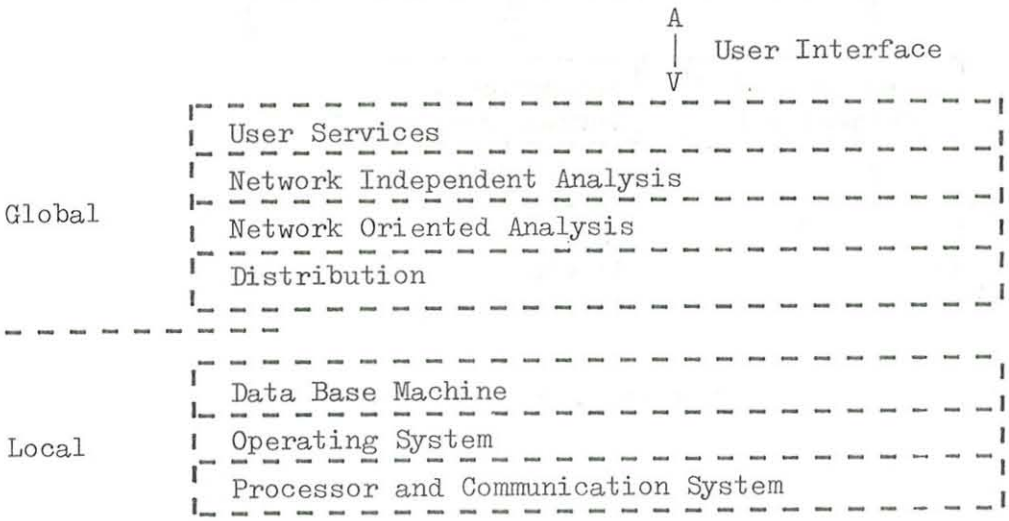


Figure 2: The Logical Structure of POREL

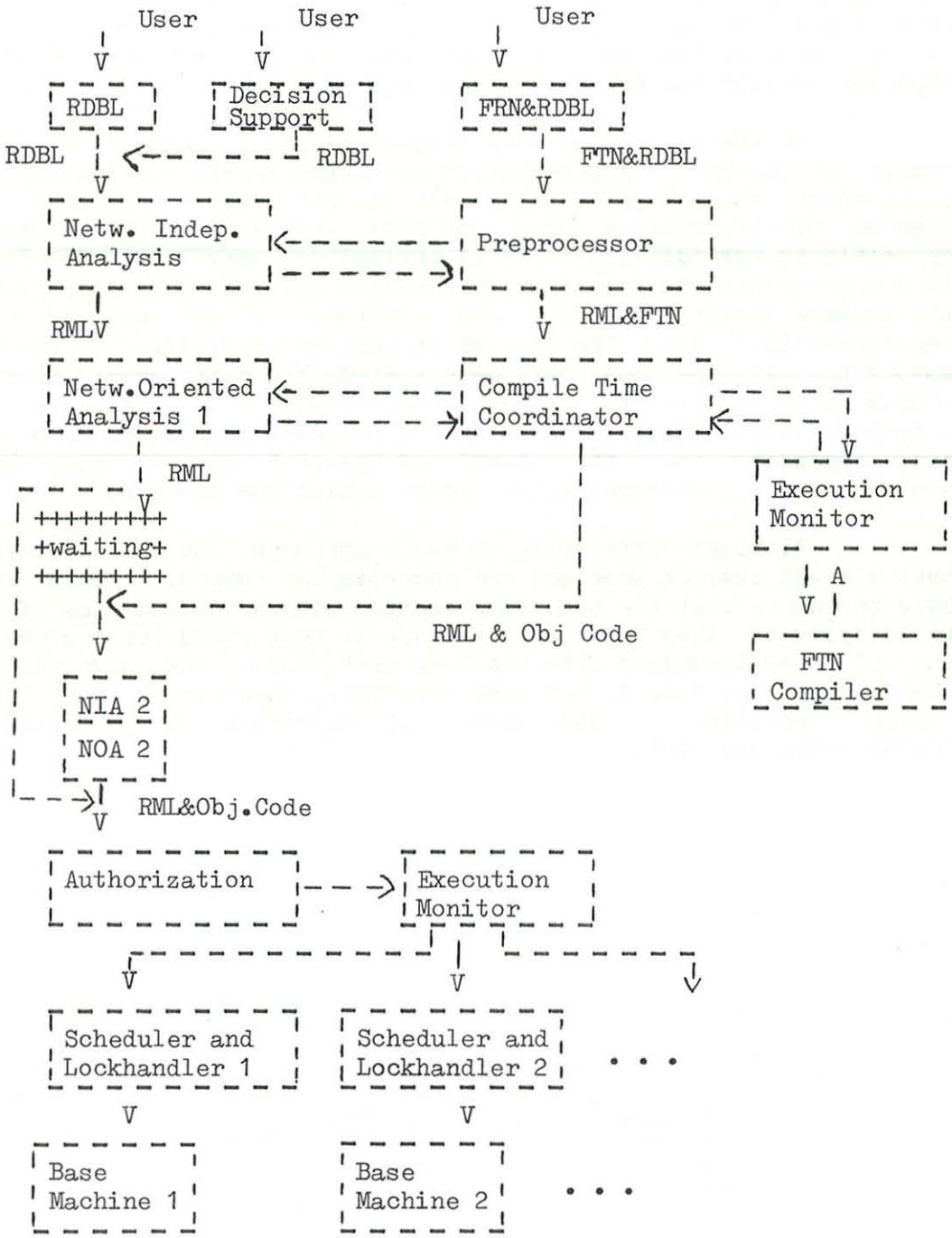


Figure 3: Process (Module) Structure of POREL

The questions which are of special importance for a DDBMS are

a) how and where to store data,
b) how to locate data,
c) how to control concurrent manipulation,
d) how to provide acceptable cost/performance,
e) how to provide reliability and recoverability,

and we shall investigate them in somewhat more detail in the framework of the POREL distributed data base management system (4).

In Figure 2 we show the logical structure of POREL whereas in Figure 3 a more process (module) oriented representation is given. The figures illustrate the system as it would be seen by a user, they do not represent the multi-user, multi process environment which actually exists. That is, more than one editor, more than one network independent analyser etc. may be executing at the same time. However each computer in the network contains only a single scheduler and lockhandler and is represented by a single (software implemented) base machine.

POREL has been designed around the relational data model as originally proposed in (12) and an algebra oriented data manipulation language.

POREL supports three interfaces for the user of the distributed data base:

1. A Relational Data Base Language (REBL); a nonprocedural, algebra-oriented, interactive language for data definition, data manipulation and control.

2. FORTRAN (or some other programming language) and RDBL as a data language, whereby RDBL has been extended with a cursor concept similar to the one found in SEQUEL 2 (15).

3. A problem solving, decision support system which provides the user with a working place oriented environment.

A detailed description of these features is beyond the scope of our current discussions. Many aspects of them are quite independent of the distributed network environment and the interested reader is referred to (13). In the later sections we shall use examples written in RDBL but we hope that the selected language features will be self-explanatory.

At the other end of the spectrum of POREL features is the computer network we are using. The network contains PDP11's and German minicomputers and one TR440 system, a system of IBM 360/65 size and technology. The computers in the network communicate via the X.25 interface (14) which was developed by CCITT (Consultative Committee on International Telegraph and Telephone) and will become the standard communication vehicle in the Federal Republic of Germany.

The different components of the POREL system will not be

discussed individually but only in connection with the distributed data base management features which they support and which will be described in the next sections.

3. Data Storage

In a distributed data base system there is a clear benefit in storing data at the processor site where it is most frequently used since

a) long distance communication is much slower than local access to data, and

b) communication costs are a very significant part of any network system and are directly dependent on the amount of information transmitted across communication lines.

If we would know the frequency of access for all data objects in the data base, an optimal allocation of the data objects to processor sites could be found which would minimise access time and network traffic. Unfortunately this problem has been shown to be NP-complete (16), and therefore today's best known algorithms are much too slow for practical applications. In addition the problem becomes even more complex if we distinguish between retrieval and update traffic as the latter always involves at least two transfers, a retrieval and then a replacement access. We may also gain advantage by storing more than one copy of a data object. This tends to reduce retrieval time but increases the time required for updates and also complicates the mechanisms which are to ensure the consistency of the data in the data base.

After the decision has been made on which processor site a data object is to be kept we have also to determine the storage technique to be used locally for the object. Because of the relatively long time required to read or write secondary storage devices it is important that data which are frequently used together are also placed "close" to each other on these storage media, for example in the same physical block, or in blocks where no mechanical actions are involved when accessing them together. Again the problem of finding an optimal placement is NP-complete and consequently no algorithm is available for practical applications.

Heuristic techniques have been used for both the site selection and data placement problem, and have been found to work quite well. Many of the techniques are based on mathematical programming optimisation models, some also introduce the notion of imperfect knowledge of access and update statistics (17). Another study (18) developed a very comprehensive model that distinguishes query and update traffic. The problem attacked is how to allocate copies of data objects to processors and also how to allocate communication bandwidth in order to minimise the combined storage and transmission costs. In addition the model allows to specify that the average access time to data is to be bounded by a designer supplied parameter. The heuristic technique employed for the evaluation hwoever still turned out to be too slow for large applications. Clearly more research is needed in this area of where and how to store data objects in a distributed data base system.

In POREL we have developed RDBL commands which allow the data base administrator (or an authorised user) to control the placement of data in the data base. To illustrate a few of these commands the relational data base is thought to consist of

```
supplier (sno, sname, city)
project (jno, jname, city)
supply (sno, jno, amount)
```

The DISTRIBUTE commands then may have the form

```
DISTR supplier
        LOC s1 FIX WHERE city = "New York"
        LOC s2 FIX, s1 OPT
                WHERE city = "BOSTON"
        OTHER TO LOC s3 FIX
DISTR project
        OTHER TO LOC s1 FIX
DISTR supply
        OTHER TO LOC s2 FIX, s3 FIX
```

describing a distribution as illustrated in Figure 4.

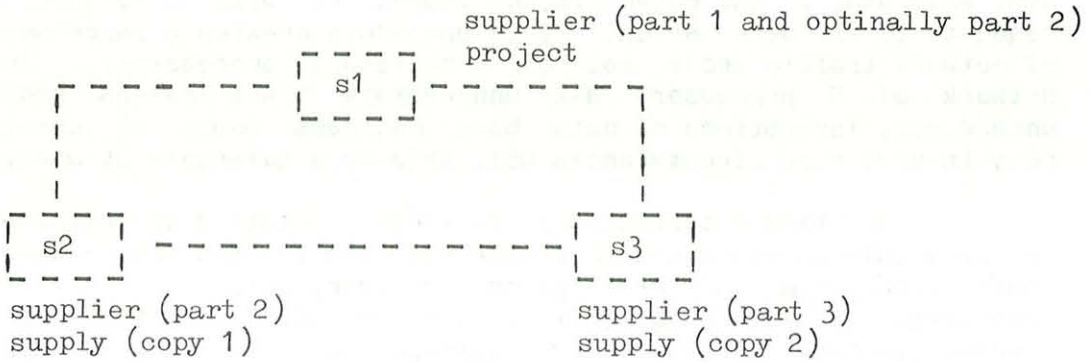


Figure 4: Data Distribution

To define the grouping of data objects on a single processor site a clustering technique for data has been implemented in POREL (19). A command to organise such a cluster for our sample data base may have the form

```
DEFINE CLUSTERSET supplying
    USING SUPPLIER (sno, sname)
          project (jno, jname)
    WHERE supplier.sno = supply.sno &
          supply.jno=project.jno
```

In our current implementation of POREL the above commands have to be given by the Data Base Administrator. In the future we plan to implement optimisation algorithms. The DBA can then use these programs and the resulting restructuring information to make decisions on the placement of data. In its final stage the placement and reorganisation of the data should be carried out automatically by the network oriented analysis (NOA) and the base machine mechanisms.

4. Locating Data

A request for a data object kept in a distributed data base system can originate from any of the processor sites. The management routines of that processor now have to know some way for finding the data object. Several alternative strategies for locating a data object are available and the selection of a strategy will depend on the size of the data base, the expected distribution of the data, and the frequency of reorganisation of the data base (3).

The simplest method is to store all descriptive information of the data, that is, the knowledge base, on a centrally located processor site. However much of the advantages of distributing the data now gets lost again as this site always has to be accessed before the actual data reference takes place. In addition a breakdown of this processor will inhibit all operations on the data base. For these reasons this simple algorithm is not actually acceptable.

Another simple approach is to store the descriptive information on the site where the data are physically kept. An accessing algorithm will check first whether the data are locally available and, if the object is not found, it will broadcast the request to all other sites. This procedure creates a large amount of network traffic and a lot of unnecessary processing. In a network of N processor, N-2 unnecessary transmissions and N-2 unnecessary invocations of data base managers would be created. Only in very rare circumstances will this be a tolerable strategy.

A third alternative is to store a detailed description of the data only at the sites where the data are located but to keep a short and compact description on every other site. This information should just be sufficient to provide information for the correct parsing of the input commands and for the network independent analysis of the requested operations. This analysis includes things like the static checking of correctness and consistency of the request, and optimisation mechanisms. In POREL we have chosen this third approach. As long as the compact description of the data base as it is kept on every network site is stable over longer periods of time, this solution produces the lowest network traffic of the three techniques presented.

After the requested data have been located in the network the network oriented analysis (NOA) of POREL will analyse the user transaction to determine how and where the request is to be processed. The algorithm employed is similar to the algorithms discussed in (20,25). A multivariable transaction thereby is decomposed into a sequence of one variable actions. Such an action then can be executed on a single location by moving all the relevant data to that site, or it can be moved to all the sites where (part of) the data reside which are identified by the single remaining variable. The network oriented analysis will decide which of these two solutions is preferable by the amount of network traffic it creates and the degree of concurrency which becomes feasible.

Let us assume for example a command to be issued on site s1.

```
ASSIGN SMITH_supplies DISPLAY ALL
        SELECT (supply .jno)
        WHERE (supply .sno=supplier.sno &
                  supplier.sname='SMITH')
```

For this query the algorithm then proceeds as follows:

1. Do the one-variable subquery

```
ASSIGN temp DISPLAY
     SELECT (supplier.sno)
     WHERE (supplier.sname = 'SMITH')
```

The network oriented analysis decides that the result most probably will be a unique sno (or at least only a very small set of such numbers if sname is not a candidate key of the relation supplier) and that the query will best be executed at all the sites where parts of the supplier relation reside. The execution monitor therefore will start such queries at the sites s1, s2, and s3 of our example. In the general case the result is a relation "temp" distributed on all three sites.

2. The original query now becomes

```
ASSIGN SMITH_supplies DISPLAY
     SELECT (supply.jno)
     WHERE (supply.sno = temp.sno)
```

In our example the supply relation resides one copy each on the two sites, s2 and s3. If we assume the size of temp to be much smaller than the size of supply the network oriented analysis will

a) select as the execution site that site out of s2 and s3 where the larger segment of temp resides,

b) create commands to send the other segments of temp to that site, and

c) create commands to transport the result relation SMITH_supplies to site s1.

The execution monitor will start the execution of the above query. After it has finished, the monitor will execute the proper transfer actions to place SMITH_supplies at the original requesting site (s1).

5. Concurrency Control

To maximise the concurrent use of system resources by many users, shared access to these resources has to be possible. A data object is "locked" by a process whenever the process has to be sure that the object is not in some transient state. As soon as locking is permitted the possibility of deadlock arises when two or more users each are trying to reach an object locked by the other. Locking strategies and deadlocks have been investigated quite extensively for operating systems and for centralised data base systems (22, 23, 24).

In distributed data base systems however deadlock control is made more difficult by the existence of multiple lock and concurrency managers (21, 26, 27). The problem we are faced with is to ensure that the data in the distributed data base remain consistent despite attempts at concurrent access and update from different processors. This consistency has to be ensured even if duplicate data exist in the system or (partial) system breakdown occurs. Of the referenced proposals only the paper by Rosenkrantz et. al. (27) deals with all of these aspects.

In general a user will specify a group of one or more RDBL commands which are closely related to each other. Thereby he constructs a semantically consistent portion of the total data manipulation he plans to execute. These command groups are termed transactions and, as seen from the user, they represent a semantically meaningful transformation on the data base. As a consequence the system has to ensure that during the execution of such a transaction the data base cannot be changed by parallel processes in a fashion which would destroy the meaning of the result expected by the user from his transaction. That is, concurrency and locking managers have to ensure the consistency of the data base throughout the processing of such a transaction.

Strict control, in other words no interference, can be very costly in a data base with many users. Therefore many data base management systems do allow a user (or data base administrator) to specify different levels of consistency, where the least restrictive one usually ensures only the well definedness of a data object during a single read or single update activity. The most restrictive will of course ensure that a person executing a whole transaction can look at the data base as if he would be the only user at that time.

To allow system control of concurrency and of locking, the network independent analysis (NIA) of POREL constructs a time graph relating time dependencies between the different actions found in the user specified transaction currently processed. The time graph reflects possible (network independent) parallelism which could be utilised and forms the basis for creating separate (partial) transactions and the necessary synchronisation and resource requests to execute these transactions properly. The network oriented analysis then updates this graph to reflect possibly created new partial transactions and to incorporate the new interdependencies and parallelisms which arise through the use of the network.

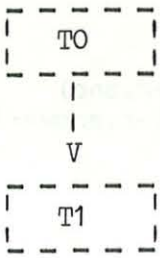
For the one-variable subquery of section 4

```
ASSIGN temp DISPLAY
    SELECT (supplier.sno)
    WHERE (supplier.sname= 'SMITH')
```

NIA would produce two partial transactions and a time graph as follows:

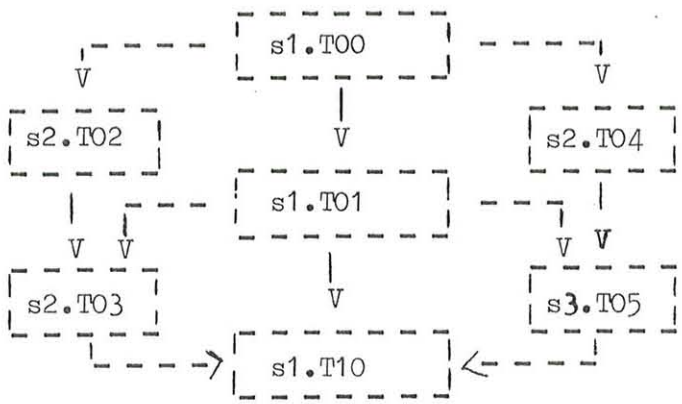
```
TO:  WAIT()
     R_LOCK supplier
     CREATE TEMPORARY temp
     ASSIGN temp
        SELECT (supplier.sno)
           WHERE (supplier.sname='SMITH')
     R_UNLOCK supplier
     END TO


T1:  WAIT(TO)
     DISPLAY temp
     DROP temp
     END t1
```

The network oriented analysis of POREL then uses the information which describes the distribution of the relation supplier to create partial transactions of, for example, the form



```
S1.T00:  WAIT ()
         R_LOCK supplier ON(s1,s2,s3)
         CREATE TEMPORARY temp ON (s1,s2,s3)
         END s1.T00 TO (s1,s2,s3)

s1.T01:  WAIT (s1.T00)
         ASSIGN s1.TEMP
            SELECT (s1.supplier.sno)
              WHERE (s1.supplier.sname='SMITH')
         R_UNLOCK s1.supplier
         END s1.T01 TO (s1,s2,s3)

s2.T02:  WAIT(s1.T00)
         ASSIGN s2.temp
            SELECT (s2.supplier.sno)
              WHERE (s2.supplier.sname='SMITH')
         R_UNLOCK s2.supplier
         END s2.T02 TO (s2)

s2.T03:  WAIT (s2.T02,s1.T01)
         I_LOCK s1.temp
         MOVE s2.temp TO(s1.temp)
         I_UNLOCK s1.temp
         END s2.T03 TO(s1)

s3.T04:  WAIT(s1.T00)
         ASSIGN s3.temp
            SELECT (s3.supplier.sno)
              WHERE (s3.supplier.sname='SMITH')
         R_UNLOCK s3.supplier
         END s3.T04 TO (s3)

s3.T05:  WAIT (s3.T04, s1.T01)
         I_LOCK s1.temp
         MOVE s3.temp TO(s1.temp)
         I_UNLOCK s1.temp
         END s3.T05 TO (s1)

s1.T10:  WAIT (s1.T01, s2.T03, s3.T05)
         DISPLAY temp
         DROP temp ON(s1)
         END s1.T10
```

and the time graph



The synchronisation between s1.TO1 and the transactions s2.TO3 and s3.TO5 ensures independent recoverability of the assign actions. If s2 or s3 would already send data before TO1 was finished, a failure of TO1 would require a reexectuion of TO1,TO2, and TO3 since it would be impossible to separate the data in temp which were the results of the partial execution of the assign command in TO1 from the data sent from s2 and s3.

During the execution of the MOVE commands the locking protocol ensures that the inserts into the temporary relation part on site s1 cannot interfere with each other. As can be seen by this example it is possible in POREL to dynamically request resources. In general we do not prevent deadlocks in our system but the resource request protocol reduces deadlock occurrences to only such circumstances where backout and recoverability has been found to be acceptable. If such failures seem unacceptable a deadlock prevention strategy for the involved processes is chosen. That is, the process is not started if it cannot be assured that it also can finish without becoming involved in a deadlock situation.

6. Cost/Performance

Distributed data base management systems will only be used in the praxis if one can demonstrate an improvement in cost/performance over the same application done in a centralised system. In special cases the added reliability and the "fail safe" properties of a distributed system may be of such importance that they outweigh strict cost/performance considerations, but in general cost/performance will be of very strong influence.

Many factors will influence the behaviour and the performance of data base systems. Unfortunately even for centralised data bases no comprehensive performance data are available or have been analysed. Much work remains to be done in this area to enable us to predict the behaviour of a data base system in a specific usage environment. Data base designers therefore try to include as many parameters as possible into their system to allow for adjustments to the specific requirements of a work environment even after the system has been installed there. Of course so far the responsibility for these adjustments mostly rests on the data base administrator and his ability to select proper storage strategies, clustering and data access techniques, and in network environments the proper distribution of data in the system.

When deciding on data distribution a data base administrator has to determine the cost of typical data manipulations expected in the system and has to try to minimise this cost. This same evaluation however will have to be done when individual queries are to be optimised by the data base management system and therefore the necessary algorithms have to be included in the system.

Whenever the DBMS is to process a user transaction it will use the existing system and data descriptions to minimise the cost of processing the transaction.

Optimisation techniques as they have been employed for

nonprocedural languages are applied by POREL during the network independent analysis.

Breaking up transactions into sub-tractions very often can be done in more than one way. The network oriented analysis of POREL uses the network description to minimise the total cost of executing a transaction. Again only heuristic techniques are available for doing so but the experience with some of them in the centralised data base environment (25) has been encouraging enough to allow predictions about their usefulness in distributed systems.

In a network environment the cost of processing a (sub) transaction can be formulated as

c1*network-traffic + c2*total-processing-time

To analyse this formula and to minimise its values we restrict ourselves to the simple example of a n-variable query.

Let us assume there are

N sites in the network, and
n relations referenced in the query.

We assume that whenever the query is to be processed only one of the relations, Rp, is allowed to remain distributed, all others will be completely available at the processing site(s). We now have to determine

K as the number of processing sites of the query,
Rp as the relation to remain fragmented,
R as the segment of the relation Rp residing on site j

The communication cost of moving relations to the K processing site can be calculated as follows

$$\text{comm cost} = \sum_{j=1}^{K} \left(c_{K-1} * \left(\sum_{i \neq p} |R_i^j| \right) \right) +$$

$$\sum_{j=K+1}^{M} \left(c_{K-1} * \left(\sum_{i \neq p} |R_i^j| \right) + c_1 * |R_p^j| \right)$$

where $|R|$ denotes the size of a relation (fragment) and c (x) the cost of sending x bytes of data to K sites, a cost that depends on the network that is used. The first part of the formula reflects the cost of sending relation fragments from a processing site to the K-1 other processing sites. The second part reflects the cost of sending fragments of the relations Ri, i p, from non-processing sites to processing sites and of sending a fragment Rj, j>K, to any one of the processing sites.

The analysis of the processing costs of a query becomes more complicated due to the fact the processing time in general will not be a linear function of the size of the data processed. In addition, system overhead will usually be more severe for small processes than for large ones. Since an analysis of these time

dependencies requires detailed knowledge of the data base design we shall not further discuss it in this overview oriented paper.

To illustrate the communication cost formula let us however investigate our earlier query

```
ASSIGN SMITH_supplies DISPLAY
    SELECT (supply.jno)
    WHERE (supply.sno=temp.sno)
```

The relation temp resides on all three processors in our network. The supply relation is not distributed but a copy exists on s2 and also s3. We now can distinguish the following cases.

1. Execution sites: s1, s2, s3

   Distributed: temp

   $\underline{comm}\_c = c_1 * (|2.supply|)$

2. Execution site: s1

   Distributed: supply

   $\underline{comm}\_c = c_1 * (|s2.temp| + |s3.temp| + |s2.supply|)$

   (Note: either s2.supply or s3.supply can be chosen)

3. Execution site: s2

   Distributed: supply

   $\underline{comm}\_c = c_1 * (|s1.temp| + |s3.temp|)$

4. Execution site: s3

   Distributed: supply

   $\underline{comm}\_c := c * (|s1.temp| + |s2.temp|)$

If we assume $|temp| < |supply|$ then one of the cases 3 or 4 should be chosen depending on the sizes of s2.temp and s3.temp. These results confirm the intuitive site selection we have done in section 4.

7. Reliability, Recoverability, Integrity

Reliability, recoverability and integrity are closely interrelated properties of any data base management system. No system and especially no distributed system will ever be free of breakdowns or partial breakdowns. Some processor may fail, external storage devices may fail, or communication lines may not work. In a data base system it is especially important to keep the influence of such failures as local as possible, that is, increase the reliability of the total system by leaving as many users as possible unaffected by such a breakdown. After the cause of the failure has been found and eliminated a recovery process for the data base system has to take place. Processors that may have become separated from the network have to be integrated again, processes that may have been finished only partially will have to be completed or backed out. In all these actions it is very important that both the integrity of the data in the data base and the

integrity of the program executions remain ensured. For example, if a user transaction results in updates of parts of the data base which are located at different processor sites the updates have to take place at all these sites or at none of them. Proper recovery and backout mechanisms have to ensure this behaviour of the system. However this problem is not a simple one.

Suppose for example that a transaction updates three data objects each stored on a different processor site. The system now ensures via a commit-strategy (30), that none of these three updates is in effect until all have been completed and acknowledged. Assume now that the originating site of the transaction has received all three acknowledgements and sends out the commit message, but one of the three processors goes down before it has put the update into effect.

The problem is closely analogous to the problem of data transmission protocols where correctness through messages alone cannot be absolutely guaranteed (28). A general mechanism for recovery which is based on checkpointing the data base and keeping action-traces will resolve this problem as long as checkpoint data and traces do not get lost. Unfortunately such a mechanism is very time consuming and uses up a large amount of storage space. In addition actions which produce an effect outside of the computing system can only be handled with great difficulty. For example, if a check already has been printed and mailed with a wrong value, how do I get it back? These recovery problems however are not restricted to the distributed data base environment but also exist in centralised systems.

As it is very often the case in data base systems a tradeoff between function (in this case data integrity) versus cost has to be made.

8. Conclusions

Distributed data base management systems have been investigated intensively during the last two years, but much work remains to be done. To be manageable these investigations usually concentrate on a specific aspect of the system and make many (restrictive) assumptions about the other system properties.

In actually constructing a DDBMS, as we are currently doing in Stuttgart, it is very difficult to integrate current knowledge into a single system since many of the results found in the literature are too restrictive and sometimes make quite contradictory assumptions about the behaviour of the system. In many situations simple minded (and usually slow) solutions can be found and by heavy modularisation of POREL we should be able to replace such "solutions" by better ones found through our own research or by others.

Acknowledgement

The POREL distributed data base project could not exist without the close cooperation of all the researchers involved in the project. Thanks are due to these people and especially to H. Biller and U. Fouser who have guided the project during my absence from Stuttgart through many difficult times.

Discussion

Professor Dijkstra: You should store the data in a distributed system close to the point at which it is most urgently required, rather than that at which it is most frequently required.

Professor Page: These are two fundamentally different criteria. For urgency you must guess or forecast where it will be required, whereas most frequent need can be determined from past use.

Professor Neuhold: Even if you know the precise requirements on frequency of access, the placement problem of where to allocate the data, and what bandwidth communication lines to allow, is NP-complete. I think therefore that we cannot allow the distribution to be carried out automatically, but should allow the database administrator to specify a distribution initially, and adjust it later according to the usage of the data.

Mrs. Ringland: I think that you are never going to get away from having to design your system intelligently. For example, taking the problem of producing a total of values spread across the whole database, you are never going to be in the position of wanting a user to sit there and do that on demand, no matter how much processing power you may predict. Implicitly there is some sort of serial nature to the problem. You are going to have to think ahead, which comes back to the need to do some sort of offline, housekeeping or background procedures in order to produce this sort of information.

Professor Neuhold: I think I agree with you. In general however, you have a dynamic system, you can't think ahead for the system. In some limited systems it may be possible, for example in an airline reservation system where the number of different transactions is very limited. In some such limited systems you may be able to pre-program, at least for a specific class of users of the system.

Mrs. Ringland: I disagree with you somewhat here in that one thing that requires intelligence is anticipating which will be the sort of thing that you have to do in advance, because they would take impossible processing times to produce on demand.

Professor Neuhold: I think that preplanning such systems is a very hard problem.

Professor Dijkstra: Is preplanning in contrast to post-planning? (laughter).

Professor Neuhold: They are rather closely linked, in fact, because you somehow have to undo the effects of all the things that have gone wrong!

References

1) Deppe, M.E., Fry, J.P., "Distributed Data Bases: A Summary of Research", Computer Networks, Vol. 1, No. 2, September 1976.

2) Rothnie, J.B., Goodman, N., "An Overview of the Preliminary Design of SDD-1, A system for Distributed Data Bases", Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, 1977.

3) Stonebraker, M., Neuhold, E., "A Distributed Database Version of INGRES", Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, May 1977.

4) Neuhold, E.J., Biller, H., "POREL: A Distributed Data Base on an Inhomogeneous Computer Network", Proc. of the Conference on Very Large Data Bases, Tokyo, 1977.

5) Epstein, R., Stonebraker, M., Wong, E., "Distributed Query Processing in a Relational Data Base System", Proc. of ACM SIGMOD 78, Austin, May 1978.

6) Schneider, H.J., Munz, R., "1977 Report on the Distributed Data Net Project (VDN)", Report No. 1/78 Inst. fuer Angewandte Informatik, T.U. Berlin, 1978.

7) Peebles, R., Manning, E., "A Computer Architecture for Large (Distributed) Data Bases", Proc. of the Conference on Very Large Data Bases, Brussels, September 1976.

8) Adiba, M., Caleca, J.Y., Elizet, C., "A Distributed Data Base System Using Logical Relational Machines", Research Report USMG, Grenoble, 1978.

9) Nahouraii, E., Cardenas, A.F., Brooks, O., "An Approach to Data Communication between Different Generalised DMBS", Proc. of the Conference on Very Large Data Bases, Brussels, September 1976.

10) Voss, K., "A Distributed Inhomogeneous Data Base System", Gesellschaft fur Mathematik und Datenvararbeitung, Birlinghoven/Bonn, 1977 (personal communication).

11) CODASYL Systems Committee, "Distributed Data Base Technology - An interim report", Proc. of ACM SIGMOD 78, Austin, 1978.

12) Codd, E., "A Relational Model of Data for Large Shared Data Banks", C. ACM, Vol. 13, No. 6, June 1970.

13) E.J. Neuhold (et. al.), "POREL Design Specifications: System Design", Institut fuer Informatik, University of Stuttgart, 1978.

14) Solmonides, C., "X25 Interface Definition", National Physics Laboratory, Oxford, 1977.

15) "Chamberlin, D.D., et. al., "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control", IBM Journal of Res. and Dev., Vol 20, No. 6, 1976.

16) Eswaran, K., "Placement of Records in a File and File Allocation in a Computer Network", Proc. IFIP Congress, Stockholm, 1974.

17) Levin, K.D., Morgan, H.L., "A Dynamic Model for Distributed Data Bases," Proc. ORSA/TIMS Conference, 1975.

18) Mahmoud, S.A., Riordan, J.S., "Optional Allocation of Resources in Distributed Information Networks", ACM TODS, Vol. 1, No. 1, 1976.

19) Poschik, S., "A Portable Relational Interface for the Distributed DBMS POREL", Proc. of the GI Fachtagung on Data Bases in Minicomputer Nets, Karlsruhe, 1978.

20) Wong, E., "Retrieving Dispersed Data from SDD-1: A System for Distributed Data Bases", Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, 1977.

21) Chu, W.W., Ohlmacher, G., "Avoiding Deadlock in Distributed Data Bases," Proc. ACM National Conference, November 1974.

22) Coffman, E.G., Elphick, Shoshani, A., "System Deadlocks", Computing Surveys, Vol. 3, No. 2, 1971.

23) Eswaran, K.P., Gray, J.N., Lorie, R.A., Traiger, T.L., "The Notion of Consistency and Preicate Locks in a Database System", C. of ACM, Vol. 19, No. 11, November 1976.

24) Lomet, D.B., "Multi-Level Locking with Deadlock Avoidance", IBM Research, RC7019, 1978.

25) Wong, E., Youssefi, K., "Decomposition Strategy for Query Processing", ACM Transactions on Database Systems, Vol. 1, No. 3, September, 1976.

26) Goldman, B., "Deadlock Detection in Computer Networks", Thesis Mass. Institute of Technology, June 1977, 84p.

27) Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M., "A System Level Concurrency Control for Distributed Database Systems", Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, 1977.

28) Morrissey, J., "Distributed Processing Systems - Some Economic and Technical Issues", Proc. of a Distributed Processing Workshop, ACM Computer Architecture News, Vol. 5, No. 6, 1977.

29) Lampson, B.,Sturgis, H., "Crash Recovery in a Distributed Data Storage System", Internal Report, Computer Science Laboratory, Xerox, Palo Alto Research Center, 1976.

30) Blasgen, M.W., "Issues in the Design and Implementation of Data Base Management Systems", Proceedings of AICA 1977, Invited paper, Pisa, 1977.